AGARD-CP-68-70

AGARD

ADVISORY GROUP FOR AEROSPACE RESEARCH & DEVELOPMENT

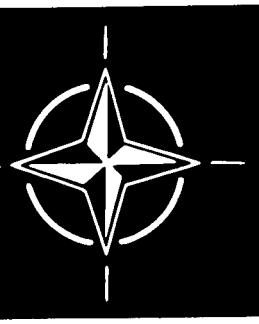
7 RUE ANCELLE 92 NEUILLY-SUR-SEINE FRANCE

AGARD CONFERENCE PROCEEDINGS No. 68

on

The Application of Digital Computers to Guidance and Control

NORTH ATLANTIC TREATY ORGANIZATION



DISTRIBUTION AND AVAILABILITY
ON BACK COVER

AGARD Conference Proceedings No.68

NORTH ATLANTIC TREATY ORGANIZATION

ADVISORY GROUP FOR AEROSPACE RESEARCH AND DEVELOPMENT

(ORGANISATION DU TRAITE DE L'ATLANTIQUE NORD)

THE APPLICATION OF DIGITAL COMPUTERS TO GUIDANCE AND CONTROL

Papers presented at the 10th Meeting of the Guidance and Control Panel of AGARD,
held in London, England, 2 - 4 June 1970

Panel Chairman	:	Professor C.T.Leondes (US)
Panel Executive	:	Major C.D.Mount, USAF
Program Chairman	:	Dr G.E.Roberts (UK)
Program Co-Chairman	:	Dr G.E.Knausenberger (US)
Local Conference Coordinator	:	Mr D.B.Smith (UK)

CONTENTS

	Page
FOREWORD	iii

SESSION I - GENERAL REVIEW OF COMPUTER SYSTEMS AND OUTLOOK FOR THE FUTURE

	Reference
KEYNOTE PAPER: OVERVIEW OF AEROSPACE VEHICLE COMPUTER APPLICATIONS by Dr R.K.Smyth	1
TRENDS IN THE APPLICATION OF DIGITAL COMPUTERS TO GUIDANCE AND CONTROL by W.H.McKinlay, M.A.V.Matthews and R.Wright	2
SYSTEMS TASKS FOR THE DATA PROCESSING EQUIPMENT OF ADVANCED AIRCRAFT NAVIGATION SYSTEMS by Dr F.G.Unger and R.S.Sindlinger	3

SESSION II - COMPUTER OPTIONS AND ORGANIZATION ASPECTS

FEDERATED VS. INTEGRATED COMPUTER SYSTEMS by J.H.Crenshaw	4
PROGRAMMING CHARACTERISTICS OF FUTURE GUIDANCE AND CONTROL COMPUTERS by A.J.Maher	5
AEROSPACE COMPUTER WORDLENGTH CONSIDERATIONS by G.W.Braudaway	6
THE CASE FOR SPECIALISED SYSTEM PROCESSORS IN AIRBORNE INSTALLATIONS by P.A.Hearne	7

SESSION III(a) - AIRCRAFT NAVIGATION AND CONTROL

ECONOMICS OF INTERFACING WITH SMALL SENSOR COMPUTERS by K.R.Brown	8
FLIGHT RESEARCH EXPERIENCE WITH GUIDANCE AND CONTROL COMPUTERS RELATED TO GENERAL APPLICATIONS by M.E.Burke	9
DIGITAL COMPUTING ASPECTS OF THE JAGUAR NAV/ATTACK SYSTEM by D.E.Humphries	10
THE EFFECT OF DIGITAL COMPUTER LIMITATIONS AND SENSOR ERRORS ON THE ACCURACY OF AN AUTOMATIC BOMB RELEASE by G.Schroeter	11

SESSION III(b) - AIRCRAFT FLIGHT CONTROLS

RECENT ADVANCES IN SELF-ORGANIZING AND LEARNING CONTROLLERS FOR AERONAUTICAL SYSTEMS by Dr C.W.Gwinn and R.L.Barron	12
HIGH INTEGRITY DIGITAL FLIGHT CONTROL by Dr J.F.Meredith and K.A.Helps	13
REALISATION OF NONLINEAR CONTROL METHODS WITH DIGITAL CONTROL UNITS by Dr W.Sobotta and H.J.Berger	14
DIGITAL DATA PROCESSING IN AUTOMATIC FLIGHT CONTROL SYSTEMS by R.W.Howard	15

SESSION III(c) - SPACE AND MISSILE APPLICATIONS

	Reference
THE USE OF A DIGITAL COMPUTER FOR THE ELDO INERTIAL GUIDANCE SYSTEM by M.J.W.Gage	16
THE STAR COMPUTER: A SELF-TESTING AND REPAIRING COMPUTER FOR SPACECRAFT GUIDANCE, CONTROL AND AUTOMATIC MAINTENANCE by A.Avizienis	17
SENSITIVITY ANALYSIS OF APPROXIMATE OPTIMUM GUIDANCE PROCEDURES FOR ON-LINE OPERATION by E.Gottzein and H.Bittner	18
REAL-TIME DATA PROCESSING AND ORBIT DETERMINATION ON THE APOLLO INSTRUMENTATION SHIPS by Dr F.C.Johnson	19

SESSION IV - OVERALL GUIDANCE ASPECTS

DISPLAY AND INTERFACE IMPLICATIONS IN THE USE OF DIGITAL COMPUTERS by J.G.Carr	20
THE TESTING OF COMPUTER-EQUIPPED GUIDANCE AND NAVIGATION SYSTEMS by M.G.Jaenke	21
FAULT ISOLATION IN A DIGITAL GUIDANCE AND CONTROL COMPUTER by D.H.Blauvelt	22
SPACEBORNE COMPUTER FOR ATTITUDE CONTROL AND DATA HANDLING by P. van Otterloo	23

OVERVIEW OF AEROSPACE VEHICLE COMPUTER APPLICATIONS

by

Dr. Richard K. Smyth, President, Ralph E. Manns Co. (USA)

Gordon H. Smith, Chief Scientist, Autonetics (USA)

SUMMARY

During the 1950's, many debates arose among the system designers of avionics for aerospace vehicles on the topic: "should this function be mechanized with analog or digital techniques?" The proponents of the digital mechanization side of the debate won a few of the arguments and airborne digital computers were introduced. These early applications of aerospace digital computers occurred only where accuracy of computation was an overriding factor in the system design criteria. In all other instances, analog mechanizations won hands down. Specifically, digital mechanizations first proved desirable and feasible for performing the computations associated with inertial navigation and guidance and for weapon delivery of free-fall ballistic weapons. In the case of the third generation ballistic missiles developed in the late 1950's, the flight control function rode along on the coattails of the inertial guidance system and, thus, digital flight control systems were born.

During the 1960's, dramatic things happened to the technology of digital computers. Microelectronic Integrated Circuits (MIC's) and microminiaturized magnetic core memories were introduced. This new silicon technology gave birth to a new generation of airborne digital computers which is characterized by low cost per bit, small volume per bit, and small weight per bit of memory, and high speed. Typical parameters of these computers are for a total machine cost of $0.20 per bit of programmable memory, 200 bits/in^3, and 400 bits/oz or some 340,000 bits of memory in a one cubic foot volume, weighing about 50 pounds. The tremendous computational capability of these airborne machines made the many debates of analog versus digital computation obsolete. In systems with complex mission computation functions, digital computation was much cheaper than analog computation and that settled the debates.

This is not to say that the introduction of large capacity digital computers into aerospace applications did not bring a number of problems. The problems of digital computer hardware faded into the background to be replaced with new problems of using the computers - how do you communicate with the displays, the actuators, the sensors and the pilot; all of which are analog devices? - how do you program the computer to do all the things you are now asked it to do? - how do you discover when the computer is malfunctioning; before it causes a catastrophic loss of mission capability (or the vehicle itself)? All of these problems had to be addressed and solved by the system engineers.

The powerful vehicle-borne digital computers, coupled with the ingenuity of a new generation of engineers raised on modern control theory, have made feasible the widespread use of Kalman filtering applications to guidance and control. When Dr. Rudy Kalman first introduced his optimum filtering concepts in the mid-1950's, the computers required to implement the concept were almost too large to get in a room - much less into an airplane or missile. Advanced computer technology has changed all of this. The current MIC computers use only a small fraction of their capacity of Kalman filtering and perform many other important mission functions as well.

The new generation of solid state technology promises to introduce another round of revolutionary digital hardware that will have as great-if not greater-impact on systems mechanization as did the MIC computer. The metal oxide semiconductor (MOS) devices are now being produced in production quantities. The MOS circuits will give rise to a new round of debates for the 1970's - should we use general purpose computers or special purpose computers? The answer shall be most probably that we should use <u>both</u>. The MOS technology will make special purpose digital computers and devices feasible that were not otherwise. For example, consider the impact on the system designers thinking if 1024 bit MOS shift registers could be mass-produced at $1.00 per device. This would make the cost of memory 0.1 cents per bit! The application of this type of memory to such functions as display scan converters and radar doppler processing would result in many system breakthroughs in the area of target acquisition systems.

The new computer technologies allow the system engineer to implement almost any mission functions which he can conceive of limited only by the capabilities of the sensors, displays, and human operators.

OVERVIEW OF AEROSPACE VEHICLE COMPUTER APPLICATIONS

Dr. Richard K. Smyth
Gordon H. Smith

I. INTRODUCTION

During the 1950's, many debates arose among the system designers of avionics for aerospace vehicles on the topic: "should this function be mechanized with analog or digital techniques?" The proponents of the digital mechanization side of the debates won a few of the arguments and airborne digital computers were introduced. These early applications of aerospace digital computers occurred only where accuracy of computation was an overriding factor in the system design criteria. In all other instances, analog mechanizations won hands down. Specifically, digital mechanizations first proved desirable and feasible for performing the computations associated with fire control, inertial navigation and guidance and later for weapon delivery of free-fall ballistic weapons. In the case of the third generation ballistic missiles developed in the late 1950's, the flight control function rode along on the coattails of the inertial guidance system and, thus, digital flight control systems were born.

During the 1960's, dramatic things happened to the technology of digital computers. Microelectronic Integrated Circuits (MIC's) and microminiaturized magnetic core memories were introduced. This new silicon technology gave birth to a new generation of airborne digital computers which is characterized by low cost per bit, small volume per bit, and small weight per bit of memory, and high speed. Typical parameters of these computers are for a total machine cost of $0.20 per bit of programmable memory, 200 bits/in^3, and 400 bits/oz or some 340,000 bits of memory in a one cubic foot volume, weighing about 50 pounds. The tremendous computational capability of these airborne machines made the many debates of analog versus digital computation obsolete. In systems with complex mission computation functions, digital computation was much cheaper than analog computation and that settled the debates. In these systems which use high capacity digital computers, system designers found that the application of the computer went far beyond the benefits of accuracy. With all system data in digital form, such mission functions as mode switching, failure detecting and warning, and status indication became simple and inexpensive to implement within the computer.

This is not to say that the introduction of large capacity digital computers into aerospace applications did not bring a number of problems. The problems of digital computer hardware faded into the background to be replaced with new problems of using the computers - how do you communicate with the displays, the actuators, the sensors and the pilot; all of which are analog devices? - how do you program the computer to do all the things you are now asking it to do? - how do you discover when the computer is malfunctioning; before it causes a catastrophic loss of mission capability (or the vehicle itself)? All of these problems had to be addressed and solved by the system engineers. These problems are discussed in subsequent sections of this paper.

The earliest application of airborne digital computers was in the early 1950's. The first digital computer to fly was used for fire control computations. It had 250 vacuum tubes and was four cubic feet in size. The technology in two decades has exploded to the point where a much more sophisticated computer can be built in a few cubic inches instead of a few cubic feet. The impact of this galloping technology can best be seen in Figure 1, where computer characteristics are compared on a per equivalent component basis to get differing technologies into a common basis. The chart shows that in the decade between 1960 and 1970, the equivalent cost of computers was reduced by a factor of eight, the reliability was improved by a factor of twelve, power requirements were reduced by a factor of four, weight was reduced by a factor of ten and volume has been reduced by a factor of 100. These significant improvements show the transition from vacuum tube to transistor to integrated circuit technology. As startling as these improvements are, the projections for the next few years show that the application of large scale integrated circuits is going to double these gains in many areas.

At the same time that the computer technology was improving, one finds that the actual size of the digital computers did not change a great deal. The reason for this apparent anomaly was that the application of the computer to many additional functions was developing at the same time. The early computers were relegated to fire control functions or to inertial navigation functions with some possible steering and fixtaking capability included. The next generation of computers added doppler, doppler inertial, optimal filtering, and deadreckoning to the navigation function, added weapon delivery functions and added sensor utilization functions for pointing and designation and also incorporated the necessary interfaces with controls and displays for these new functions. Since the system computation became more complex, sophisticated executive routines and self-test routines for both the computer and system were incorporated.

The following generation of computers will add radio navigation and more sophisticated optimal filtering concepts in order to optimally combine the information from all navigation sensors. This next generation of computers will also be concerned with the countermeasures function both for threat identification functions and also for selecting the most appropriate countermeasures for the particular threats encountered. The integration of computer technology will go much deeper into all the facets of the system to include communication, in-depth system testing, damage assessment, premission planning and postmission evaluation functions and will provide sophisticated failure circumvention procedures. During the same period of time as this third generation of computer applications, special-purpose capabilities are being expanded to include multiplexing, digital filtering, and learning and self-organizing functions.

The growth in vehicle-borne computer requirements over a fifteen year period in terms of input/output requirements, memory capacity and computational speed is summarized in Figure 2.

II. COMMUNICATING WITH THE COMPUTER

Everyone who has ever used a digital computer for any purpose discovers the "catch" with computers - it is communication. A computer is useless unless one feeds it good data in a form it can assimilate, extracts the data and transforms it into a usable form to be used by the system action elements. The communication problem for aerospace vehicle application has many facets including analog to digital (and vice versa) conversion of signals, protection against spurious noise, protection against inadvertent short circuits, determination of appropriate sampling rates, matching of impedance characteristics, and detecting errors during transmission of digital information within the system.

The hardware required to perform the signal conversion function for a digital system can be as complex and as costly as the computer itself. The conversion requirements can be simplified if the sensors, displays, and actuation elements are designed to be compatible with digital systems. The signals which require conversion include D.C. voltage, A.C. voltage, synchros, and digital discretes control signals. The cost of the conversion equipment is sensitive to the accuracy required of the converted signal and to the specific characteristics of the signals themselves. It is a particularly gruelling task to completely specify all signals and their associated parameters. However, this is a task which must be done early and must be done before the signal conversion equipment design can be completed. It is necessary to specify a number of signal parameters for each signal including voltage range, phase shift, voltage reference, phase reference, voltage drive power, and required tolerances. Many of these parameters are not known or specified by the equipment manufacturers.

A significant part of the signal conversion problem is tying down the signal interface definition including all signal characteristics. Changes in the signal characteristics have a significant impact on the design of the conversion equipment circuitry. Therefore, signal interface control documents have to be developed and imposed upon the equipment manufacturers to insure that changes are not indiscriminantly introduced which would seriously impact the accuracy of the system. Many of the signal parameters are dependent upon the electrical cable characteristics. Therefore, it is important that the transmission characteristics of the cable be specified and controlled, including cable line impedance. The agency installing the vehicle cabling must purchase cable with the specified parameters controlled.

The system will normally include a number of subsystems which transmit and receive data in digital form. These subsystems provide the analog to digital and digital to analog conversions internally within the subsystem. A convenient and frequently used method of digital data transmission is through the use of multiplexed, serial digital lines from the sensors, displays, and actuators. Such transmission can be by modulation of a carrier or base band (dc) modulation. The serial digital signal must be converted to a form compatible with the computer control processor. This conversion is quite simple, however. It is important in the transmission of digital data to indicate to a computer complex that the data it is receiving is valid and that errors have not been introduced in the transmission. Such indication can be achieved by using a parity bit to indicate that the data transmitted is valid. Other codes allow error correcting.

The interface between the vehicle crew and the computer is an important consideration also. It is imperative that a simple means be provided for the crew to communicate with the computer, but it is equally imperative that the computer memory be protected against inadvertant erasures of critical memory areas - particularly those areas of memory containing program instructions or stored, invariant parameters. There are a number of simple memory protection techniques and one of these should be used. The man-machine interface should allow the aircrew to enter mission dependent data in flight. Also, the aircrew should be able to interrogate the computer to obtain specific

information. Provision should be made for the ground crew to enter specific mission data through a ground prepared tape. Typical data for a tactical avionic system may include data such as the type of munitions being carried for weapon delivery computations, and the route points, destinations, and targets for the navigation computations. Normally, such mission dependent data is so extensive that it would be too time consuming to enter the data manually. The implications for the equipment needed to prepare the mission data tapes are apparent.

Future vehicle systems and subsystems should be designed such that the requirement for signal conversion by the computer complex is minimized. Conversion of synchro signals is particularly costly in terms of circuit complexity. The design requirement for the future is clear - eliminate synchros to the extent possible and design the vehicle subsystems such that all parameters are converted to digital form and transmitted to the computer complex in serial digital form.

III. PROGRAMMING THE COMPUTER

As the role of the vehicle-borne computer expands, the task of producing associated software in a manner responsive to constantly changing flight requirements increases in difficulty. Software production on a tight schedule is possible today only by employing large numbers of people with no single person having a detailed familiarity with the entire program. Therefore, future research in flight software development will be directed at solving the enormous information and communication problem which necessarily arises from such circumstances.

Considerable emphasis is being placed on the development of higher order languages to simplify the task of preparing operational flight programs. Although this should help solve an important aspect of the problem, no amount of programming elegance can alleviate the systems engineering task of assuring that the software specification is complete and accurate or that the documentation is adequate, uniform, and readily understandable.

It is generally agreed that generating the code for a flight program represents only a small portion of the total development cycle. The really difficult and time-consuming job is program validation. Since much of the validation task is clerical in nature, high-speed digital data processing can and should be more effectively employed. Future requirements will demand, for example, the development of such ingenious tools as automatic program analyzers to supplement the use of digital and hybrid flight simulators.

In a vehicle-borne system, using digital computers, the cost of the software to implement the system functions is a major system cost. Because of the importance of the software, software specifications must be written and change control procedures utilized to assure that the software production does what is required and is adequately documented for posterity. The need for these kinds of controls and the increasing complexity of the computer function in the system have caused an escalation of programming costs as shown in Figure 3. The shaded area shows a tremendous variation in projected software costs. This variation depends upon whether or not implementation of higher order languages similar to FORTRAN or COBOL for the real-time vehicle system computations is successful. If the past process of instruction-by-instruction programming is used, the cost will continue to escalate with the complexity of the computation. If a higher order language is successful, the cost curve can turn down work and significant dollar savings can be achieved.

Since higher order languages have been developed for every major scientific and business computer for the last few years, one may ask: "Why have the aerospace computers been so slow in seizing upon the advantages of writing programs in pseudo languages?" The answer is that generation of higher order languages for real-time processing is extremely difficult. First of all, in order to be able to install a computer aboard an aircraft or a space vehicle, stringent requirements have been put on its physical size, its weight, and the amount of power available. Consequently, designs of these machines have utilized the minimum practical number of instructions which means that the instruction list is generally very simple. Furthermore, the amount of memory capacity has been limited to the minimum amount initially forecast as required, plus a safety factor so that very efficient utilization of memory capacity is required to implement the initially planned system program. Typically, most vehicle-borne computers have absorbed the safety factor early in the development process and, in fact, the program requirements usually exceed the capacity of the memory by 25 percent or so at some point during the development process. This phenomena occurs since the programming job has always been tougher than predicted and people have always found a few dozen extra things for the computer to do that were not planned when the design was committed to hardware. Consequently, a great deal of effort goes into inventing clever mechanizations so that everything that ultimately is required can be squeezed into the memory.

The second thing which makes higher order languages difficult to implement for real-time control functions typical of aircraft and space vehicle applications is the fact that the functions really are real time - the computations are part of closed loop control systems and all the concerns for system stability, etc., must be considered. Furthermore, various functions must be performed within equal intervals of time so that the numerical integration processes for navigation, for instance, can be performed successfully. Typically, the programming complication arises and is compounded since, at the same time that velocity is being integrated to obtain position information, radar pointing commands may be required from the computer along with a still different coordinate transformation which generates a symbol on a headup display to show the predicted target location regardless of the attitude of the aircraft. The real-time control aspects involved with each of these types of functions force rigid timing and sophisticated executive functions to be implemented so that every computation function is performed within the time period which is required. This whole timing process is further complicated by the fact that a man may flip a switch which requires rescheduling to incorporate a new function he has commanded with other functions already in process. These timing requirements force very precise control of the timing of every flow path through the computer to be considered with every practical combination and practical sequence of functions to be performed and, consequently, greatly complicate the programming process.

A third factor which complicates generating higher order languages for the aerospace computers is the fact that the technology has been galloping so rapidly that any single computer developed may be utilized in relatively few different system applications before it becomes obsolete and is superseded by a more advanced unit. Consequently, it has been less expensive to generate the programs directly for each application than it would be to generate a higher order language and then apply it to a specific application. Usually, any additional application of the computer occurs after the first application is too far underway to consider generating a higher order language even though savings in both applications might have been made if the effort had been started initially.

As a result of the above factors, real-time aerospace programming is in about the same state that scientific programming was ten to fifteen years ago. The number of skilled programmers is very small, probably only the order of not more than 100. They are highly skilled inventive programmers who because of the constraints imposed by aerospace computers have developed sophisticated techniques for squeezing the last bit of computational capacity out of the computers.

As new and more powerful flight computer hardware is evolved, the logical design phase must emphasize much more careful attention to the associated software problems than has been evidenced in the past. For example, recovery from intermittent hardware failures cannot be left entirely to the software without adding immeasurably to the problems of insuring the integrity of the flight program. One should give careful attention to the selection of the computer instruction list so that a balanced optimization is achieved between the computer hardware and software tasks.

IV. COMPUTER SYSTEM OPERATION VERIFICATION

There are several levels of computer testing which are required in a system application of digital computers. These include premission testing of the computer to verify that it is operating properly, during mission testing of the computer for identification of malfunctions which may require circumvention techniques in order to successfully complete the mission, during mission use of the computer for testing the operation of other equipment within the system, and finally, tape verification testing which is used to assure that the computer is performing the proper computations at system level. The first three elements of test occur over the complete life of the system from development through the operational phases. The tape verification test procedure is required during the development phase and when any modifications are made to an operational system.

The self-test procedure is used to thoroughly exercise the arithmetic and control registers, the memory section and the input/output section of the computer by executing all the instructions in a variety of conditions in order to detect malfunctioning components. The amount of memory capacity required and the amount of computational time required increase significantly as higher and higher levels of confidence that the computer is operating properly is required. This is shown in Figure 4.

When there was just one computer aboard a vehicle, the testing performed during the mission was minimal since in most applications very little could be done except abort when a failure occurred. However, as additional computers are added, and reconfigurable organizations are implemented, the computer test concurrent with mission operation activity becomes significantly more important.

The prerequisite for switching to a backup mode or alternate computation path is the establishment that there is some difficulty with one of the computational black boxes in the primary or current computational organization. This same prerequisite is required at the system level to select appropriate data from alternate sensors which may provide similar information. This system level function also complicates the testing procedure. Current applications may require 20 percent of the computing capacity to be utilized for mission oriented testing. Because the great advances which are expected in self-healing or self-reconfiguring system organizations will further complicate testing procedures, more significant effort is to be expected in developing mixtures of hardware and software approaches to facilitate testing during mission execution.

In the early days of aerospace vehicle computer applications, the tape verification testing was a difficult chore and methodology was developed to insure the computer would operate in its anticipated environment. It has become necessary to simulate the dynamic data to be expected in an operational vehicle with a static computer system and verify that all of the software paths operated in the static environment as if they were operating in a vehicle performing its various mission phases. It has become necessary to use auxiliary computers - sometimes analog, sometimes digital, and sometimes both - to simulate the dynamic environment and it is necessary to simulate interfaces and equipments which were not immediately available to the computer integration job.

V. SYNERGISTIC COMPUTER APPLICATIONS

The powerful vehicle-borne digital computers, coupled with the ingenuity of a new generation of engineers raised on modern control theory, have made feasible the widespread use of Kalman filtering applications to guidance and control. When Dr. Rudy Kalman first introduced his optimum filtering concepts in the mid-1950's, the computers required to implement the concept were almost too large to get in a room - much less into an airplane or missile. Advanced computer technology has changed all of this. The current MIC computers use only a small fraction of their capacity for Kalman filtering and perform many other important mission functions as well.

A Kalman filter implements two different functions. The first is a statistical error analysis of the system using all available a priori knowledge of system error dynamics and error source statistics. The second function is the estimation of the system errors included in the model using weighted observations of some of the errors. The weighting matrices required for the estimation function are obtained from the error covariance matrix computed by the error analysis portion of the filter. In most applications, part or all of the error estimates are then used to correct the system variables.

Kalman filtering has been used extensively in aircraft navigation systems which use a variety of navigator sensors and fixtaking means. A Kalman filter is simply a systematic way of mechanizing in a computer the good judgment of a good human navigator. The Kalman filter uses measurements from independent sources of reference information, estimates the probable errors from these sources, and then weights the position from the references inversely to the magnitude of the probable errors, and finally makes the best estimate of the position fix.

High powered airborne computers have given a tremendous impetus to strapdown inertial systems. The coordinate axis transformations which are so easily accomplished in the digital computer will allow the elimination of the heavy, costly, and, sometimes, unreliable electromechanical gimbals of inertial systems. With some additional advances in strapdown sensors, the new generation computers should make strapdown systems competitive even in high accuracy applications.

Self-test and repair of computers is an important function both for long space probe missions and manned aircraft. The specifics of such computer functions are discussed in detail in subsequent conference papers. The self-test and repair function is performed at the computer hardware level as well as at the system mode level. Such mechanizations involve functional redundancy which includes both software redundancy as well as hardware redundancy. In addition to self-test and repair applications for in-flight operation, self-diagnostic algorithms have been developed which provide for isolation of failures to a replaceable module for ground maintenance operations.

The conference papers discuss a number of other imaginative applications of computers to aerospace vehicles, including nonlinear control algorithms, attitude control algorithms, vertical velocity obtained from Kalman filtering of barometric and inertial rates of descent, automatic flight control system computation, achieving a high degree of system integrity through multiple subchannels with switch-over in event of failures, interfacing with small sensor computers, self-organizing and learning algorithms, interface with small sensor oriented computers, and interface with CRT displays.

A most important synergistic effect of computer systems in aerospace vehicle applications is the availability within the computer in digital form virtually all system parameters and variables such as position, velocity, attitude, time-to-go, angles and ranges from the vehicle to targets as determined by sensors, and mode status informations. The availability of all this mission critical information allows the synthesis of new mission modes and functions to be developed after the hardware is all delivered by changes to software. The new computer technologies allow the system engineer to implement almost any mission function which he can conceive - limited only by his imagination and the capabilities of the sensors, displays, and human operators.

VI. OUTLOOK FOR 1970's

In looking forward to the application of digital computing techniques, during the next decade, we can see that the growth of applications will be even more startling and more exciting than it was in the last decade. The capability inherent in the large scale integrated circuit which allows thousands of active devices comprising hundreds of logic gates to be combined in a single circuit device provides computational functions at a cost ridiculously low when compared with the last decade. Computers comparable in capacity with anything flying today can be built in a size approximating a large matchbox. Such computers will require less than 10 watts of power and be priced in the range of the cost of an automobile. Whether that cost is that of a Rolls Royce or a Volkswagen is yet to be determined but, nevertheless, the cost for computing capacity will be significantly reduced. Furthermore, the small size, the low power and the low cost of this next generation of computers will allow computer organizations to be implemented which formerly were impractical but highly desirable. Computer organizations are now practical which will allow each functional area, such as navigation, or weapon delivery, or countermeasures, or sensor processing to have its own dedicated computation capability, and, consequently, allow a digital black box approach to system integration much in the same way that analog black box systems were formerly integrated. This provides the software flexibility and precision of the digital approaches to be combined with the independency and simplicity of the analog approach. Furthermore, the sophisticated processing which can be done with computers dedicated to specific functions may allow simplification in the sensor hardware and may extract more useful information than was formerly possible.

In the early aerospace application of digital computers, many kinds of computations were processed through a single arithmetic center because that was the most economical way to do the job. As the systems became more complex, the software problem associated with running everything through one funnel was difficult. This, combined with the concern that if a single computer failed, that all capability in the system was lost, led people into a multicomputer or multiprocessor organization so that some back-up capability would still be available. With the advent of the ability to use many computers without exorbitant penalties in cost, weight, power, or volume, organizations with many processing paths and ability to self-heal or reconfigure around malfunctions will be not only very attractive, but also very practical. These kinds of organizations are essential for the long term, deep space probe type of missions requiring successful operations for years without interruption.

The application of large scale integrated circuit devices will also provide another direction to aerospace computing in the next decade and that will be the use of special-purpose computers coupled into the sensing or display subsystems. In fact, many of the old tradeoffs which were made in deciding to go from analog computing to digital computing techniques will have to be re-examined to decide if the function should be implemented in a general purpose computer or if a few special-purpose digital devices should be used. Several companies have commercially available circuits which provide a digital differential analyzer integrator function. These integrators can be interconnected to provide coordinate transformations, multiplication, division, and square root functions, as well as the ability to be interconnected to directly and continuously solve differential equations. Another approach to special-purpose computations provides circuit chips which perform multiplication, addition, and delay functions which can be interconnected in such a manner to perform digital filtering for either bandpass or notch filtering, spectral analysis, difference equation solution, match filters, radar filters, digital flight control, and other similar functions. Tradeoffs will need to be made comparing the cost, flexibility, computer rates, etc., for each potential application before a final decision can be made. In any event, the next decade will see applications of digital computers and digital computing techniques into areas that were impossible to even contemplate in the last decade.

Figure 1. Computer Characteristics Per Equivalent Component

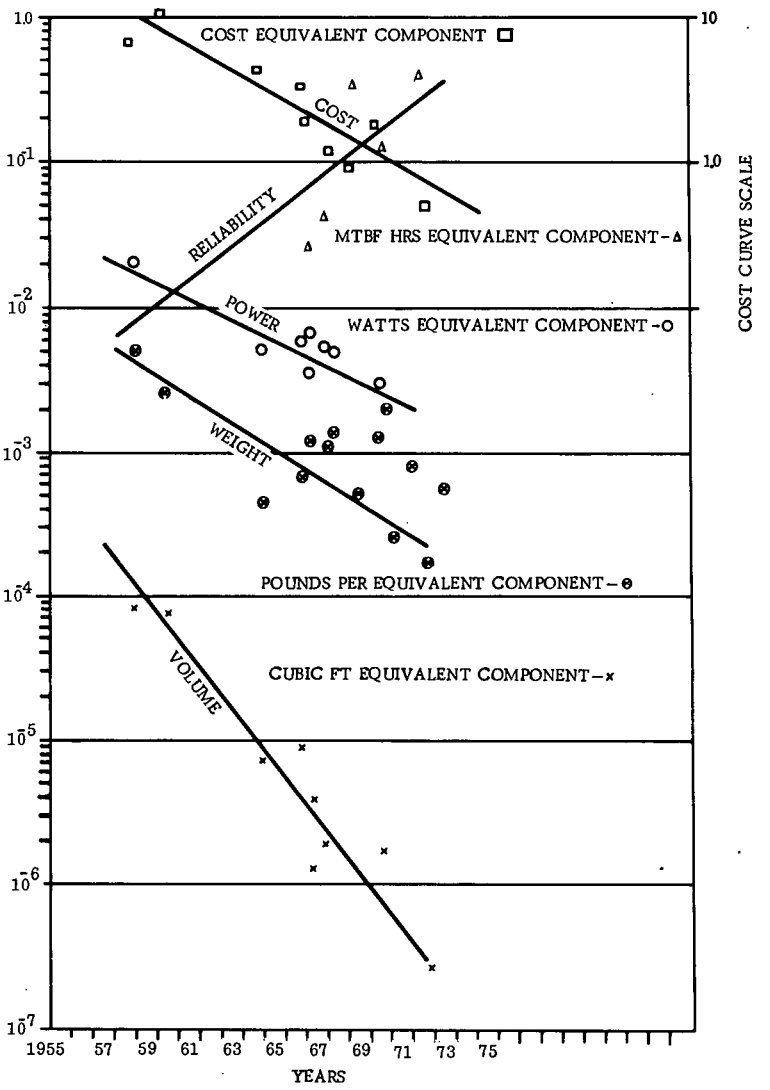


Figure 2. Computer Requirements

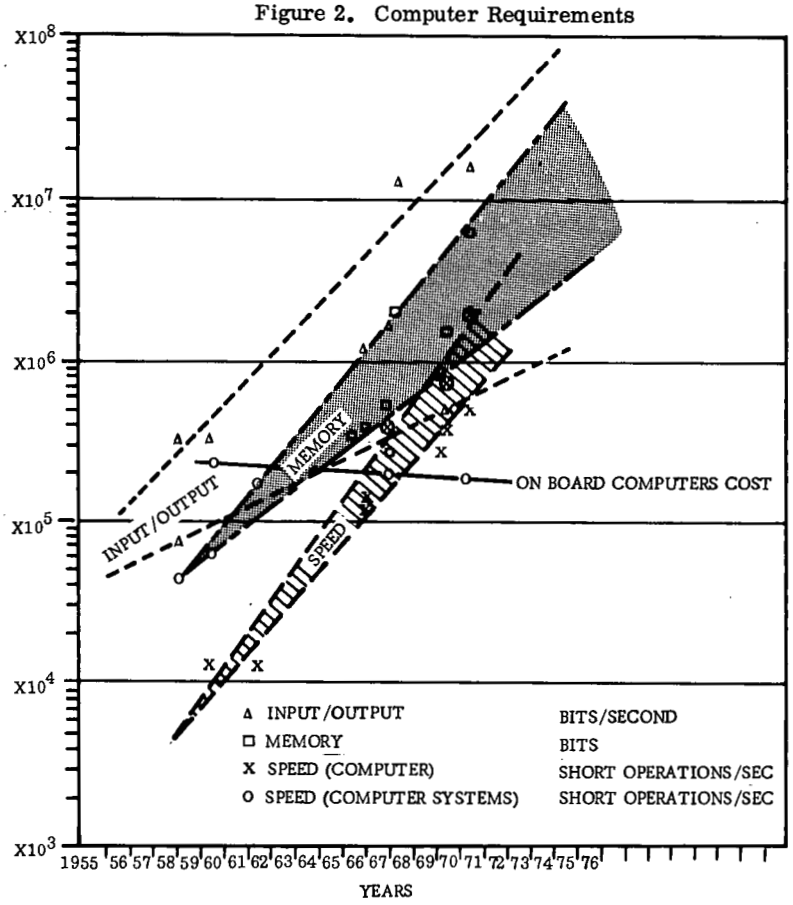


Figure 3. Dollar Programming Costs

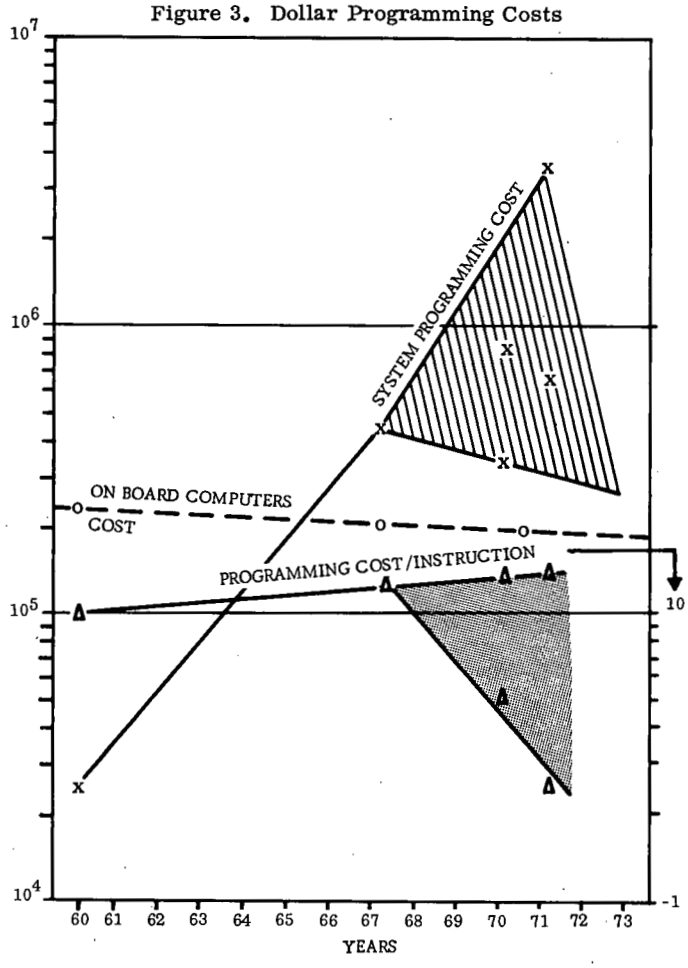
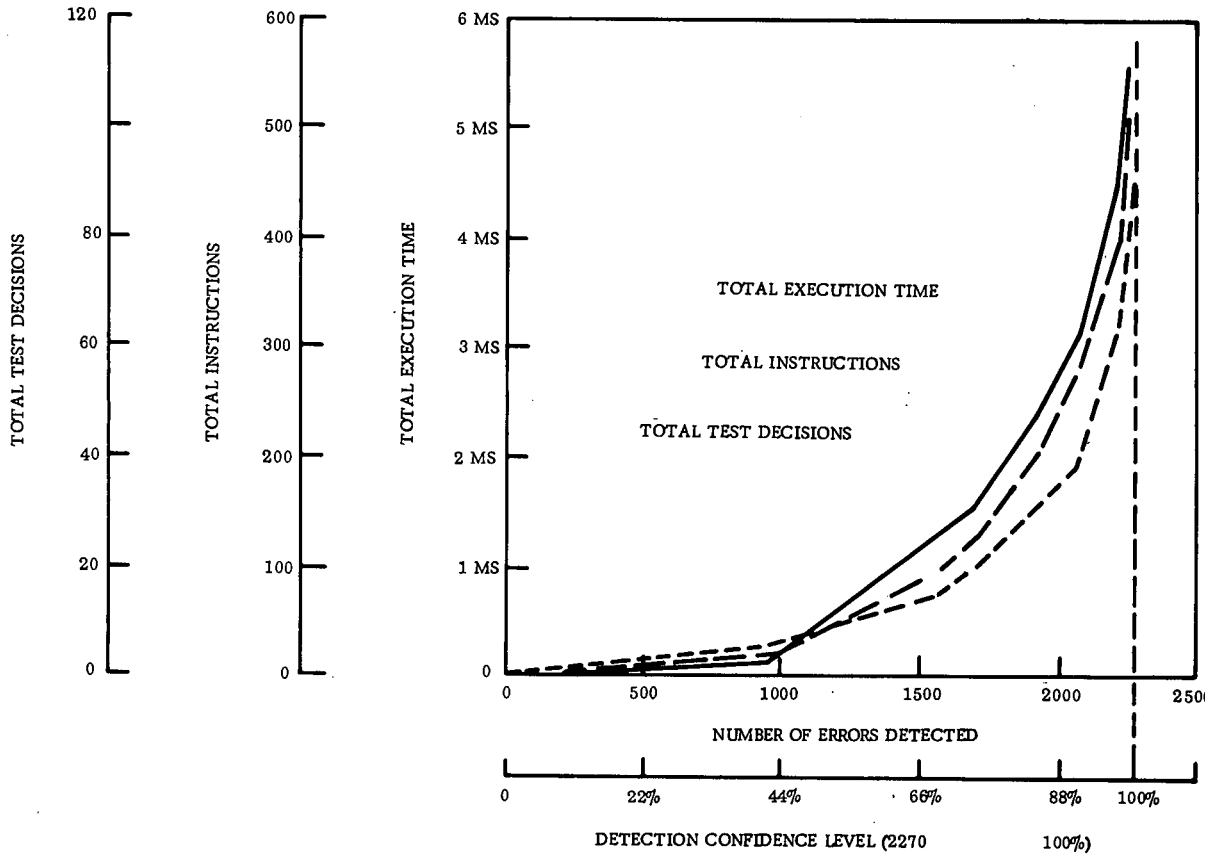


Figure 4. Detection Confidence for a General Purpose Processor Self-Test



TRENDS IN THE APPLICATION OF DIGITAL COMPUTERS TO GUIDANCE AND CONTROL

by

W.H. McKinlay, M.A.V. Matthews and R. Wright

Ferranti Limited

SUMMARY

This paper looks historically at the growth of the total guidance and control system requirement, primarily in manned aircraft. It touches on Flight Control, the significance of displays as a man-machine interface and the growth of navigation from a human operated task to one demanding processing and automatic computation.

Having dealt briefly with specific aspects of these requirements which indicate advantages in mechanising them using airborne digital computers it touches on the effect of progressing from analogue to digital techniques.

Some of the system options available in designing a digital system for an aircraft are summarised and machines which have been used in experimental work or as the basis for studies are described. The paper deals with the problems of input and output control and describes the Ferranti "S" interface developed for use with the FM 1200 computer. Reference is made to the use of sensor processing techniques as opposed to central computing.

TRENDS IN THE APPLICATION OF DIGITAL COMPUTERS TO GUIDANCE AND CONTROL

by

W.H. McKinlay, M.A.V. Matthews and R. Wright, Ferranti Limited

Introduction

In all areas of advanced technology there are two processes of evolution which can be viewed separately but which interact to product what is sometimes called "the state of the art". The first is the growth of operational needs which present problems requiring technical solutions. The second is the evolution of the technologies available to meet the needs. Neither of these two processes is continuous and their rates of progress do not match. To some extent they proceed independently, becoming locked together whenever a major project results in their implementation as hardware.

If they are mismatched, for example if an operational requirement more exacting than the available technologies is tackled or if the technologies are applied without due regard to the operational requirements the result is an unsatisfactory equipment or system. Clearly the problem becomes even more interesting if there is a sudden spurt in the operational or technological areas. It is for this reason that the present increasing application of digital computing techniques to guidance and control is of particular interest. There has been a massive spurt forward in technology and whereas many operational problems were unsolved 20 years ago because adequate technologies did not exist the problem now is often the reverse. It is necessary to decide very precisely how to use powerful digital techniques to solve operational problems if systems are not to become too complex and too expensive. To some extent the question is not "how much should we do" but "how little must we do".

The advent of digital technology has been accompanied by a rapid development of microminiaturisation and integrated circuit techniques which have the potential to lead to much higher standards of reliability than were available a decade ago. This feature, together with the trend to approach problems of safety or mission accomplishment in statistical terms, is leading to a much more sophisticated approach to the problems of redundancy and reversion in system design. The aim of this paper is partly to look at some of these general trends and partly to illustrate them by reference to techniques and equipments which are available today and which can serve as examples around which to illustrate some of the points made.

Growth of Operational Requirements

The aircraft is the first vehicle available to man which is capable of freedom of movement in three dimensions anywhere over the surface of the earth. In the early days of flying this fact was appreciated because it was at the centre of the dominant problem of achieving adequate manual control. The early European pioneers favoured an inherently stable aircraft which tended to fly straight in level without human intervention but was relatively unmanoeuvreable, and all their efforts were aimed at imparting increased manoeuvreability. In contrast the Wright Brothers had realised from the start than an aircraft which was basically unstable had much greater potential freedom of manoeuvre provided that adequate flying controls could be developed and full use made of the potential skill of the human pilot. That the Wright Brothers had made a correct judgement was apparent when their aircraft was first demonstrated in Europe and it was realised that they had succeeded by solving a basic problem which is still at the heart of all new developments in aviation: the man/machine interface.

The basic information required for control was primarily visual, full use being made of the human beings ability to detect both position and rate of displacement relative to a visual horizon. Gradually flight instruments were developed to provide longer term references and eventually these advanced to the stage at which sufficient redundant instrumental information was available for manual flight without an external visual reference. At the same time aircraft designers concentrated on producing adequate control characteristics by tailoring the design of their aircraft to meet handling requirements which were based primarily on historical knowledge of pilot skill.

The idea of using instrumental and control engineering techniques to increase the stability of an aircraft while preserving its manoeuvreability appeared quite early in the history of aviation and there is in our company archives some interesting correspondence between our founder, Dr S.Z. de Ferranti, and some of the early European aviation pioneers. In general the idea of using gyroscopes and other devices failed to catch on and a common response from the aircraft constructors was that they were too busy making aeroplanes to consider such developments.

Ultimately as the range and endurance of aircraft increased the problem of pilot fatigue and the necessity to reduce workload emerged, with the result that the Automatic Pilot was developed. The technologies were not very advanced but they had had to await the appearance of an operational need created by increased aircraft performance.

In those early days navigation was akin to the marine art of pilotage. It was concerned largely with map reading and the use of local knowledge. As aircraft flew longer distances and in particular as Trans-Oceanic flights were attempted the traditional techniques of marine navigation were adapted to aerial use. Long range navigation was as much an art as a science and was based on a combination of dead reckoning with discontinuous position fixing provided by astronomical and other means. Navigation was traditional guidance was provided by basic flight instruments and short term control characteristics were a function of the aircraft's characteristics. There was very little integration between these areas except as provided by the human operator.

Clearly any successful integration of navigation aids and automatic flight control depended on the transmission of information round the aircraft and had to await the application of electronics. The first electrical autopilots capable of accepting guidance inputs became available during the Second World War and subsequently, and had been preceded by the first navigation aids to produce electrical outputs defining

the deviation from a planned track. The ILS system was one of the earliest examples.

With the appearance of electrical technology and primitive analogue computers it became possible to tackle some of the more difficult operational problems which had hitherto been unsolved. These include:-

Precise all-weather navigation for civil aircraft in a defined air traffic control system.

All-weather long range navigation and target-finding capabilities in military aircraft.

Accurate automatic flight control capable of flying the aircraft along defined paths in space without frequent pilot intervention.

Operations in increasingly lower weather minima.

In the 1950s the need to solve these problems more effectively and the advent of much faster aircraft with much more sophisticated operational roles led to a further important landmark in the development of operational requirements. Up to that time navigation, guidance and control systems had been required to carry out tasks which would normally be within the capability of the human pilot given sufficient information. As confidence in the new technologies developed operational requirements emerged which were beyond the ability of the human operator but capable of solution by electronic means, the most notable of these being automatic landing in conditions of extremely low visibility. This led to a new concept in the use of redundancy in aircraft systems. Whereas redundancy had previously been provided for use at the pilot's discretion it now became necessary to develop systems which would themselves handle failure modes and reversionary procedures following failures and this is exemplified in the various configurations of failure survival autopilots.

In the military field a parallel situation exists in the use of terrain following radars for low level flight, where equipment failures can create hazards which cannot be contained by the human pilot.

It is interesting to note that nearly 50 years elapsed from the early days of aviation to the widespread acceptance that instrumental and control engineering techniques could be used as an essential part of the basic airframe in order to impart the required handling characteristics. It took the advent of the swept wing high subsonic or supersonic aircraft to gain full acceptance of autostabilisation as a means of creating acceptable handling characteristics, and even now the use of electrical control signalling in conjunction with autostabilisation is only just being recognised as a valid solution to handling problems which would otherwise involve unacceptable penalties in structural weight or mechanical complexity and reliability.

At the same time the appearance of systems carrying out tasks clearly beyond the capability of the human operator has refocused attention on the man-machine interface, the significance of which was originally recognised by the Wright Brothers. The contrast is that the interface is becoming much more complex as electronics are deployed to permit aircraft to carry out more complex missions or operate in more complicated environments such as the ATC systems which we expect to see in the 1970s and the 1980s.

Evolution of Technology

From the days of the first electronic and electrical autopilots the gains achieved by improving technology were primarily in terms of increased performance and lower weight and volume. It was some time before the operational requirements and the complexity of the systems reached a level at which technological deficiencies became a major source of embarrassment. However by the late 1950s a number of recurring problems had appeared, including among others:-

The difficulty of filtering noisy radio guidance information without introducing system lags having comparatively severe operational penalties.

A tendency for complex systems to impose a heavy workload on the crew because much of the logic in mode selection between one flight phase and another had to be supplied by the human pilot.

An inability to solve the navigation problem satisfactorily because the nature of the computations involved made any available computers complex and unreliable.

An increase in complexity in automatic pilots due to the need to employ such techniques as gain scheduling or disturbance compensation in order to achieve precise performance.

The difficulties of mechanising multiple redundant failure survival automatic flight control systems in the presence of the high drift and tolerance levels generated by analogue components in an airborne environment.

The heavy maintenance load imposed in complex analogue systems because of these drift tendencies and their inherent unreliability.

The excessive cable weight associated with integrating analogue equipments which were fundamentally incapable of employing multiplexing or wire sharing techniques without undue complexity and unreliability.

The need for excessive amounts of flight test time to optimise analogue flight control systems and the difficulty of incorporating modifications to control laws found necessary during development.

The appearance of the digital computer in a form suitable for airborne use occurred at a time when these technical problems and growing operational requirements threatened to produce a plateau in the development of guidance and control systems. With hindsight it is possible to see that the potential of the digital computer was oversold in its early days. For example its reliability was not as great as had been predicted in some quarters and some of the early systems proposed were much more expensive than their analogue counterparts, although at their initiation the reverse claims had been made.

The claim that problems of system optimisation and modification could easily be dealt with because of the flexibility of software was not as well founded as it at first appeared to be, and evidence to this effect exists in all the systems which ran out of computer capacity or suffered severe development delays because of the need to compress, optimise and modify computer programmes.

The imposition of digital techniques in systems having analogue sensors and displays resulted in a multiplicity of input-output functions and severe complexity in an equipment area which was still fundamentally analogue. However with the appearance of 2nd, 3rd and 4th generation digital systems and the accumulation of experience in their development problems the full benefits of the change of technique are now from time to time realised, particularly if operational requirements are scrutinised and adjusted to match the technology. Certainly one particular problem area has benefited immediately from the proper application of digital techniques: aircraft wiring has been greatly reduced and maintenance and reliability problems have been alleviated.

Computers for Navigation Guidance and System Integration

It is interesting to compare the developments in airborne computing to the evolution of digital techniques in other areas such as business data processing, weapon control systems and air traffic control systems. Here the general purpose computer has become the main tool for computation, data manipulation and decision making and its versatility and flexibility has allowed the development of systems that would not otherwise have been practical. The digital technology has extended to areas outside that of the initial application. The trend is now clearly towards integrated aircraft digital systems which will themselves be integrated, via digital communications systems, with ground based digital air traffic control and airline management systems.

In designing a digital system in an aircraft many choices are open including extreme alternatives and intermediate solutions. These choices include:-

Special Purpose Computers	General Purpose Computers
Distributed Computation	Central Computation
Sub-system isolation	System integration
No reversionary facilities	Full reversionary facilities
Analogue data transmission	Digital data transmission
Individual signal paths	Signals multi-plexed
Variable program storage	Fixed program storage
Short word length	Long word length
Fault detection by hardware	Fault detection by software
Machine code programming	High level language programming

A number of these choices will be treated in depth in later papers. Within the Ferranti Company three computers have been developed for use in airborne systems. Although each of these machines has arisen mainly as the result of historical events and marketing necessity in different Departments of the Company, they do offer complementary characteristics and ranges of potential applications, and serve to illustrate different choices between the parameters listed before.

All of the computers use integrated circuits. The first to be developed was the ARGUS 400, and we believe it to be the first micro-miniature airborne computer to be designed and developed in Europe. It is basically a re-engineered version of the earlier ARGUS general purpose computers, initially developed for the control of the BLOODHOUND missile. It is a 24-bit, "one and a half" address machine using a serial arithmetic unit and a ferrite-core memory operating in the parallel mode. The speed of the store is not fully used by the computer, and the resultant waiting time is utilised by allowing direct access to the store for input and output without loss of computing speed. The computer also has facilities for directly addressing input-output by program. Some applications of this computer will be described later, but it is of interest to note that its main airborne application has been in experimental navigation systems. This confirms our view that a 24-bit data word is sufficient for most navigational systems, and illustrates the flexibility of using a variable program store. In these applications protected volatile program storage appears to have been satisfactory.

A more recent development is the FM 1200 computer. This machine is particularly interesting because its design originated from a proposal to meet the E.S.R.O. requirement for a data handling system for the Large Astronomical Satellite (L.A.S.),(see Ref.1). One of the requirements of the system was that it should have an operational lifetime of about one year. Consideration was given to achieving the required reliability by introducing redundancy at the logic element and also by using error correcting codes. It was eventually decided that the most appropriate arrangement consisted of duplication at the sub-system level, so that in-flight repair could be effected by reconfiguring the system under the control of a ground station. The switching elements used to select sub-systems for service used quadded-logic. The proposed system containing two computers, two input-output sub-systems, two power supply units, eight blocks of core-store and an unduplicated, but internally redundant, switching unit controlled by a redundant tone digital command system. With this arrangement it was predicted that faults could occur which would degrade

the system, but that there was a reasonable probability (0.89) of having a minimal useable system after one year's life.

The computer proposed was a small G.P., single address machine, operating in a serial-parallel mode. It had a word length of 13 bits. A short word length was desirable in order to minimise hardware, and it appeared that at least 12-bits were required to specify an efficient instruction code. The data word length used in associated scientific experiments was to be 13-bits, including parity, and so this word length was adopted for both instructions and data. It is worth commenting here that the adoption of standard word formats for ground-to-air data links might have a similar effect on the choice of word length for future airborne computers. The design was based on low-power T.T.L. logic. The central processor design used some 280 integrated circuit flat-packs, and was estimated to weigh 3 Kgms and have a power consumption of 5 watts. The speed of a basic instruction, using a 1 micro-second beat and an 8 micro-second store cycle time, was about 36 micro-seconds. This was adequate to deal with the estimated computing load of 20,000 simple orders a second.

In the L.A.S. study the achievement of reliability and power consumption targets presented the greatest difficulty; the achievement of acceptable weights, costs and computational speed did not appear to be difficult.

Although the L.A.S. project was not funded, the study of a number of new aerospace and control projects, including jet-engine control and the ARINC 561 Inertial Navigator, indicated that there was a need for a small cheap and reliable computer with similar facilities to those of the L.A.S. machine, but with a higher computing power.

A minimum hardware solution and short word length were still acceptable and were desirable, not only to achieve high reliability, but also low cost.

The word length was reduced to the more conventional length of 12-bits, but the design included features to facilitate multi-length working. The required increased computing power was achieved by adopting fast T.T.L. logic (Ferranti Micronor 5), an existing fast (1 microsecond) core-store, a more powerful function code (including hardware division) and a parallel arithmetic unit. Simple orders are executed in about 4 microseconds and multiplication in 14 microseconds. The central processor unit comprises some 360 dual-in-line elements and occupies 10 cms length of a ½ATR box.

The FM 1200 computer has been used as the basis of a number of studies of aircraft systems, including engine control in multi-engined aircraft and advanced military navigation and attack systems. In general in these systems, where the computing capacity of the machine has been approached, it has been advantageous to use multiple computer systems in order to provide reversion. These studies have also indicated that 24-bit computations are adequate for navigation, and that there is not significant reduction in the proportion of double-length working should the machine word length be increased to 16-bits.

For a number of applications the FM 1200 is now being offered with semi-conductor memories, both for fixed and volatile storage. This gives a common technology with the central processor, and avoids the difficulties of operating a fast ferrite-core store over a wide temperature range. The use of fixed-storage for program allows the G.P. computer to be dedicated to a particular application, and makes it less vulnerable to catastrophic failure as the result of transient noise.

However, available fixed storage elements require the contents of the memory to be determined during the manufacturing process, so that re-programming of a fixed semi-conductor store can be expensive and time consuming.

The third Ferranti Computer is the DISC machine (DIgital Sensor Computer), which belongs to the class of small processors associated with a single sensor and is the basis of a paper by Mr K.R. Brown which will be presented later in the session. The machine was designed initially as a special processor for use in I.N. systems, and with a view to exploiting an available hybrid thin-film technology. For the application a small fixed-program store and a moderate computing speed were acceptable. However precision arithmetic was considered essential. The machine uses an 8-bit instruction word and can perform 16-bit and 32-bit computations. It was originally designed to operate with a fixed sequence of orders, but the order code has since been made more flexible, and in particular now includes jump orders. This has enabled the machine to be more generally applied to sensor and sub-system applications. When used in I.N. systems the computer has a restricted capacity for other tasks.

In addition to the development of these computers Ferranti has acquired experience of the integration of digital computers into systems through two experimental flight programmes. The first of these involved the use of ARGUS 400 as what has since come to be called an area navigation computer. The system was designed to receive inputs from Doppler, heading, inertial, air data, VOR/DME and hyperbolic sensors and to provide present positions, steering and data link outputs. The pilot interface was a small control panel having numerical readouts with switches and keys for mode selection and data input. The equipment was flown on an evaluation basis in a B.O.A.C. VC 10 aircraft in regular service.

The second programme involved flying a version of the Automatic Chart Display originally developed for the Concorde prototype in a Comet operated by the A. & A.E.E., Boscombe Down. The computer in this system was originally intended to interface the A.C.D. with the comprehensive data system in the Comet and its task was conceived as being mainly co-ordinate conversion from latitude and longitude to film co-ordinates. Since the computing task was considered to be too insignificant to justify use of an ARGUS 400, and since the FM 1200 was not developed at that time, it was decided to use a small American computer, the Bosch Arma Micro-D. This machine is particularly compactly constructed and its use enabled all the electronics - computer, I/O and power supply to be packed into a 1ATR unit (7" x 12" x 9" approx.). Eventually, because of problems in interfacing with the aircraft data bus, and because of the desire to demonstrate and evaluate a large number of operational modes, it was decided to interface the computer directly with air data compass, I.N., and Tacan, and to expand the computer programme until it,too, performed many of the functions

of an area navigation system. This system was also flown successfully and demonstrated on a number of occasions in the U.K., Europe, the U.S.A. and Canada.

Input-Output Control

The demands of present systems on the input-output speeds and organisation are not usually critical, but requirements are bound to be increased, for example by the use of electronic displays and on-line signal processing.

In some simple systems direct programme control of input and output is acceptable, the main programme effectively stopping while the full input or output sequence is completed, this sequence being intimately connected with the timing of the processor and its programmes. Alternatively service by the central processor can be requested by a peripheral device, the normal programmes being interrupted to service the programmes required by the peripheral (Programme Interrupt). This system requires a method of allocating priorities if more than one peripheral can bid for the computer at the same time. These approaches were generally adopted in early system applications, in which designers, impressed by the power and flexibility of the new digital computer and by the economics of scale obtainable by maximum use of it tended to put as many functions as possible into the programme, which as a result became deeply involved in the physical functioning of the peripheral sensors. For example, the computer was used to complete the display servo loops in the experimental chart display system previously referred to. While systems designed on this principle can be and have been made to operate successfully, grave disadvantages have been met with. In the typical instance where development of computer software and peripheral hardware proceed in parallel, the problems thrown up in one area will react on the other. The same applies to any changes in the operational requirement. The effect on software development can be particularly bad. As successive modifications become necessary, programmes are altered, rearranged and patched. Under pressure of time, each modification is made not in the most efficient way, but in the way which can be made to work in the shortest time. Soon the available storage becomes full and further modifications are needed to make more space. These may well react in unpredicted ways on other previously satisfactory parts of the programme. Multiple programme interrupts for input/output can be a frequent source of trouble, leading to interactions which are often difficult to recognise and put right.

As a result of these experiences several trends have emerged, all tending to divide the overall system into a number of simple clearly understood sub-systems each having a well-defined interface with the rest of the system. These interfaces are also designed not to require tight synchronisation. Thus there is a tendency to isolate development problems to a particular sub-system. The following are typical of these trends:

Sensors are becoming more self-contained and less dependent on a central processor for control (e.g. the new Ferranti chart display has self-closed loops).

Standardised interface formats are being adopted (e.g. the ARINC 561 and 575 serial formats).

In some cases small digital processors are dedicated to one or a small group of sensors.

A further type of input (Data Interrupt) allows a peripheral to cause the computer to hesitate in its normal sequence while the peripheral has direct access to the computer store. Where more than one block of storage is fitted, it is possible to arrange a system whereby the peripheral can access a store block, locking out the computer from that particular block (Ported Storage).

An example of an advanced input/output system is the Ferranti 'S' Interface, which has been developed for use with the FM 1200. This enables the computer or the peripheral to initiate the transfer of single words or blocks of words from the computer store. The transfer takes place serially between the buffer register in the computer C.I.E. and the peripherals, the buffer register being loaded or unloaded from the store by a parallel-transfer Data Interrupt. At the termination of a sequence of transfers a Programme Interrupt can be initiated for the data to be serviced, but until this happens the transfer proceeds effectively independently of the computer. The rate of serial transfer over the peripheral-computer highway is dictated by the peripheral and can be at a peak rate of up to 8 megabits a second.

Programming and Higher Level Languages

The programming of computers for dedicated airborne applications involving small amounts of storage (e.g. 4000 words) can be adequately performed using efficient assembler languages, although it is often more convenient to simulate the airborne computer on a larger ground-based computer system, which need not have the same machine code. By this means the computer programmes can be developed using a simulated input/output enivronment. For large systems, such as that for a maritime reconnaissance aircraft, the use of higher-level real-time languages can probably be justified.

Future System Configurations

The range of machines and peripherals now available for airborne systems has greatly multiplied the number of options available in determining a system configuration. The following are some of the variables to be considered:-

The complexity of the most significant operational tasks to be undertaken, the processor power required and the number of data sources to which the computer must have access.

The cycle time requirements, related to the bandwidth requirements of the overall loop in which the computer appears.

The operational requirement in the event of failure: failure survival, performance degredation acceptable, reversionary operational mode acceptable, increased crew workload tolerable.

The extent to which the operational tasks can be carried out automatically without human intervention as opposed to those which involve a high degree of discretion of decision making, and hence more powerful displays and means of manual access.

The extent to which library data must be stored, whether it requires to be updated frequently and whether it must be accessed rapidly. The requirement or otherwise for it to be accessed by the crew before being used by the computer.

The extent to which library data must be stored, whether it requires to be updated frequently and whether it must be accessed rapidly. The requirement or otherwise for it to be accessed by the crew before being used by the computer.

Two extreme examples will serve to indicate how these factors can influence the choice of a system configuration.

A commercial aircraft navigating in a complex controlled air space in the 1970s and 1980s will carry an area navigation system which will be duplicated in the interests of safety. The bandwidth required in the outer navigation loop is such that a comparatively low computation speed is acceptable provided that any comparatively simple guidance computations imposed on the computer can be carried out rapidly enough. For the crew the operational problem will be to comply rapidly with air traffic control instructions and clearances which can involve choices between a large number of available options over a complete route network. For example up to 20 standard arrival or departure procedures could be associated with each major terminal airport. The workload associated with terminal area navigation is at its highest immediately after takeoff and during the descent and final approach when other sources of workload in the cockpit also reach a peak. Such a system will therefore impose heavy demands in two particular areas: the storage of large amounts of data in a cheap back-up storage medium and the provision of displays permitting the crew to access and inspect the data before using it for flight control purposes. It is probable that this requirement for the crew to monitor the computer's operation fully will persist when the A.T.C. system itself is largely automated and clearances are transmitted to the aircraft through data link.

Such a system is therefore likely to consist of two moderately powerful general purpose computers with large capacity back up stores and electronic displays. For pilot orientation purposes separate displays capable of conveying position and orientation data unambigiously are also a requirement and it is probable that the projected moving map using microfilm storage will become a standard flight instrument for this purpose, since the data stored in it is independent of that stored in the computer system and can be used to cross check it.

In contrast there will be a continuous requirement for high performance military aircraft having low cost easily maintained systems capable of continued operation following unserviceability or battle damage. These aircraft will carry comparatively complex sensors such as inertial platforms, air data sensors and multi-mode radars, each of which has a processing requirement. These requirements will probably be best met by small rugged fixed programmes dispersed computers such as DISC or its successors. Whether or not a more complex general processor or processors are carried will depend largely on the operational role, the weapons used and the extent to which the crew require assistance in their deployment and operation.

Finally a large complex military aircraft employed on maritime reconnaissance or as an airborne command post may require central computers much more powerful than any at present in service.

Reference 1: J.E. Remmington, R.A. Williams, R.E. Wright "The outline of the Ferranti data handling system proposed by the Sub-Aviation Group for project L.A.S.". Proceedings of the 1968 International Colloquium on Aerospace Computers in Rockets and Spacecraft, published by the Centre National D'Etudes Spatiales, Paris.

SYSTEMS TASKS FOR THE DATA PROCESSING EQUIPMENT OF ADVANCED AIRCRAFT NAVIGATION SYSTEMS

by

F.G. Unger and R.S. Sindlinger

TELDIX G.M.B.H.

Heidelberg, Germany

SUMMARY

The tasks of the data processing equipment in advanced navigation systems for aircraft are vastly increased as compared to conventional systems in use today. Major tasks are processing of different sensor information, solution of navigation equations, optimal use of redundant data by filtering and estimation, automatic mode control, malfunction detection and isolation. These tasks are described to some detail and possible hardware solutions are discussed.

1. INTRODUCTION

Most military aircraft operational today carry only one means of navigation. This may be an inertial navigator, a Doppler radar, or radio position fixing devices. Sometimes two different and independent systems may be available to the pilot. The basic configuration of such systems is described in Section 2. The main task of the computer is to process the available sensor data through a set of navigation equations to come up with the required navigation information.

The next generation of navigation systems for military aircraft will undoubtedly be much more complex in order to meet the stringent requirements in performance, reliability and availability. These systems, as described in Section 3, will include an inertial platform, a Doppler radar, means for position fixing, and back-up devices such as air data unit, magnetic compass etc. These sensors provide redundant data to the computer, and the optimal use of all the available information is a much more complex task than the mere solution of a set of navigation equations in conventional systems. In addition, the different sensors allow a series of emergency modes which must be performed and controlled by the navigation computer. Additional tasks such as automatic upmoding and downmoding, fault detection and fault isolation, display and output computations etc. are required from the data processing equipment. Section 4 presents the details of these different tasks.

A summary of the multitude of tasks of the data processing equipment for advanced navigation systems is given in Section 5. Possible hardware solutions are outlined. Digital computers available today or under development are able to meet the requirements, but an optimal configuration is not easy to find. Although the navigation system is only a subsystem of the whole avionic system of the aircraft, the multitude of tasks for the data processing equipment of this subsystem alone makes the decision of federated vs. integrated design a vital question.

2. COMPUTER TASK IN CONVENTIONAL NAVIGATION SYSTEMS

For comparison, the task of the computer in conventional navigation systems will now be described. A pure inertial system is used as an example, but the basic configuration, as shown in Fig. 1, is the same for other systems, too, only the navigation equations are different.

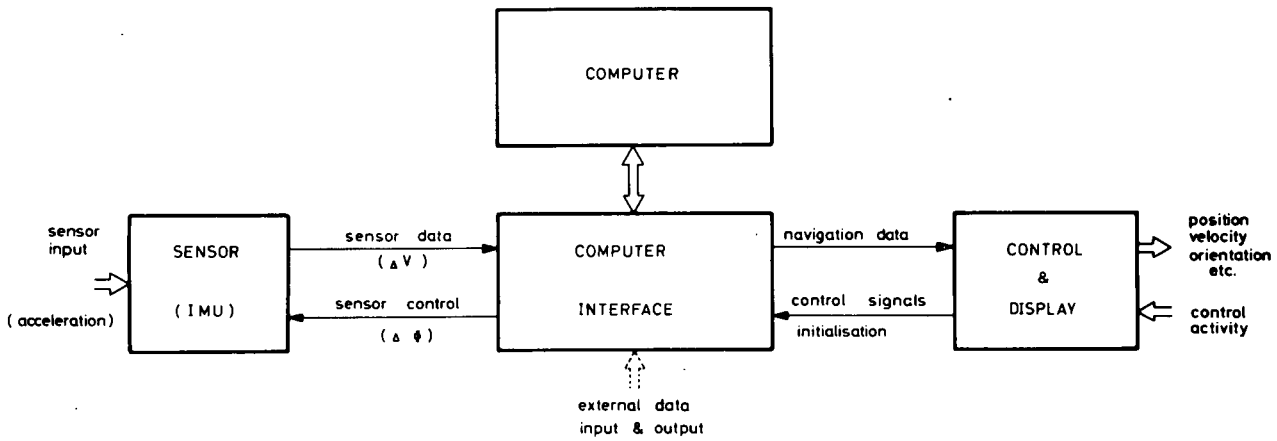


Fig. 1: BLOCK DIAGRAM OF A CONVENTIONAL NAVIGATION SYSTEM

The system consists of the sensor unit, in this case an IMU, the computer including the necessary interface, the display and control unit, and the power supply. In case of a Doppler system, the sensor unit includes the Doppler sensor and a heading reference.

The function of such a system is very straightforward. The system is turned on through the control unit. After proper warm-up, the platform is aligned to its desired orientation. This mode is controlled by the computer through a set of alignment equations. Inputs to the computer are accelerometer signals, and outputs are the torquing rates for the platform. After proper alignment the mode is switched to the main or navigation mode by the operator.

The computer task during the navigation mode can be described according to the block diagram in Fig. 2. The outputs of the IMU are measured accelerations in form of velocity increments. These velocity increments must be multiplied by proper scale factors and compensated for known biases, gravity, and coriolis accelerations etc. to determine the velocity increments in a navigation coordinate system.

These increments are then integrated for velocity and a second integrator provides position information. Additional compensation may correct for the ellipticity of the earth, if necessary. Torquing rates for the platform including compensation signals for fixed gyro drift and earth rotation are calculated from position and velocity information.

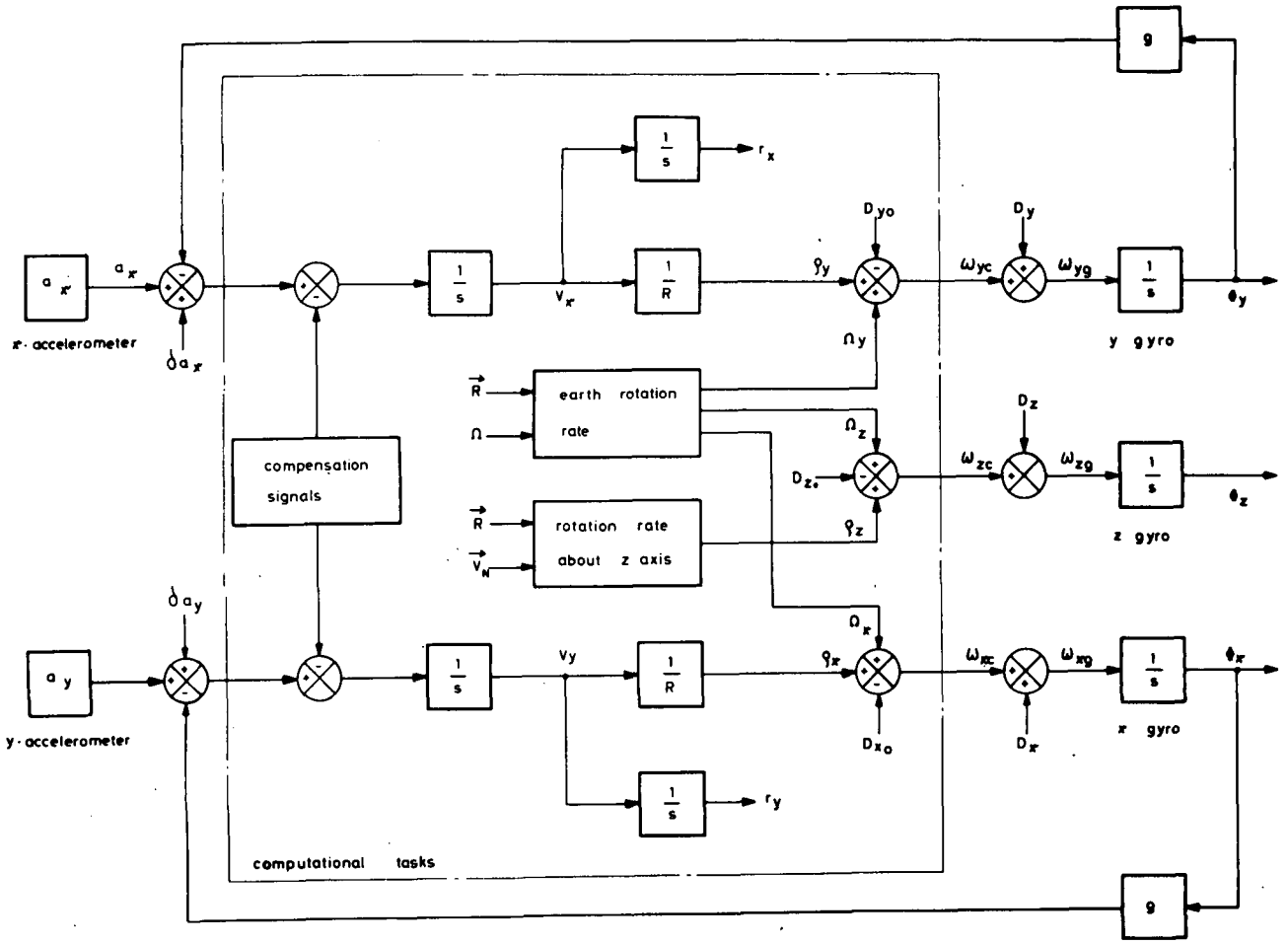


Fig. 2: SCHULER-TUNED INERTIAL NAVIGATION SYSTEM

As indicated in Fig. 2, inputs to the computer during the main or navigation mode are sensor data in form of velocity increments, and the outputs are position and velocity information to the control and display panel, and rate commands to the platform. The computer tasks as described above can be expressed in a set of navigation equations which must be processed at a relatively high rate (5 to 30 times per second). These equations can be expressed by linear differential or algebraic equations and are suitable for analog and digital computing techniques.

The computer hardware available for these tasks today can be put into four categories:

- Analog Computers
- Digital Differential Analyzers (DDA)
- Special Purpose Whole Number Computers
- General Purpose Computers

Analog computers gradually disappear and do not need to be considered further. All three digital computer types have their special advantages and disadvantages and the best solution depends on the requirements of a particular application.

The DDA is best suited for solving the navigation equations especially for a pure inertial system, because it computes incremental changes to quantities rather than whole values of quantities themselves. New LSI circuits are especially suited for DDA's and allow new designs. The biggest disadvantage of a DDA is its poor flexibility, especially if additional tasks are required.

Special purpose computers are of increasing importance through the use of read-only-memories. They present a good solution from a cost-effectiveness point of view in cases where the computational

problem is clearly defined at the beginning.

General purpose computers were reduced in size, cost, weight and volume during the past few years through the use of microelectronics, and so-called single task GP computers are coming into use. Their prime advantage is their flexibility when the computational requirements are changed.

3. CONFIGURATION OF ADVANCED NAVIGATION SYSTEMS

Next generation navigation systems for aircraft will certainly combine a multitude of sensors to meet increased requirements in accuracy, reliability and availability. A typical configuration of such systems is shown in the block diagram of Fig. 3. The central unit of the system is the data processor or navigation computer whose inputs are supplied by a series of sensors shown on the left hand side of the block diagram. The navigation information is made available through output and display units as indicated on the right hand side of the block diagram. Only the flow of navigation information is shown in Fig. 3, while control signals are not detailed for reasons of clarity. The processing of control signals such as self-test information is also part of the computer task.

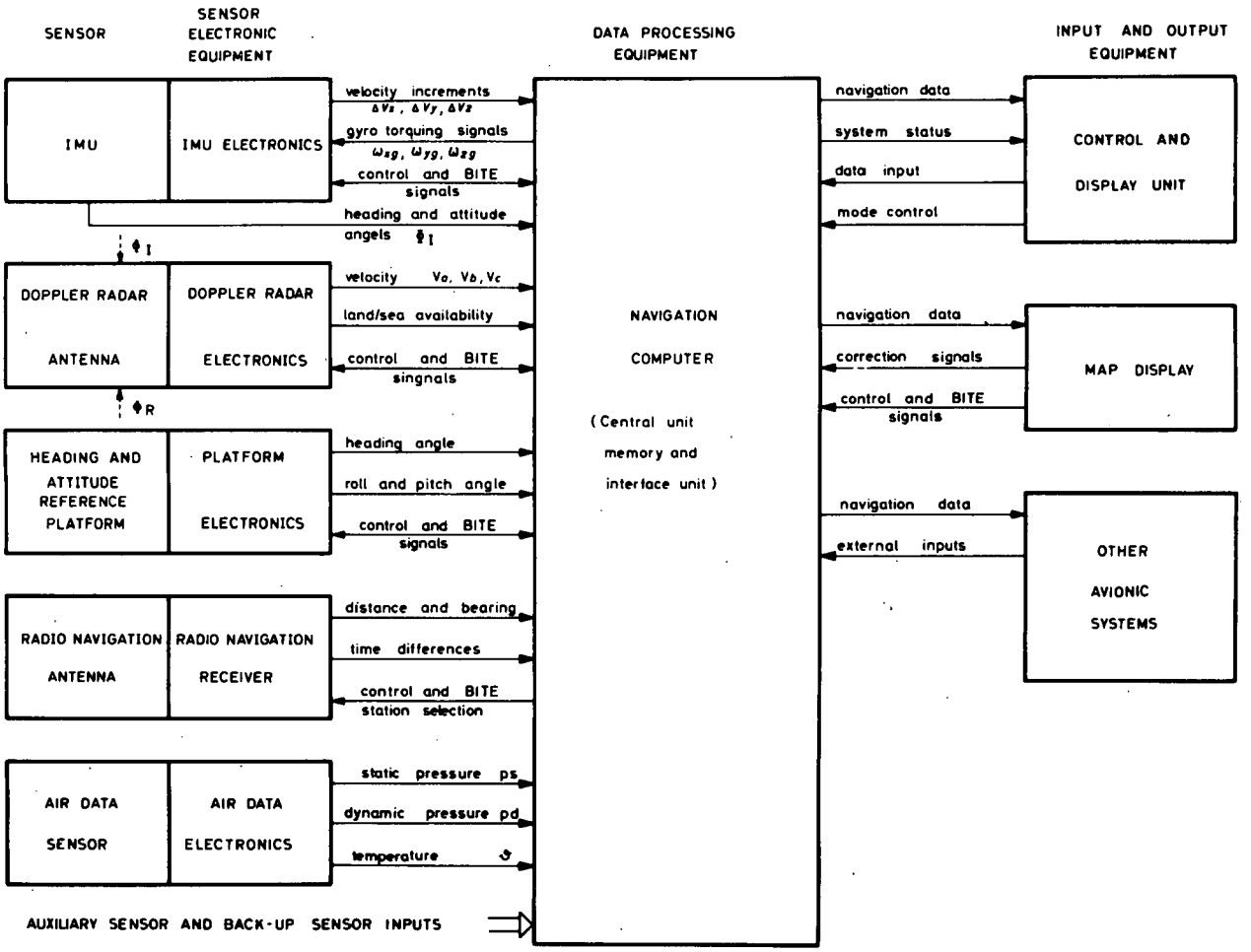


Fig. 3: ADVANCED NAVIGATION SYSTEM (MULTISENSOR SYSTEM)

Any particular system may not necessarily contain all of the units shown in Fig. 3, but this will not change the computer task significantly, as long as redundant sensor information is available and must be processed in an optimal sense.

The different sensors, their measured quantities and their possible outputs and inputs are now described.

- Inertial Measurement Unit (IMU)

  The inertial platform is stabilized in a known coordinate frame and measures acceleration along platform axes. Partial integration is accomplished in the platform electronics.

  Available outputs: velocity increments ΔV in platform coordinates x, y, z

  orientation angles Φ_I (usually analog)

Required inputs: gyro torquing rates ω_g for platform orientation (usually in digital form as angular increments $^g\Delta\Phi_I$)

Control signals to the computer: ready signals - depending on temperature, gyro wheel supply, self-test outputs etc. - and IMU mode indication signals.

Control signals from the computer: turn-on and turn-off signals for mode control

- Back-up Heading and Attitude Reference Unit

This unit provides orientation angles in case of a malfunction of the inertial platform.

- Doppler Radar:

The Doppler radar sensor consists of a body-fixed or a stabilized antenna and the required electronics. Orientation angles for stabilization can be provided by the inertial platform or a vertical gyro. The Doppler radar measures ground speed of the aircraft.

Available outputs: distance increments or velocity components in body-fixed axes
or: velocity and drift angle;
land-sea criterion

Required inputs: orientation angles Φ in case of a stabilized antenna.

Control signals: availability and self-test signals, mode control signals.

- Radio Receiver Unit:

Radio position-fixing sensors under consideration are Loran C and D, Decca, Tacan and Omega. These sensors give position information with respect to known positions of the transmitter stations.

Available outputs: distance differences to three stations in form of time difference pulses (in case of Loran, Decca and Omega)
or: range and bearing to one or more stations (Tacan)

Required inputs: station selection signals, phase rate commands (Loran).

Control signals: availability and self-test signals, station identification signals; mode control signals.

- Air Data Sensor:

The air data sensor measures static and dynamic air pressure and air temperature. From these data can be derived vehicle true airspeed and barometric altitude.

Available outputs: static and dynamic air pressure, air temperature (these outputs are usually analog)

Required inputs: none

Control signals: availability and self-test signals.

- Additional back-up sensors for emergency modes:

Magnetic compass, vertical and directional gyro etc.

The different output and display units and their corresponding input and output signals are as follows:

- Control and Display Unit:

This unit is the interface between the pilot and the navigation system. It provides display of navigation information as well as system status, such as mode of operation, availability of subsystems, malfunction occurrence etc.

Inputs from the computer: position, velocity, heading, attitude; time of flight, bearing and distance to targets.

Outputs to the computer: initial position, flight plan data.

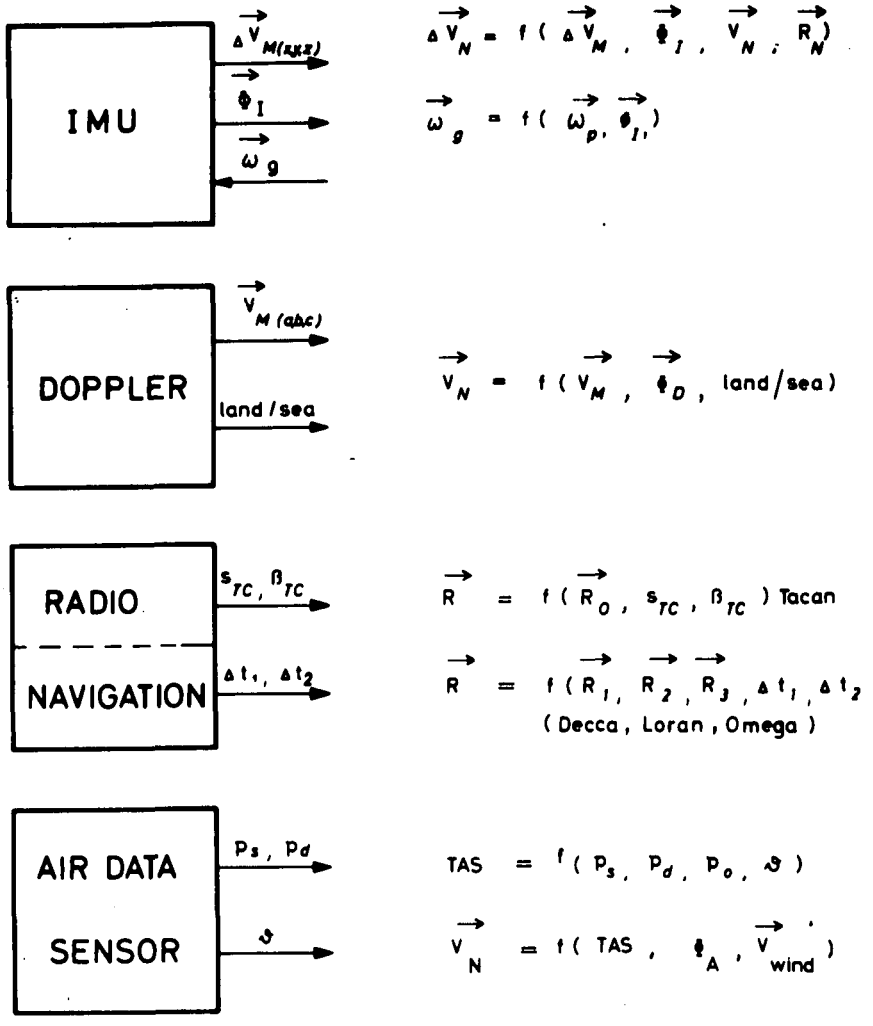


Fig. 4: PROCESSING OF SENSOR DATA

The necessary gyro torquing rates $\vec{\omega}_g$ must be derived from the desired platform rotation rate $\vec{\omega}_p$ and the platform orientation $\vec{\Phi}_I$.

The platform gimbal angles are not directly required as navigation information, but they are used to determine Doppler and air data sensor orientation.

Doppler Sensor:

The Doppler sensor measures ground velocity $\vec{V}_{M(a,b,c)}$ along aircraft or Doppler beam axes a, b, c. The measured data must be transformed into a velocity vector $\vec{V}_N$ along navigational coordinates by using Doppler orientation angles $\vec{\Phi}_D$. Changes of the Doppler scale factor must be included by using land/sea criteria.

Radio Navigation Sensor:

The required transformations depend on the type of system used. In case of a Tacan receiver, position information $\vec{R}$ must be computed from the measured distances s_{TC} and the bearing angle β_{TC} to a Tacan station and the known position $\vec{R}_o$ of this station. Simultaneous or sequential measurements to two or more stations may considerably reduce the position error because the error contribution of the bearing angle uncertainty is relatively high, if only one station is used.

In case of hyperbolic systems (Loran, Decca, Omega) the position $\vec{R}$ of the aircraft is comput-

ed from the known positions of three stations $\vec{R}_1$, $\vec{R}_2$, $\vec{R}_3$ and two measured time differences Δt_1 and Δt_2 which are proportional to the difference in distance to these three stations.

Air Data Sensor:

True air speed TAS can be computed from measured static pressure p_s, dynamic pressure p_d, air temperature ϑ, and known normal pressure at sea level p_o. This function is highly non-linear and is often empirically determined. The velocity in navigational coordinates $\vec{V}_N$ is the vector sum of true airspeed, transformed by using air data sensor orientation $\vec{\Phi}_A$ and wind velocity $\vec{V}_{wind}$.

The required word length for these calculations is between 12 and 24 bits. The iteration rate is usually 1 to 100 times per second.

4.2. Solution of Navigation Equations

The required navigation equations are depending on the mode of operation. During the main mode the inertial platform is used as the prime source for navigation information while the data of the other sensors are incorporated through the measurement matrix of the Kalman filter.

The tasks of solving the navigation equations as outlined in Fig. 5 are:

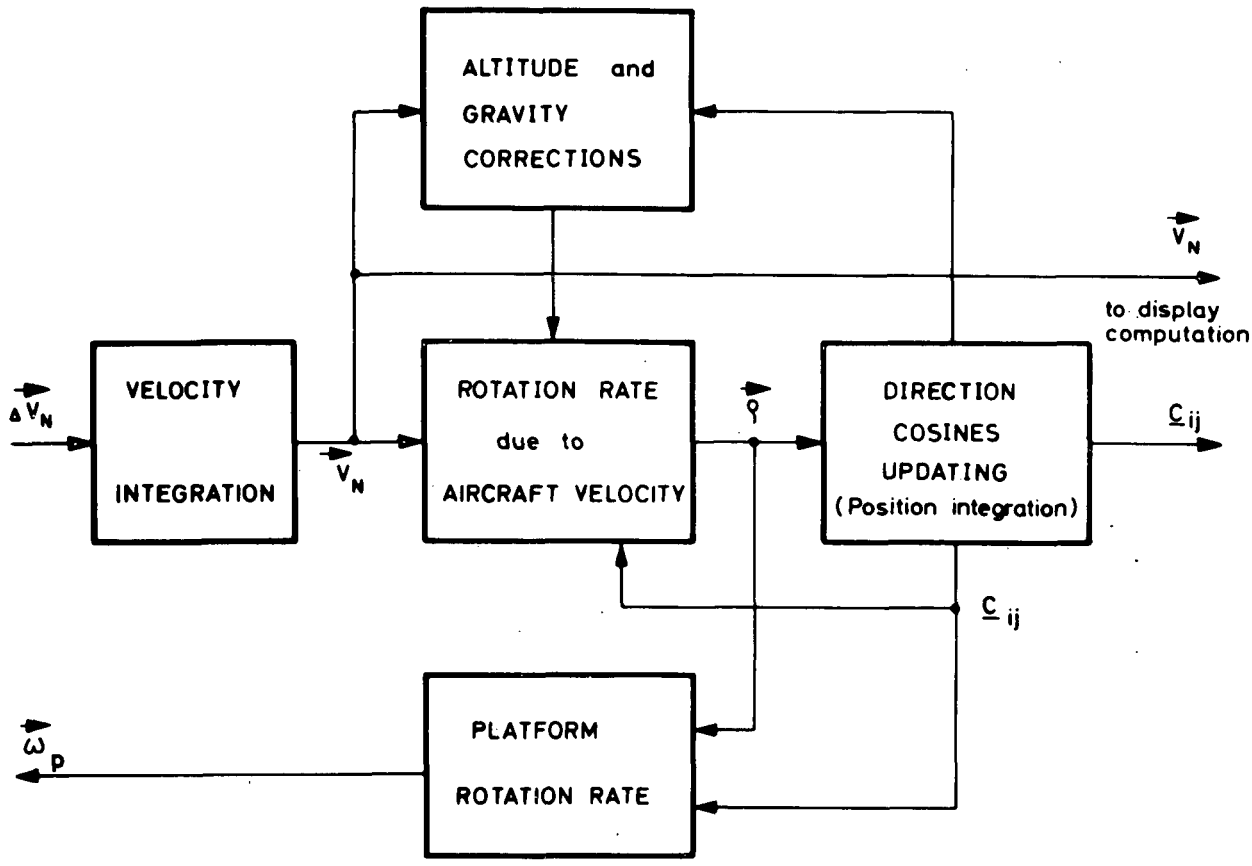


Fig. 5: SOLUTION OF NAVIGATION EQUATIONS

- Integration of velocity increments $\vec{\Delta V}_N$ for aircraft velocity $\vec{V}_N$.
- Computation of the rotation rate $\vec{\varrho}$ of the navigational frame with respect to the earth using position, velocity and altitude.
- Position integration, usually performed by updating a direction cosine matrix $\underline{C}_{ij}$. The use of a direction cosine matrix eliminates the problems otherwise encountered with polar flights.
- Computation of altitude and gravity corrections of the vertical channel. The vertical channel is not always used in the inertial platform. In this case, altitude is derived from outputs of the air data sensor alone.

- Computation of the necessary platform rotation rate $\vec{\omega}_p$ to keep the platform locally levelled. The azimuth axis of the platform can be slaved to true north, not rotated at all (free azimuth) or rotated at a constant rate (up to 1 rpm). This platform rotation reduces the effects of bias type errors of the horizontal gyros and accelerometers.

Velocity integration and computation of the platform rotation rate incorporate the required computations for Schuler tuning of the platform.

The required word length for these computations is between 16 and 24 bits depending on system accuracy. The necessary solution rates vary from 5 to 30 times per second.

4.3. Filtering and Estimation

All available sensor signals are deteriorated by random errors whose values as a function of time are not predictable. However, if the statistical properties of these random errors are known or can be approximated by mathematical models, it is possible to obtain navigation information with minimum errors by application of optimal filtering and estimation theory. The mathematical algorithm for this task is the minimum variance or Kalman filter. This filter performs the following two tasks:

- Determination of the desired navigation information from all available input data with minimum error variance in real time.

- Control or updating of the system error states.

The Kalman filter consists of a set of iterative filtering equations which are based on the linearized differential error equations of the system.

The set of equations (1) through (6) is the basic filter in state-space notation.

$$\vec{X}_k^* = \underline{\Phi}_{k,k-1}\,\hat{\vec{X}}_{k-1} + \underline{\Gamma}_{k,k-1}\,\vec{U}_{k-1} \qquad (n \times 1) \qquad (1)$$

$$\vec{Y}_k^* = \underline{M}_k\,\vec{X}_k^* \qquad (m \times 1) \qquad (2)$$

$$\hat{\vec{X}}_k = \vec{X}_k^* + \underline{B}_k\,(\vec{Y}_k - \vec{Y}_k^*) \qquad (n \times 1) \qquad (3)$$

$$\underline{B}_k = \underline{P}_k^*\,\underline{M}_k\,(\underline{M}_k\,\underline{P}_k^*\,\underline{M}_k^T + \underline{V}_k)^{-1} \qquad (n \times m) \qquad (4)$$

$$\underline{P}_k = (\underline{I} - \underline{B}_k\,\underline{M}_k)\,\underline{P}_k^* \qquad (n \times n) \qquad (5)$$

$$\underline{P}_{k+1}^* = \underline{\Phi}_{k+1,k}\,\underline{P}_k\,\underline{\Phi}_{k+1,k}^T + \underline{H}_k \qquad (n \times n) \qquad (6)$$

The size of the vectors and matrices is indicated by n and m, where n is the number of variables in the state vector $\vec{X}_k$ and m is the number of measurements made at time t_k.

The following notation is used in the above equations:

$\vec{X}_k^*$ = a-priori-estimate of the system state vector predicted at time t_k

$\hat{\vec{X}}_k$ = a-posteriori-estimate of the system state vector at time t_k after using measurements

$\underline{\Phi}_{k,k-1}$ = state transition matrix relating the state vectors $\vec{X}_k$ and $\vec{X}_{k-1}$

$\underline{\Gamma}_{k,k-1}$ = distribution matrix relating state vector $\vec{X}_k$ and control vector $\vec{U}_{k-1}$.

$\vec{Y}_k^*$ = observation vector of measurements predicted at time t_k.

$\underline{M}_k$ = measurement matrix relating measurement vector $\vec{Y}_k$ to state vector $\vec{X}_k$

$\underline{B}_k$ = weighting (gain) matrix at time t_k

$\underline{P}_k^*$ = covariance matrix of the a-priori estimation error $\vec{X}_k - \vec{X}_k^*$

$\underline{P}_k$ = covariance matrix of the a-posteriori estimation error $\vec{X}_k - \hat{\vec{X}}_k$

$\underline{V}_k$ = covariance matrix of the measurement white noise vector

$\underline{H}_k$ = covariance matrix of the system white noise vector.

The Kalman filter equations are processed on an iterative basis to determine the optimum estimate $\hat{\vec{X}}_k$ of the system error state at sampling time t_k.

The different computations occurring during the time period from t_{k-1} to t_k are outlined in the schematic diagram of Fig. 6. The required computations are separated into four blocks while the real system is represented by dotted lines in the middle part of the left hand side of the block diagram. This part is included to demonstrate the flow of signals between the filter in the computer and the system.

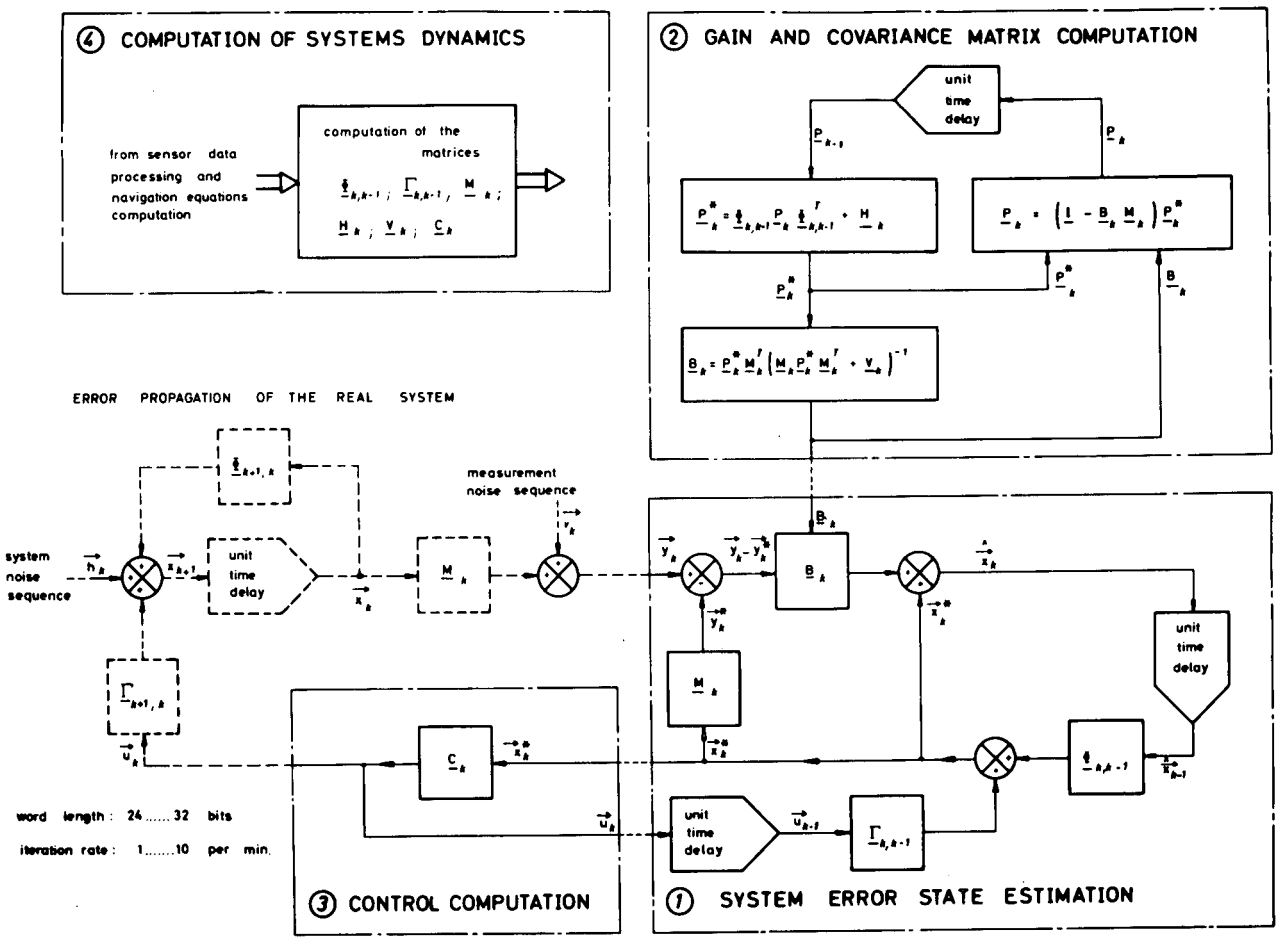


Fig. 6: BLOCK DIAGRAM OF A LINEAR OPTIMAL ESTIMATION AND CONTROL FILTER

The four different computational tasks as indicated in Fig. 6 are:

(1) System state estimation:
This task is mathematically described by equations (1), (2) and (3). The estimated measurement errors $\vec{Y}_k^*$, corresponding to the a priori estimate of the state vector $\vec{X}_k^*$, are compared with the real measurement errors $\vec{Y}_k$. The difference, multiplied by the weighting matrix $\underline{B}_k$, is then used to determine the best estimate $\hat{\vec{X}}_k$ of the error state vector.

(2) Computation of the covariance matrices and the weighting matrix $\underline{B}$:

The weighting matrix $\underline{B}$, required for the estimation of the state vector, is calculated by using equations (4), (5) and (6). These equations also include the calculation of the covariance matrices $\underline{P}_k$ and $\underline{P}_k^*$ of the error state vector.

(3) Control computations:

The Kalman filter equations are based on the linearized error equations of the system. This

linear error model is only valid as long as the components of the error state vector are small. This requirement can only be achieved by controlling the state vector in a closed loop operation. This means that computed values of the state vector, after proper transformations into the control vector $\underline{U}_k$, are fed back to the system and used to correct system error parameters (i.e. gyro drift, biases, scale factors etc.).

(4) Computation of systems dynamics:

The matrices representing the systems dynamics are required for the filter equations and must be consecutively computed.

The state transition matrix $\underline{\Phi}_{k,k-1}$ describes the error propagation in the system from time t_{k-1} to time t_k. It is derived from the coefficient matrix A(t) given by the system error equations and valid error models.

The measurement matrix $\underline{M}_k$ is determined by the systems configuration.

The distribution matrix $\underline{\Gamma}_{k,\,k-1}$ is determined by system control equations.

The covariance matrices of system noise $\underline{H}_k$ and measurement noise $\underline{V}_k$ are also determined from noise error models and systems dynamics.

The processing of the Kalman filter equation represents a very heavy burden on the navigation computer, because many matrix transformations and multiplications with large matrices are involved. It is therefore very important to keep the order of the error state vector, and accordingly the order of the matrices, as low as possible and still make maximal use of the redundant data. Very extensive studies and simulations are required to optimize the Kalman filter for a particular system with regard to size of the state vector, required word length, and iteration rate. Usually, adequate word length lies between 24 and 32 bits and the required iteration rate for the complete filter is between 1 and 10 times per minute. But a small part of the computations must be performed at higher rates.

4.4. Submodes and Emergency Modes

The navigation computer must be programmed to perform certain tasks associated with the different possible modes of operation. The submodes are used during normal operation while emergency modes are only used in case of equipment malfunction. During these modes, it is possible to use parts of the navigation program or the optimal filtering program in combination with additional equations. Logic decisions are required to combine these program parts.

The different modes that must be considered are:

Stand-by:	This mode is used to warm up the inertial platform and to keep it on operational temperature. No computer functions are performed and usually power is switched off from the computer.
Alignment:	This mode includes coarse and fine alignment of the inertial platform. A set of alignment equations must be processed to achieve gyrocompassing. The platform may be slewed to true north, not rotated at all, or rotated at a constant rate. The Kalman filter can be incorporated into this mode.
Navigation Emergency Modes:	In case of a malfunction of the inertial platform, navigation is performed by dead reckoning using velocity and heading information. Several different back-up modes are available using Doppler radar or air data as velocity information and the back-up heading and attitude reference, a directional gyro, or the magnetic compass as heading reference. The navigation equations that must be available in the computer are almost independent of the type of sensor used. A simplified Kalman filter can also be implemented if redundant position information is still available.
Calibration:	Calibration of certain system parameters is performed during the normal navigation mode with the Kalman filter. Additional calibration can be done during alignment, also with the aid of the Kalman filter. It may be necessary, however, to add a special calibration mode, where calibration is performed to a higher degree of accuracy under favorable environmental conditions.

Test: It may be necessary to use a special test mode for system check-out in addition to the self-test equipment which is used in all modes of operation.

4. 5. Calculation of Output and Display Information

A series of information is required for display to the pilot or as data for other avionic subsystems. Part of this data is computed by solving the navigation equations, but additional equations must be solved by the data processor on an iterative basis in order to supply the following information:

- Position and velocity data in special coordinate frames, e.g. in grid coordinates
- Aircraft heading, track and drift angle
- Distance and time of flight to several way points or targets
- Wind speed and direction
- Special data for other avionic subsystems (e.g. fire control and autopilot)
- Flight plan restrictions derived from fuel consumption and other limitations

Display information is usually accurate enough by using a 12 to 16 bit word length, information for other subsystems may require a higher accuracy.

4. 6. BITE Data Processing

The specification for advanced aircraft navigation systems include the requirement that the system has its own malfunction detection and malfunction isolation system. It must be determined by the system design to what level malfunctions must be isolated and to what degree of reliability malfunctions must be detected. It can be stated that the necessary hardware increases proportional to the requirements put on the built-in test equipment (BITE). This additional hardware adds to weight, volume and cost of the system.

The BITE task in a navigation system is outlined in Fig. 7. Each sensor or unit must have its own BITE electronics to detect and isolate malfunctions of this particular unit. The computer can be tested by a special self-test routine. The BITE signals from all the units of the system are processed in a system self-test and malfunction isolation loop. Outputs of this loop are malfunction signals for display and downgrading commands to the system mode control loop.

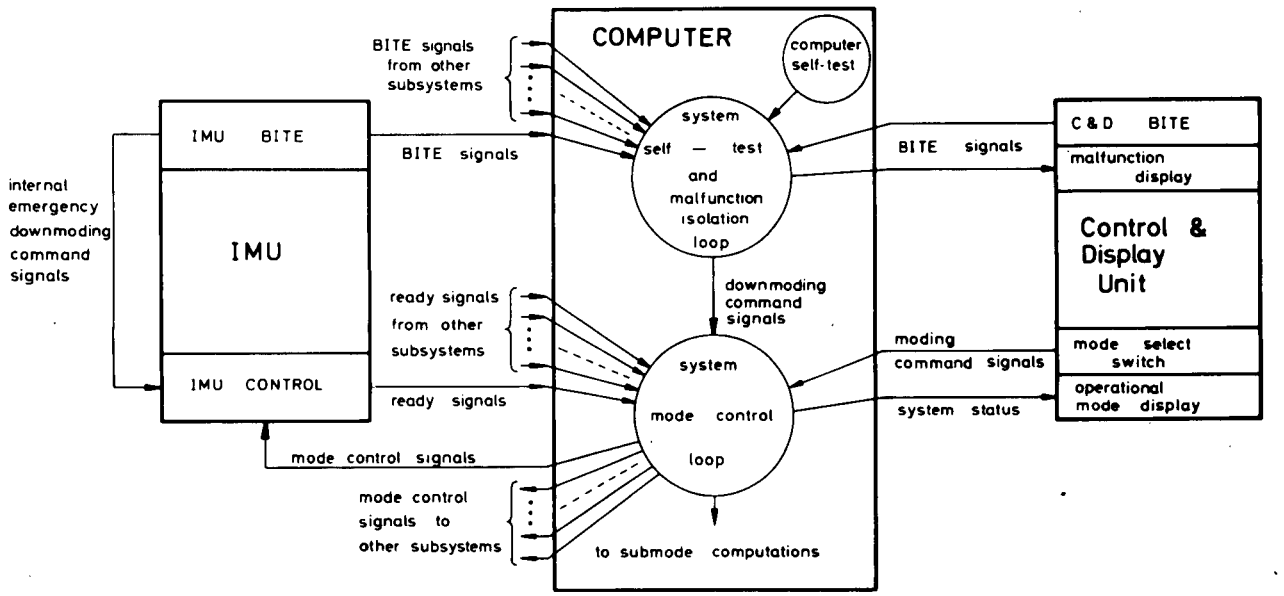


Fig. 7: BITE AND CONTROL DATA PROCESSING

The system self-test and malfunction isolation loop can be a special subroutine in the computer or can be realized by special logic hardware.

4. 7. Mode Control of the Navigation System

The desired mode of operation is selected by the pilot with the mode select switch on the control

panel. In advanced systems, it is requested that upmoding and malfunction-caused downmoding to the best available emergency mode is controlled by the data processor. This task is also outlined in Fig. 7.

Mode command signals from the control panel indicate the desired mode of operation. The computer determines in a mode control loop the required control signals and transmits these signals to the different units. Ready signals are supplied from each unit to indicate the status of this unit. Signals of this type may be "wheel power on", "heater power on", "antenna active" etc.

Signals from BITE may require automatic downmoding of the system. The resulting actions of the computer control loop are then to provide control signals to some units, signals to indicate the system status on the display panel, and the necessary changes in the navigation computation loop.

It is possible that internal emergency downmoding or turn-off signals are provided in some units, as indicated in Fig. 7 for the IMU. These signals are necessary to prevent secondary damage in case of a malfunction.

5. SUMMARY OF TASKS AND HARDWARE CONFIGURATIONS

The many different tasks for the data processing equipment, as described in the previous section, are summarized in Fig. 8. The center block covers all the different computational loops that must be performed for sensor data processing, navigation, optimal filtering and output data computations. The required iteration rates and word length for these loops are vastly different. In addition, some of the loops may have common branches as indicated for the navigation computations and the emergency navigation modes. The data processor as the central part of the navigation system must have a high reliability in order to meet required system reliability.

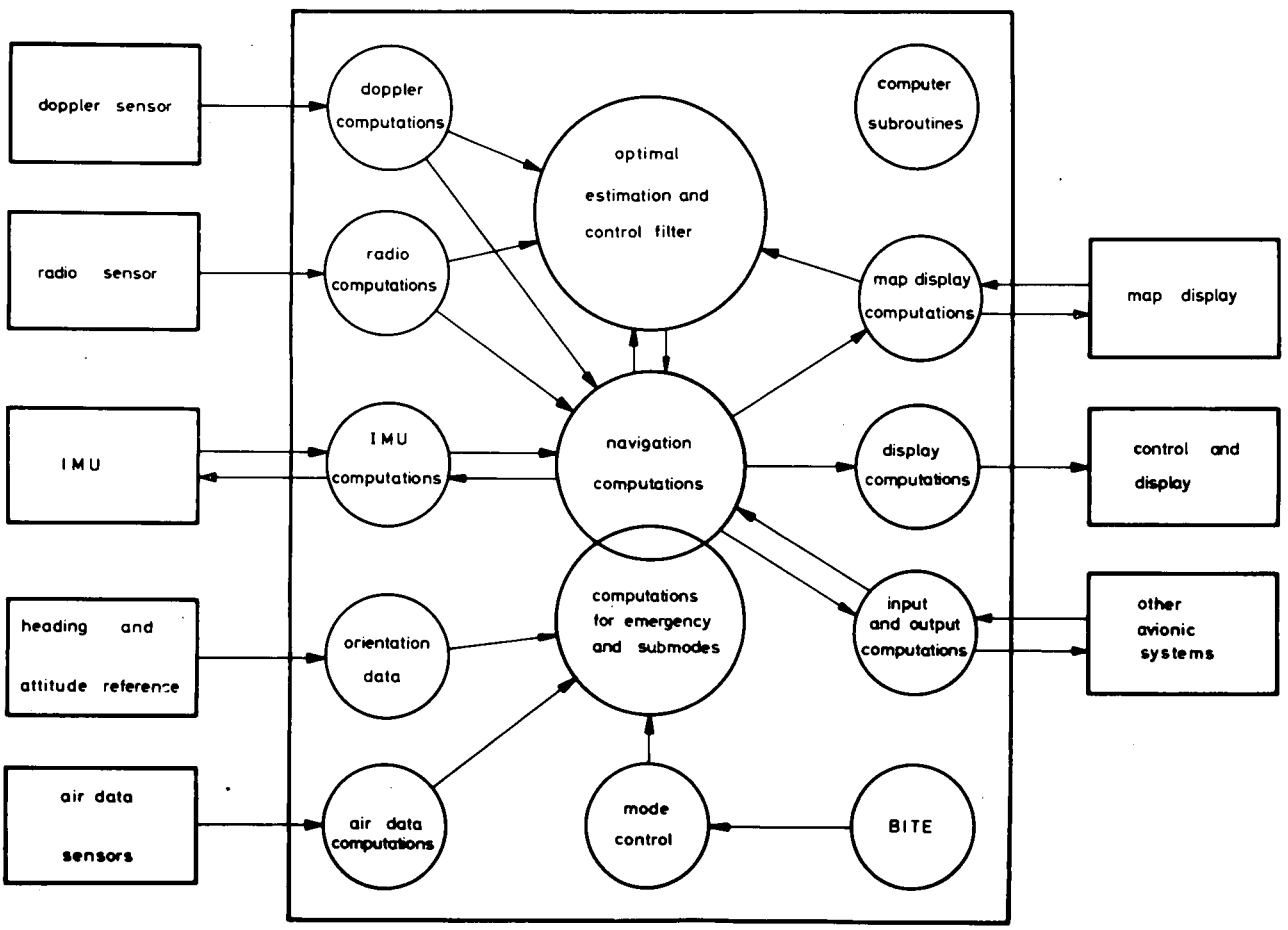


Fig. 8: COMPUTATIONAL LOOPS IN AN ADVANCED AIRCRAFT NAVIGATION SYSTEM

There are two basic hardware configurations useful for the data processing equipment:

- One central GP computer for all the tasks.
- Several small computers for some of the sensor data and output data processing and a medium size

computer for navigation, optimal filtering and mode control.

The use of a single GP computer results in the minimum hardware, and the advanced computers available today are able to meet all the requirements. But the programming of all the different loops with iteration rates ranging from 10 to 30 times per second to 1 every few minutes is a very tedious task. The accuracy requirements will probably best be met by a 16 to 24 bit word length using double precision for the filtering and part of the navigation computations. The difficult task of scaling all parameters may be alleviated in the future by using floating point machines. In order to meet the required reliability it may be necessary to use a second computer as back-up or to have double or triple redundancy inside the computer.

The use of several small computers has the advantages of simpler programming and higher reliability. These computers may be DDA's or small GP machines. The following tasks are suitable for separate processing:

- IMU sensor processing.

  Often a small inertial platform computer is used for position integration including Schuler tuning of the platform. This computer must be available if "hot insertion" is required.

- Air data computations.

  A special air data computer is advantageous because of the very special computational requirement.

- Radio navigation data processing.

  Solution of hyperbolic position information into earth-fixed coordinates can be performed by a small computer as part of the radio navigation unit.

- Display computations.

  Some of the display computations may be performed in the map display or the control and display panel.

- BITE data processing.

  The processing of all BITE information may be performed by a separate malfunction detecting and recording computer for the navigation system or for the complete avionic system.

These peripheral computers are usually matched with a central computer for optimal filtering, navigation and mode control. But emergency modes must be available, in case the central computer fails.

The broad spectrum of tasks of the data processing equipment for advanced aircraft navigation systems has been outlined. The basic specification that must be used by the system designer to detail the tasks for a particular system is that maximum use is made of all available sensor information and that the obtainable systems performance is not significantly degraded by the processing of data.

FEDERATED VS. INTEGRATED COMPUTER SYSTEMS

by

James H. Crenshaw

IBM Corporation,
Federal Systems Division,
Electronics Systems Center,
Owego, New York 13827, USA.

SUMMARY

As it becomes possible through microminiaturization techniques to provide more digital computing capability in smaller, lighter packages, the Aerospace industry is finding it desirable to utilize the advantages of digital computers in more complex and varied applications.

The problem of computer subsystem organization must be solved in such a way as to best accommodate the system requirements and also provide flexibility, expandability and reliability. Solutions of this problem can vary between the two extremes represented by (1) the completely dedicated loosely-federated, system made up of as many CPU's as there are tasks and (2) the completely integrated, simplex or multi-processor, system capable of performing all of the required tasks.

This paper reviews the system organizations utilized in such operational systems as the F111-MKII, the A-7D/E, IHASS and ILASS. It also reviews the design considerations for various other developmental systems.

The advantages and disadvantages of the Federated and Integrated approaches are reviewed from an abstract point of view and then the system dependent variables which must be considered by the designer are discussed to emphasize the real world situation. Such features as vulnerability to the "single bullet" and graceful degradation are considered along with the queuing theory, "waiting time", and loading factor trade offs. The important factor of relative cost of the two approaches is also discussed.

In conclusion, the final selection may be determined by system dependent factors although customer preference is a major consideration. In general the comparisons presented in this paper favor the integrated system with problem size determining whether a single processor or a multiple processor configuration is required.

FEDERATED VS. INTEGRATED COMPUTER SYSTEMS

James H. Crenshaw
IBM Electronics Systems Center
Owego, New York 13827

INTRODUCTION

Each advance in computer technology generates more complex and varied applications for digital computers in the aerospace industry. These advances include faster circuit and memory speed, microminiaturization, and dramatic cost reductions. The selection of computer organization is a dynamic process, continuously taking advantage of the latest technology to meet an expanding number of applications. Historically, computer organization has varied between the two extremes represented by:

- The completely dedicated, loosely-federated, system made up of as many computers as there are tasks. (This approach predates the advent of general purpose digital computation for aerospace applications and has been retained in modified form for some digital applications.)
- The completely integrated system capable of performing all the required computations within a single computer complex. (Capabilities of a single processor may be expanded by employement of multiprocessors if requirements exceed those which can be met with a simplex computer.)

In practice, computer organization may be a combination of both, as will be illustrated by examples such as the Gemini, IHAS, ILAAS, Saturn, Mark II, A-7D/E, and VS A-NEW. The advantages and disadvantages of the two organizational concepts are examined considering such factors as: size, weight, cost, storage requirements, vulnerability to battle damage, queuing problems, waiting time, and loading factor. From examination of past system organizations, technological development, and the comparison of federated vs. integrated systems, conclusions are drawn relative to future trends for aerospace applications.

This paper is primarily limited to general-purpose digital computers. Definitions of terms used in the paper as they apply to general-purpose computers are as follows:

Federated Computer System

A federated system consists of several computers, each dedicated to a particular task. The computers communicate through their I/O channels for normal functions. Redundancy may be provided, as in the case of the dual federated system, where each computer can perform backup functions in addition to its normal functions in the event of malfunction of the other computer.

Integrated Computer System

An integrated system is defined as one which performs unrelated tasks in a multiprogrammed mode of operation. The system may contain one CPU (the simplex system), or two CPU's sharing common main storage operating in a multiprocessing mode. The number of job queues distinguishes a federated system from an integrated system. A federated system requires a job queue and executive program in each computer, while an integrated system has only one job queue and executive program residing in the common storage.

Multiprogrammed System

Independent of the hardware configuration, a system may or may not be multiprogrammed. Multiprogramming is the concurrent, interleaved execution of multiple programs which are implemented by timed or externally controlled interrupts.

Applications

Table I is a partial listing of aerospace applications for general-purpose digital computers.

Many factors, some technical and others arbitrary, affect the final partitioning or assignment of functions. At the present time, special-purpose processors are generally employed for high-speed functions such as radar data processing, character generation, image enhancement, and terrain following. Other functions such as landing control, air data, and engine control are generally separated due to tradition, reliability considerations, and the desire for independent operation.

Analog and special-purpose digital processors will continue to be applied to such functions, but the rationale for each function should be re-examined for each new system in the light of advancing technology. However, this paper will be limited to the organizational forms of the general-purpose digital computer which encompasses the major portion of the computational functions.

Table I

AVIONIC SYSTEM FUNCTIONAL APPLICATIONS

Navigation
Inertial Platform Management
Guidance
Weapon Delivery
Operator Control and Display
Character Generation
Sensor Control
Sensor Data Storage
Statistical Processing
Filtering
Image Enhancement
Air Data
Terrain Following
Flight Parameter Recording
Countermeasures Detection and Control
Threat Analysis
Auto Pilot
Landing Control
Engine Control
Self Test
System Diagnostics (AIDS)

SURVEY OF EXISTING SYSTEMS

A survey of existing systems reveals that there is no single satisfactory approach. Both integrated and federated systems have been developed and successfully utilized. Each application is somewhat unique, but some insight into trends can be obtained and conclusions drawn by examination of the information generally available in the literature regarding the existing systems selected for review. All of the following computer systems described are products of IBM Owego, with the exception of IHAS and ILAAS, which were included because of their unique structure.

Gemini

The Gemini system utilizes a simplex-multiprogrammed digital computer which performs rendezvous and reentry functions and also backup ascent guidance. See Figure 1. This computer is classed as an integrated system for the functions which it performs. However, there are many other elements of the system which do not interface with the computer. Figure 2 is a functional diagram of the Gemini System.

Integrated Helicopter Avionics System (IHAS)

The IHAS computer central complex consists of a Signal Transfer Unit (STU), a Digital Interface Unit (DIU), and a Central Processor Unit (CPU). See Figure 3. Triple redundancy is employed to enhance reliability; modularity permits growth for adding functions or sensors. The computer central complex is an integrated system since it handles both the navigation and sensing functions. However, the individual elements are a hybrid consisting of a general-purpose STU and special-purpose (federated) Digital Differential Analyzer (DDA).

Integrated Light Attack Avionics System (ILAAS)

ILAAS employs a federated computer system containing two independent, dedicated-multiprogrammed digital computers: one performs navigation and weapon delivery, and the other performs inertial and radar data computations. See Figure 4. Two additional redundant computers are optional if additional reliability is desired. A core of functionally independent units is employed to reduce the impact resulting from the loss of a function. Each core unit is independent and does not depend on other elements. Full system operation is achieved with the core units and other integrated units. Graceful degradation is achieved in the event of equipment failure, since the core-functions alone provide the final backup capability.

Saturn

The Saturn Guidance and Control System, shown in Figure 5, employs a federated system in which each component performs its own particular function. The digital computer is concerned only with the guidance task. Within this task, the computer is integrated with a triple modular redundant CPU and duplex memory to achieve the reliability required for the Apollo mission. Figure 6 is a photograph of the Saturn Computer used in the Apollo mission.

Mark II

The Mark II digital computer is shown in Figure 7. The Mark II System uses two identical digital computers and a three-section signal converter. Each computer is dedicated to a specific function, i.e., weapon-delivery or navigation. However, each computer contains backup routines for the other with independent sections of the signal converter to provide graceful degradation. In addition, a third smaller computer is completely dedicated to inertial navigation computations. Figure 8 is a functional diagram of the computer complex.

The total system may be considered to be federated; however, the individual elements function in many ways similar to an integrated system. In fact, the backup weapon-delivery functions are normally computed simultaneously in the navigation computer because of the transient which would otherwise occur during a transfer of control.

A-7D/E

The A-7D/E Avionics Computer, Figure 9, is a multiprogrammed-simplex-integrated system which handles functions such as navigation, weapon delivery, and self-test in interleaved manner.

The emphasis in the computer is on simplicity and minimum computational hardware necessary to perform a relatively large number of system functions. The A-7D/E began as a replacement for an existing analog computer in the A-7A/B. During the proposal stage, as the flexibility, accuracy, and added modes which are possible with digital computation became apparent, numerous functions were added resulting in a true integrated system. Extensive signal conversion equipment is supplied and is characteristic of an integrated computer system. However, no redundancy is provided, either in hardware or in communication channels. See Figure 10.

The A-7 system provides, at a fraction of the cost, functional capabilities comparable to larger, more complex systems. However, there is no graceful degradation in the event of a computer failure.

VS A-NEW

There is a growing trend in systems currently under development to utilize multiprocessor techniques. The tasks to be performed are so complex that a single computer cannot accomplish the job. VS A-NEW is an example of this trend. Figure 11 is a simplified diagram of the system.

The VS A-NEW system is a multiprogrammed-multiprocessor-integrated system which has considerably greater capabilities than most previous avionics computers. In this system, each CPU has access to all storage and I/O units and the operation is controlled by a single executive routine and job queue. In the supervisor program, the system uses concepts and techniques similar to the dual IBM System 360 Model 65 shared-storage multiprocessing system.

The VS A-NEW, A-7D/E, and Mark II are all members of the IBM System/4 Pi family.

In all of the previous systems, even where the central computer is integrated, there are other dedicated processors assigned to specific functions which provide inputs to the central computer or utilize outputs. For example, the pilot's displays may contain various forms of general-purpose processors for generating symbols and buffering information received from the computer. Another processor may exist in the radar to operate on raw radar data, to provide filtering or weighting, and to convert information to digital format for transmittal to the central computer. Air data computations and auto-pilot functions are almost always performed independently. Therefore, in the strictest sense, all the systems selected are to a greater or lesser degree federated. These systems contain a general-purpose computer, integrated in the context of the functions which it performs, but which is federated with other special-purpose or dedicated processors.

FEDERATED vs. INTEGRATED COMPUTER SYSTEMS

The consideration of federated vs. integrated computer systems is most interesting when there is a requirement for multiple processors, whatever the reason, i.e., computational complexity exceeds the capabilities of a simplex computer, or other considerations such as redundancy, graceful degradation, or independence for flight safety. Obviously, an integrated system may use a single simplex computer.

Integrated systems are put to best advantage in situations where task queues tend to be long or unpredictable, and where processing power and flexibility are at a premium. Graceful degradation can be provided with multiple processors with either system. Federated systems may be less vulnerable to battle damage if the processors can operate independently.

Table II summarizes some of the relative advantages and disadvantages of the two configurations.

Multiprocessing in an integrated system, provides an advantage over a federated processor system, since a federated system would require multiple executive programs and program queues. An integrated multiprocessing system, on the other hand, has only one program queue and one executive. Both CPU's are controlled by the one executive, and both receive program priorities from the same queue. The benefit of this approach is examined in the next paragraph.

Table II

ADVANTAGES AND DISADVANTAGES OF COMPUTER SYSTEM ORGANIZATIONS

Integrated System	Federated System
Advantages	
• reduced program wait times	• more system design flexibility
• more powerful degraded modes	• reconfiguration automatic and simple
• more efficient load sharing	• low system vulnerability (with backup implemented)
Disadvantages	
• reconfiguration complex	• extensive intersystem communication
• main storage interference	• high storage requirements
• more complex executive software	

More than two CPU's operating in a multiprocessor mode appear unwieldy. More than two processors frequently occur in a federated system.

WAITING TIME

A simplified analysis of average wiating time spent in the queue shows that if all tasks occurred randomly, took the same amount of running time, and the computer loading was the same on both systems, the average waiting time would be as shown in Figure 13. It can be concluded that the average waiting times of programs being executed by a multiprocessor will be significantly less than the average waiting times of programs executed by federated machines. One of the basic assumptions for the queuing analysis, described above, is that all tasks occur randomly. Many tasks encountered in an aerospace computer application occur in a very ordered pattern; e.g., navigation every 100 milliseconds, etc. In this instance, the queuing theory analysis is not applicable and the waiting time for both systems can be made nearly equal. The federated system which depends on the ordered arrival of tasks for efficient operation, is less flexible. If a task is changed, added, or deleted, the entire pattern of task assignment may have to be reordered. An integrated multiprocessing system compensates automatically for shifts in program loading.

PROGRAM STORAGE

Storage interference occurs only in an integrated multiprocessor system, since more than one CPU has access to a given portion of memory. Also, additional hardware is required to provide multiple access. Interference will add wait time to the program execution time. If a large number of storage blocks are used, the probability of storage interference is low. However, if a few blocks of shared storage are used, the probability of interference increases.

A federated processor system is certain to require more main storage than an equivalent multiprocessing system, due to the need for: duplicate subroutines, status checks, self-test, and executive programs. The multiprocessing system with common storage minimizes redundancy.

COMMUNICATIONS

In any system containing two or more processors, communications must be considered. Each processor tests to see if the other is working correctly, and signals the other to shut down if a failure has been detected. It is desirable to minimize the time devoted to internal communications since this is non-productive. The integrated multiprocessing system has an advantage in communication. The common storage affords an excellent means for data transfers, requiring the direct channel only to signal the availability of data in storage. Thus, the integrated system requires less time for communications. If current information is required for backup mode regression, the penalty of additional time for I/O is encountered.

DEGRADED MODES

A problem concerning system loading, which impacts only the federated processor type system, involves the method selected for degraded mode operation. Some federated systems, for example the Mark II, are designed so that each processor does simplified backup computations for the other. In this case, the total system processing load is greater, by the precent of total time spent on the redundant tasks, than the integrated multiprocessing system.

Vulnerability to battle damage and a single part failure is a major reason for selecting dual processors. The non-redundant, federated system is more vulnerable to failures, since all processors must be operational to retain full capability. Partial function may be retained, however. If storage redundancy is employed, and the additional penalty for transferring current information between processors is accepted, an effective backup mode capability can be implemented. Loss of I/O is more serious with a federated system, and may be equivalent to losing the CPU, since the same channel is used to communicate between processors as well as between the processor and the peripheral devices. However, the multiprocessing system can be designed to survive with only limited performance degradation after loss of an I/O channel. CPU-CPU communications do not rely on I/O; therefore, the working I/O channel, if connected to all external systems, may be capable of handling the entire communication load alone.

The integrated multiprocessor system has a natural regression mode in the event of CPU failure; but the possibility that a CPU failure could destroy information in common storage is a real hazard. The dual federated system, offering physical and electrical separations, is better able to survive such battle damage. Separating components in an integrated multiprocessor system is difficult since storage and CPU's must be installed relatively close together.

The mission is an important factor when considering redundancy. Some aerospace missions involve vehicles which will be sent singly on long missions; for example, the Apollo mission. In this case, a great deal of redundancy is justified. However, in the tactical aircraft mission, two or more aircraft are generally sent together. Under these circumstances, not all systems are vital. In the final analysis, a pilot could complete a mission virtually without any of his avionics by flying formation with his wing man.

If a storage module fails, the information stored in it is lost for as long as it takes to reload or replace the module. Some systems utilize redundant program storage on a drum or tape to reload a lost program. The impact of a storage module loss depends on the system size. Where the module is a small percentage of total storage available, the loss may not be serious. Conversely, if the module is a large percentage, the impact will be severe.

In large processor systems, multiprogramming is generally used. Programs tend to become fragmented and scattered through a number of storage modules. Consequently, loss of a single module may destroy components from a number of different tasks. To minimize this effect, the system should be partitioned such that each task and all of its associated data and files can be located in a well-ordered and well-identified segment of the storage unit. Roll-in/roll-out procedures can be used to aid in maintaining clean partitions. Coalesce procedures can be employed to close up any open spaces and minimize fragmentation.

COST

Cost is an important consideration and depends heavily on the other topics already discussed. A considerable amount of redundant storage is necessary in a dual federated processor system as well as redundant CPU and I/O, resulting in higher system cost.

There is an offsetting cost impact of a multiprocessing storage system since multiple accesses to the storage module must be provided.

An integrated multiprocessing system, in order to take fullest advantage of all its capabilities, requires two I/O channels, either of which can give access to all external devices. This provides increased reliability but at the cost of additional hardware.

TECHNOLOGY TRENDS

In any new application, the latest technology should be examined. This may have a dramatic impact on the organization of the system. The following curves are representative of the advances which are taking place in digital technology.

Figure 14 is a logarithmic plot of the "add time" in microseconds of fifty aerospace computers which have been developed since 1957. Note the exponential decrease reflecting improvement in the state-of-the-art.

Figure 15 shows the improvement in "multiply time" of the aerospace computers introduced since 1957. Multiply times are characteristically an order of magnitude slower than add times but show the same relative improvement.

CONCLUSIONS

System selection is strongly influenced by part experience and customer preferences. Problems in digital computers of the past include; loss of information from drums, low computer reliability, hang-up in program loops, errors in reading analog-to-digital converter, noise, EMI, and programming errors. Past experience can lead to selection of system approaches not based on evaluation of current state-of-the-art. Technological gains in component size reduction, cost reduction, increased performance, and improved reliability make system organizational concepts previously restricted to ground installations, feasible for aerospace applications. As technology advances, applications will grow and the demand for complex systems will increase.

Table III summarizes the characteristics of state-of-the-art systems from the preceding examples. It is the writer's opinion, based on this comparison, that the system design should begin with a single computer simplex integrated system. Multiprocessors may be required if the computational requirements exceed the capabilities of a simplex system, or for graceful degradation in the event of malfunction; but it is important to separate capacity considerations from reliability. The first approach to improving operational effectiveness should be to emphasize component reliability and simplicity of design. In most airborne systems today, digital computers are more reliable than the peripheral sensors. The dramatic, two orders of magnitude, improvement in reliability over the past ten years, can be predicted to continue in the future. It is feasible to expect that digital computer mean-time-between-failures may soon be equal to the expected system life. Most of the more elaborate organizations which have been discussed in this paper can be categorized as a form of redundancy. Over the years, IBM has successfully employed duplex computers, quad-circuit redundancy, and triple modular redundancy to effect improved reliability. However, redundancy in any form should be undertaken only as a last resort, since all parameters except mission effectiveness are adversely affected. Cost, weight, volume, power, and the total number of maintenance actions are increased. As has been pointed out in the paper, the mission application also affects the selection of the computer organization. In many applications, the simplex computer may be the place to stop. Redundancy should be considered only for those areas of the system demanding such attention, rather than indiscriminately duplicating equipment.

Table III

SUMMARY OF CHARACTERISTICS

Configuration	Size	Weight	MTBF	Storage	Cost
Simplex	1	45	3000	16K	1.0
Integrated Multiprocessor	1.7	75	1400	21K	1.5
Federated	2.1	95	1000	24K	2.0
Dual Federated	2.4	100	4000*	32K	2.4

*based on backup modes

Programming simplicity is becoming increasingly important. Except in those applications where extremely large production quantities warrant customized design, it may be justifiable to accept inefficiencies. For example, standardization of software programs may become desirable. Once the program is developed and checked out, it could be employed in future programs desiring this capability even though some minor penalties might be incurred. The present duplication of effort and occurrence of costly errors could be avoided. In this respect, the federated system offers some advantage. There is a prevalent opinion that the programming, particularly of independent or nearly independent functions, is easier to manage with a federated system. I do not agree that this is necessarily true. The software interface can be defined and controlled with strict adherence to application ground rules with the same thoroughness which is required for the hardware interface between two computers.

The conclusion of federated vs. integrated for future applications is by no means obvious. A case can be made for either approach and, in fact, almost every system which was examined during the preparation of the paper proved to be neither entirely integrated nor federated, but rather a combination of both. This can be expected to continue. However, considering the growth in applications and the continuing reduction in cost and physical size, it is predicted that the integrated, multiprogrammed, multiprocessor will gain increasing acceptance in future applications, following the pattern of ground-based computational centers. The simplex integrated system will remain the most efficient organization and will find continuing usage in many applications. The federated system is expected to decline due to its more limited flexibility and equipment penalties.

ACKNOWLEDGEMENTS

The author acknowledges inputs from a number of internal study reports with particular insight gained from those by Mr. C.H. Simmons, Mr. P.H. Giroux, Mr. H.G. Jud, Mr. C.H. Powers and Mr. A.S. Buchman. Mr. S.B. Balden contributed to the assembling of data, analyzing trends, and organizing the paper.

BIBLIOGRAPHY

Baechler, Donald O. STATE OF THE ART OF AEROSPACE DIGITAL COMPUTERS, 1962-1967. Computer Group News. January 1968.

Daggert, D.H.; Lee, R.Q. The F-111D COMPUTER COMPLEX. American Institute of Aeronautics and Astronautics. 1968.

Harold, Glen M. TRENDS IN AVIONICS SYSTEMS. Control Data Corporation.

Miller, Barry. ADVANCED F-111 SYSTEM NEARS PRODUCTION. Aviation Week & Space Technology. September 9, 1968.

Miller, Barry. IHAS PROTOTYPE FLIGHT TESTS UNDER WAY. Aviation Week & Space Technology. December 4, 1967.

Price, Richard C. INTEGRATION PLUS FEDERATION. Sperry Rand.

Seidel, Paul E., and others. ILAAS -- INTEGRATED LIGHT ATTACK AVIONIC SYSTEM. Sperry Rand Engineering Review. 1968.

Hylton, Harvey I. THE F-111 MARK II COMPUTER COMPLEX BASELINE: HOW IT WAS DERIVED. NAECON Proceedings. 1967.

REFERENCES

1. Gemini Electronics Systems, SPACE/AERONAUTICS, October, 1963, pg. 83.

2. IHAS Simplified Block Diagram, AVIATION WEEK, June 21, 1965, pg. 50.

3. ILAAS Block Diagram, NAECON PROCEEDINGS, R.C. Price, Sperry Rand Corp., 1969, pg. 6.

4. Saturn Guidance & Control System, ASTRONAUTICS & AERONAUTICS, (a publication of the American Institute of Aeronautics & Astronautics, Inc.), E.D. Geissler and W. Haeussermann, NASA, Huntsville, Ala., February 1962, pg. 45.

5. The MARK II Digital Computer Complex, NAECON PROCEEDINGS, H.I. Hylton, Wright-Patterson Air Force Base, Ohio, 1967, pg. 224.

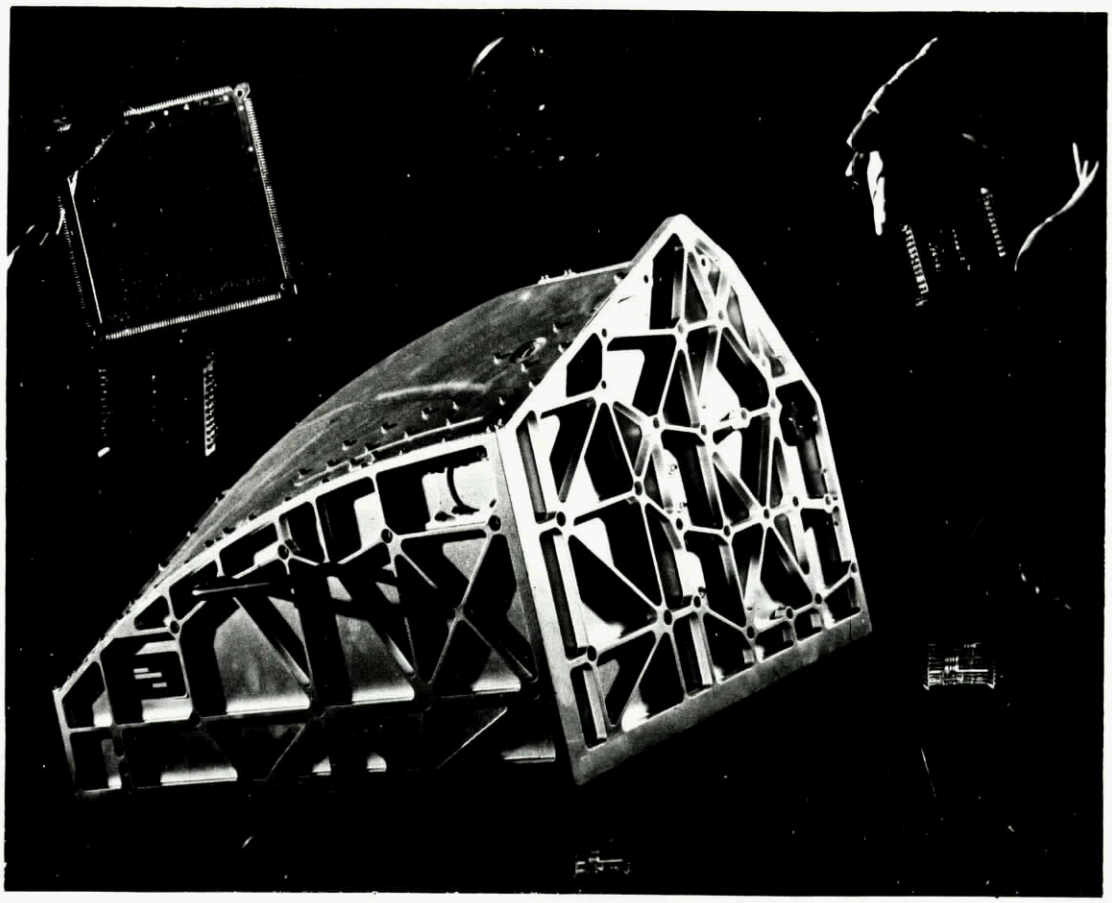


Figure 1. Gemini Computer

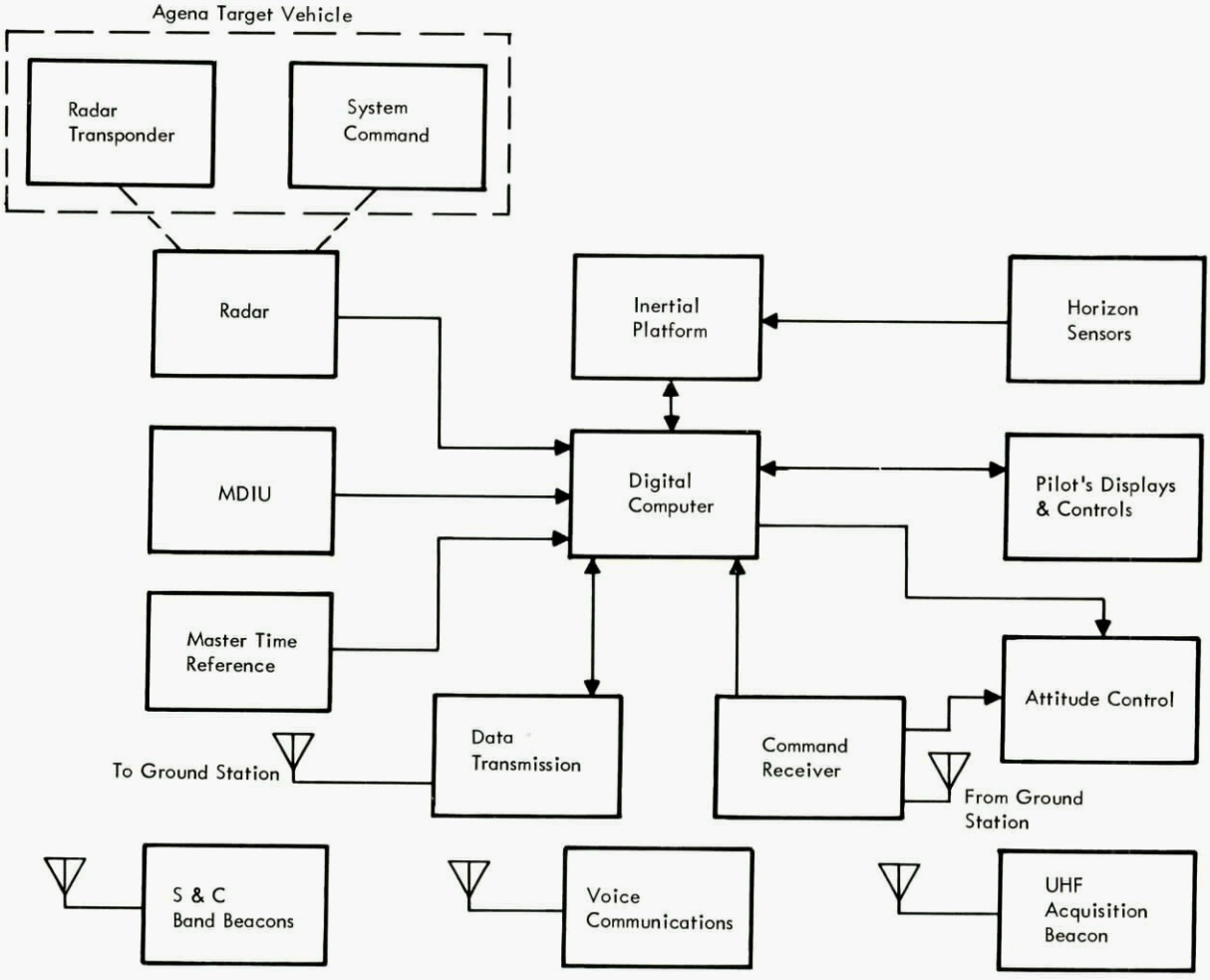


Figure 2. GEMINI Electronics System

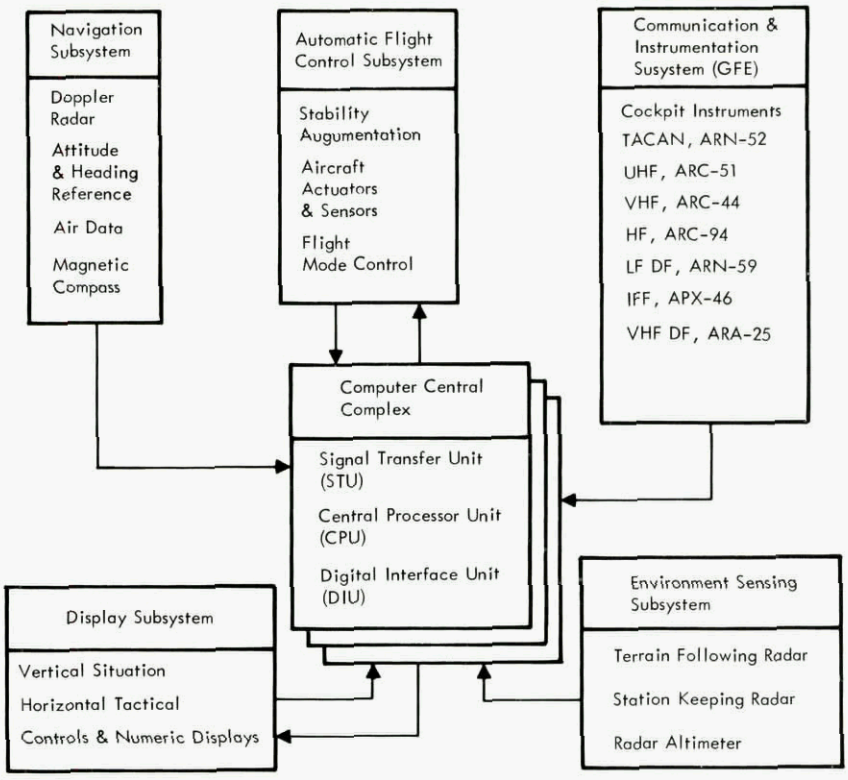


Figure 3. Simplified Block Diagram of IHAS System

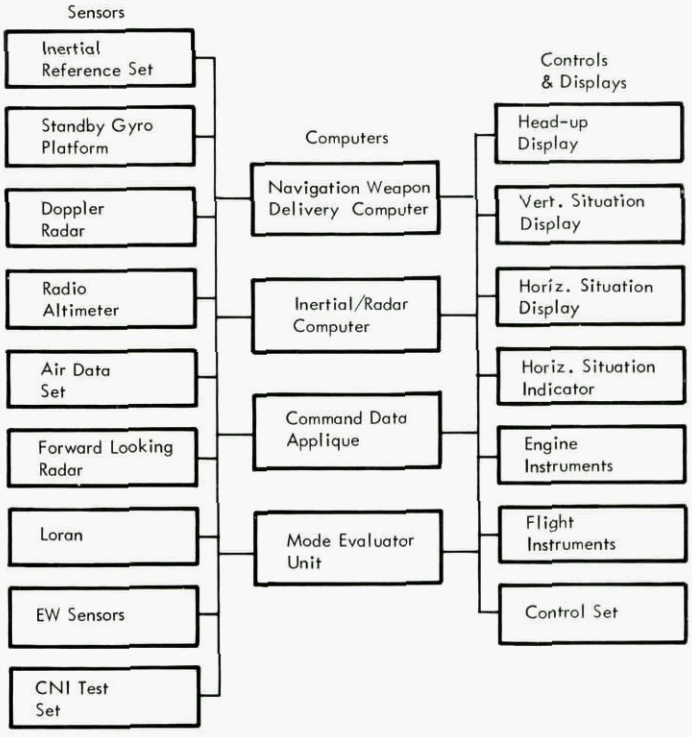


Figure 4. ILAAS Block Diagram

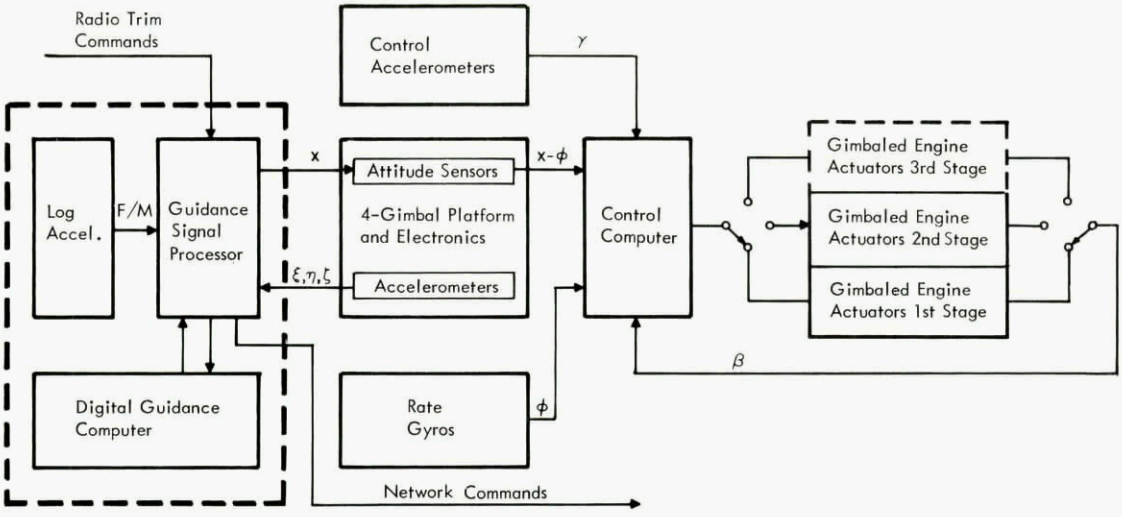


Figure 5. Saturn Guidance and Control System

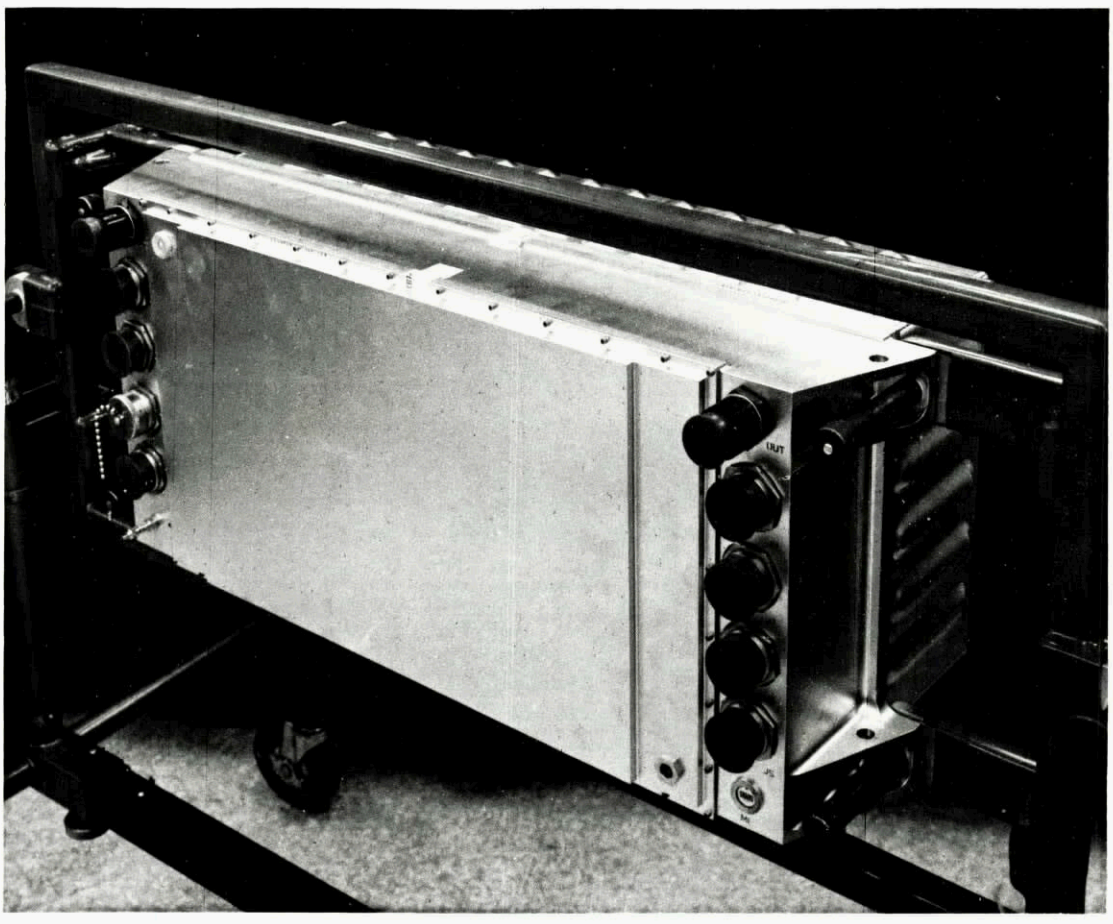


Figure 6. Saturn Computer

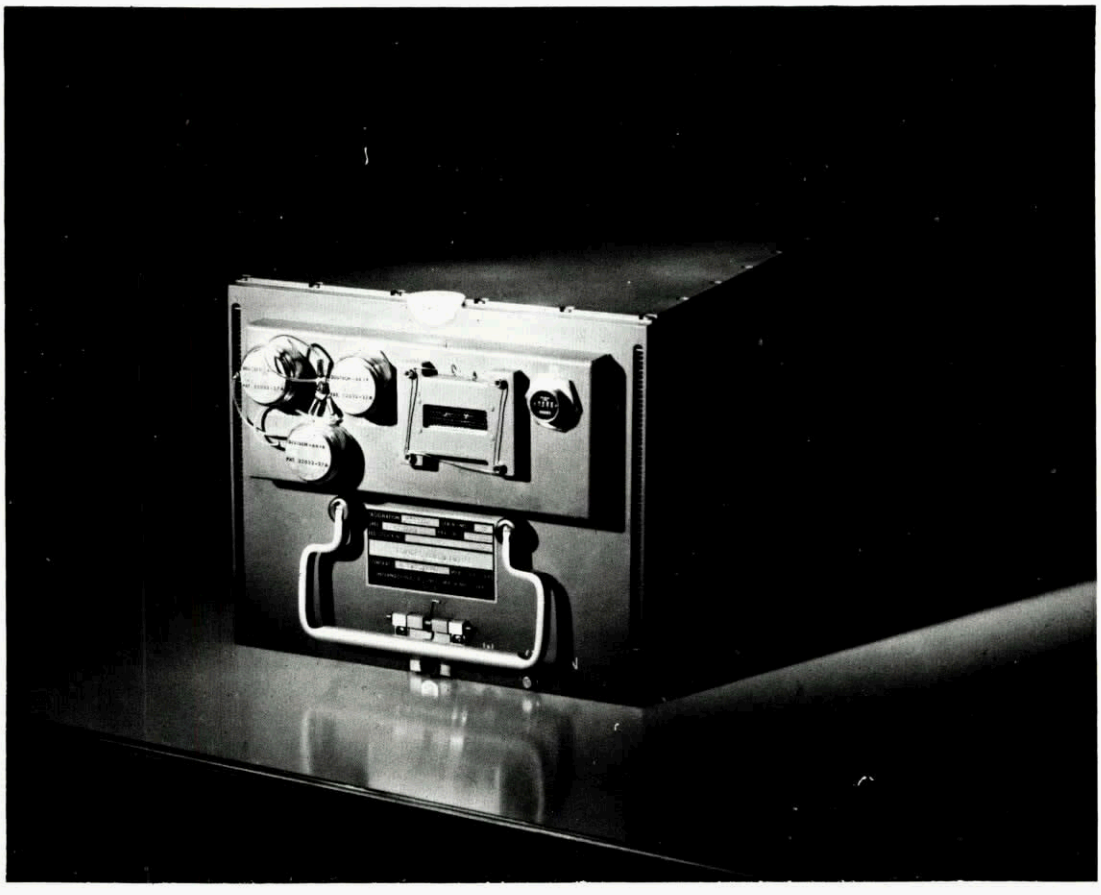


Figure 7. Mark II Digital Computer

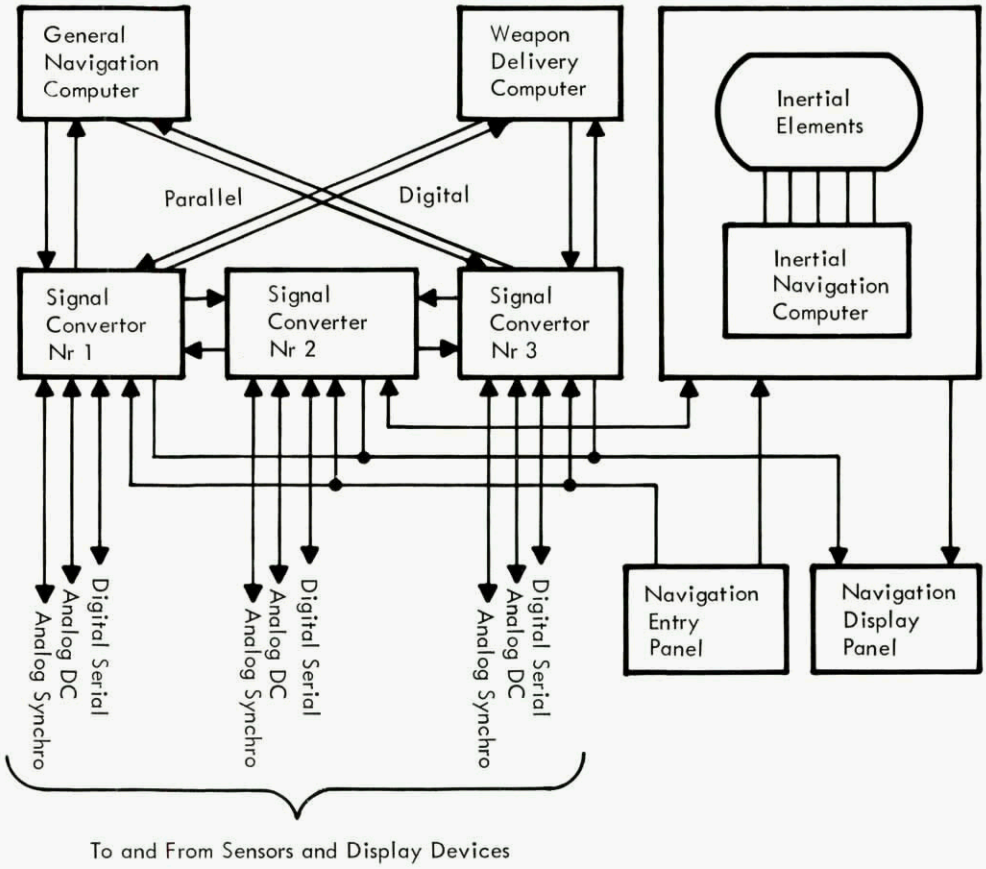


Figure 8. The Mark II Digital Computer Complex

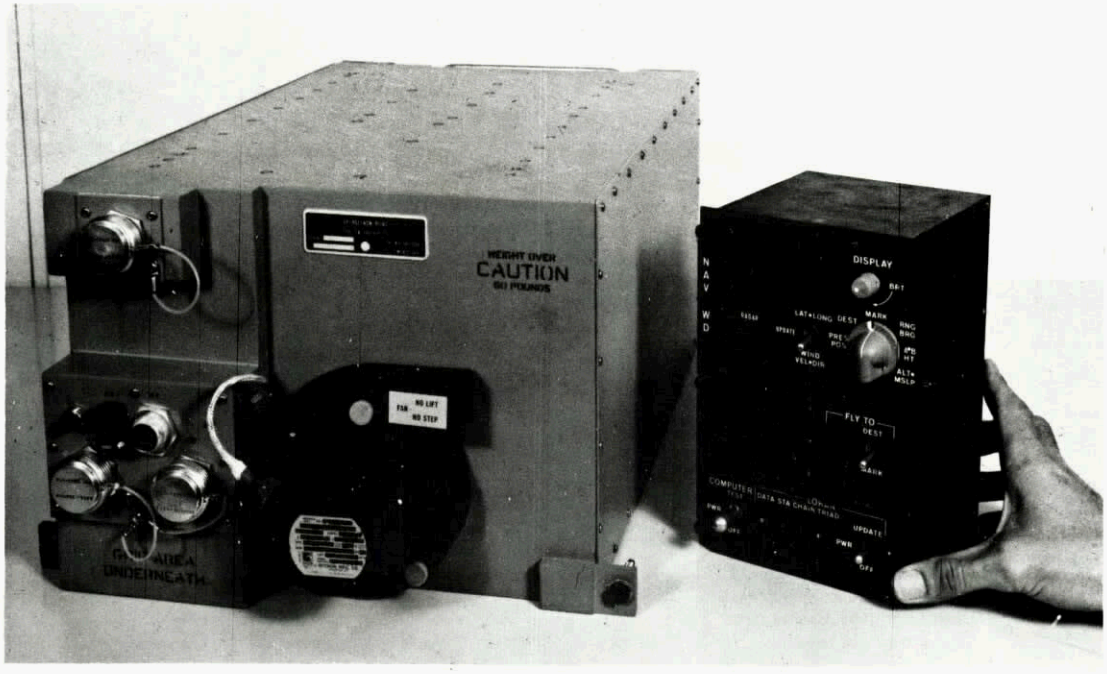


Figure 9. A-7D/E Avionics Computer

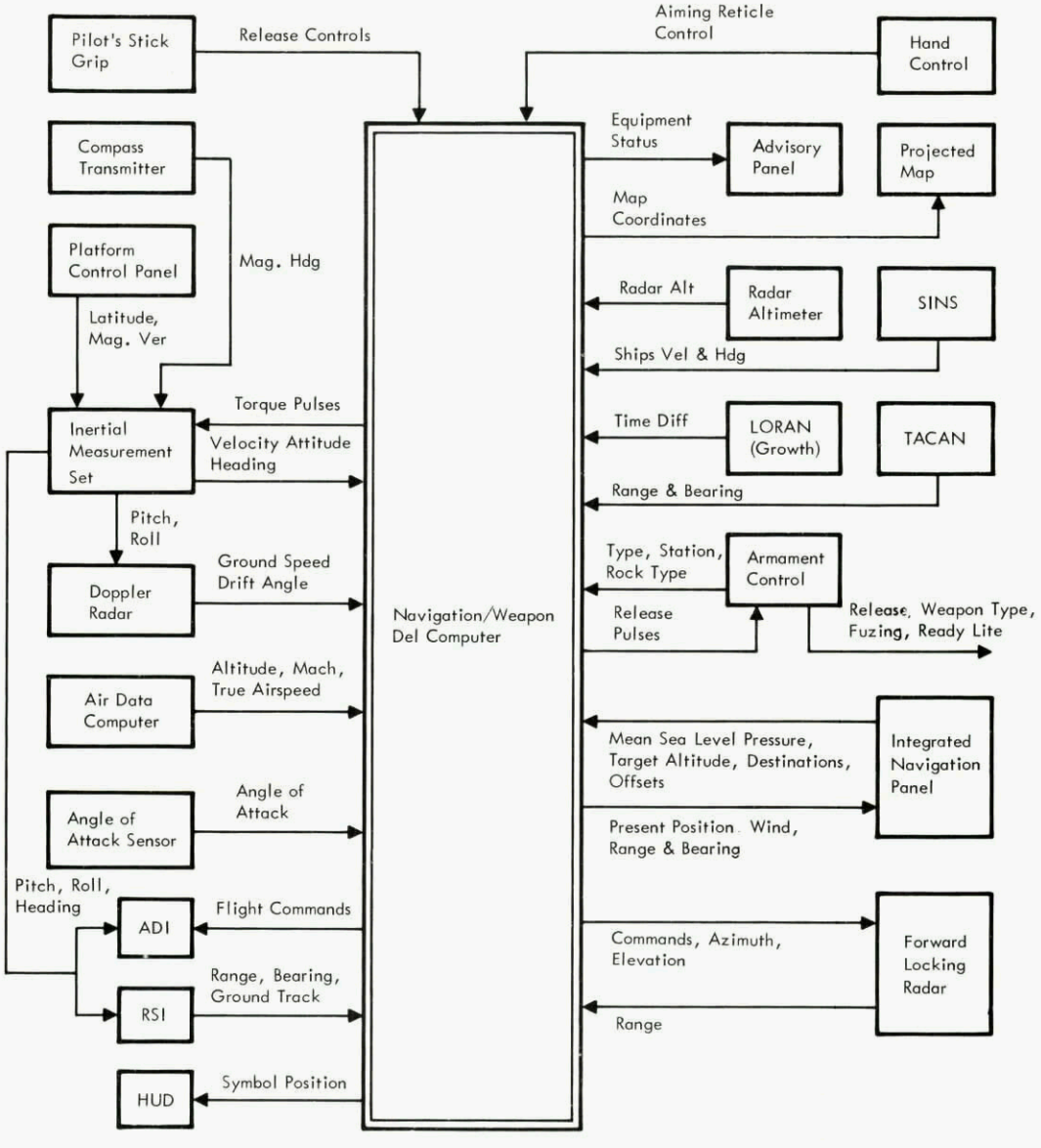


Figure 10. A-7 Avionics System Functional Diagram

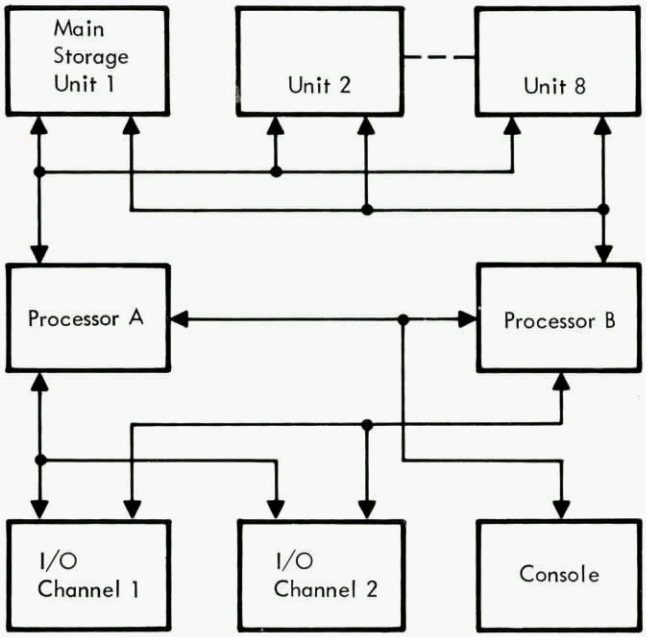


Figure 11. VS A-NEW Simplified Block Diagram

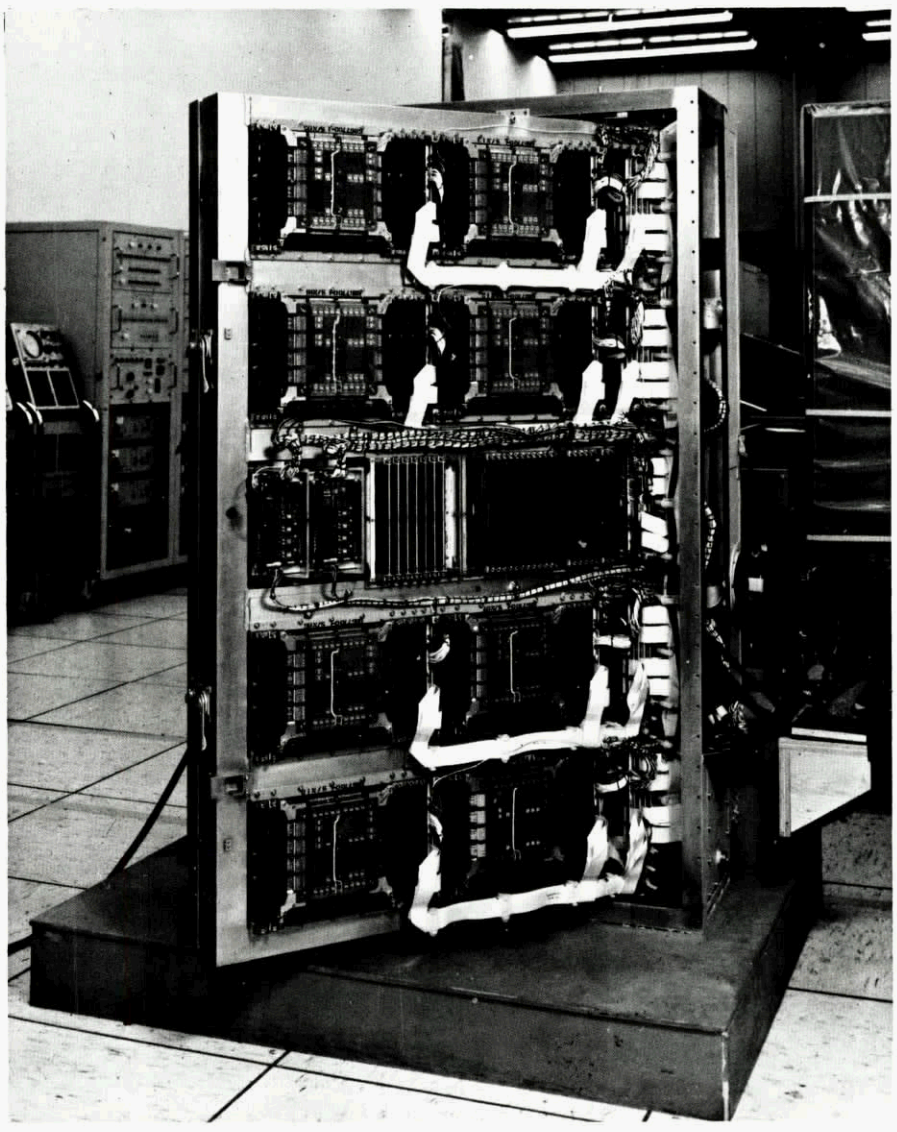


Figure 12. VS A NEW System

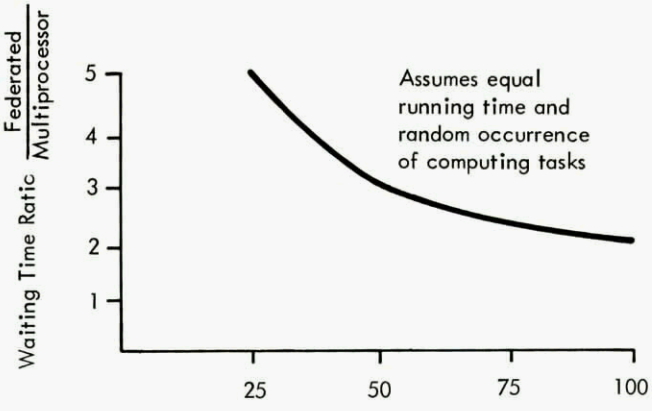


Figure 13. Waiting Time

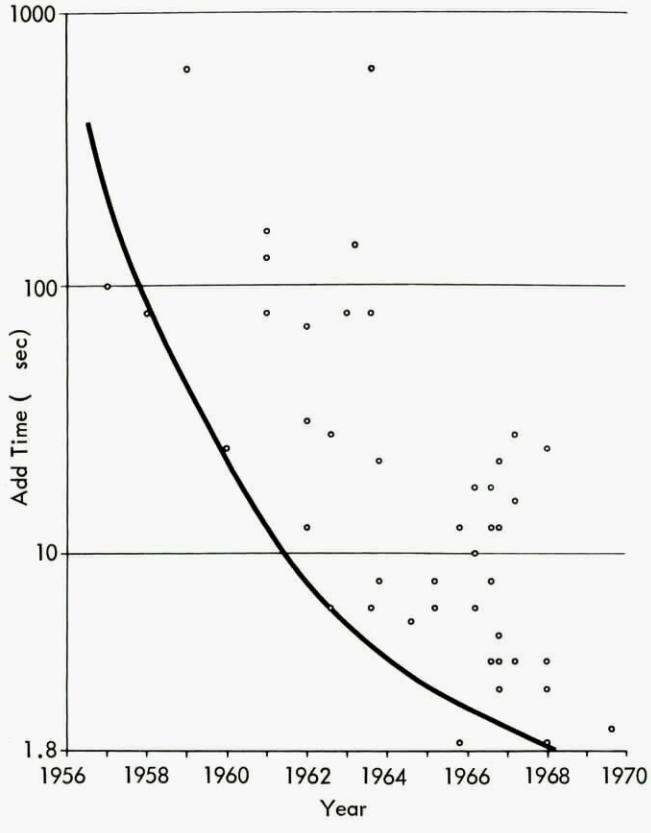


Figure 14. Add Time Vs. Year

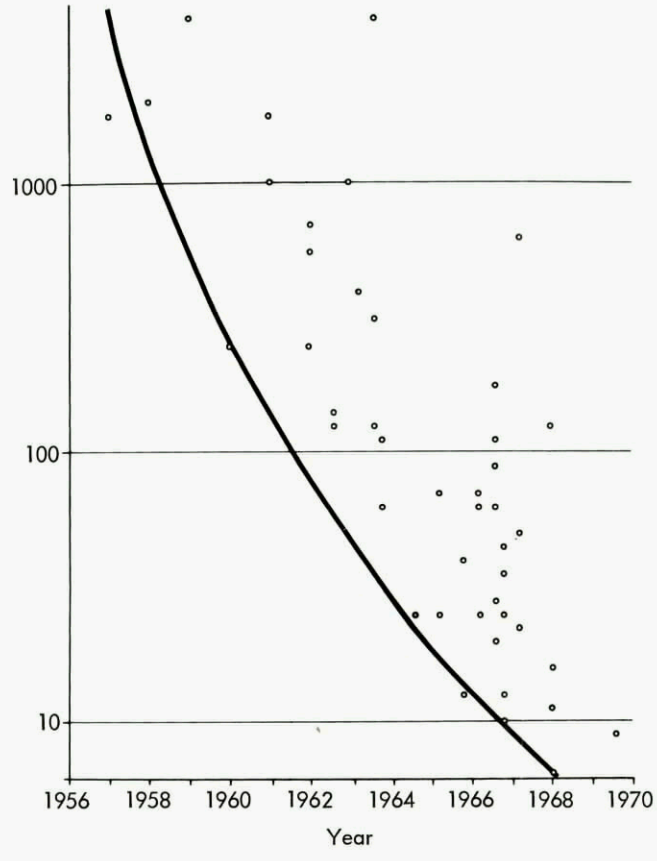


Figure 15. Multiply Time Vs. Year

PROGRAMMING CHARACTERISTICS OF FUTURE G & C COMPUTERS

by

Austin J.Maher

Kearfott Division
Singer-General Precision, Inc.

SUMMARY

The difficulties often encountered in programming digital computers have become legendary. Airborne computer programs, including those for Guidance and Control, have remained singularly difficult to develop despite the significant advances in the techniques for developing other computer programs. This is due in large measure to the austere instruction set provided in most airborne computers. The rationale for employing a minimum instruction set is based on the assumption that it is more important to avoid size and weight increases at the expense of programming ease (especially when this is a one time cost for production system). However, in Kearfott's experience on a wide variety of airborne computation problems, the impact of programming inflexibility can have serious impacts on the development and maintenance of an effective, flexible guidance and control system. Furthermore, it can limit the additional flexibility of growth through software and design improvements through software.

This paper first reviews the types of difficulties encountered in developing and maintaining airborne computer programs. These include problems encountered in the initial development of the system program and the significant changes to the original program based on laboratory and flight tests of the overall system. During this development phase, many system problem solutions result in a hardware/software tradeoff which usually forces a software change since software is more flexible than hardware. However, true software flexibility was sacrificed earlier when it was deemed to be less important than the virtues of a minimum CPU. In these final stages of system development, is is usually agreed that the choice of a more flexible CPU would now be advantageous.

As advances in MSI and LSI circuitry continue, the CPU becomes a less and less significant component in the overall computer size, weight, power, and cost. Consequently, it seems an appropriate time to ask whether for many applications it may be more effective to increase the complexity of the CPU to facilitate program development while using the advanced circuitry to maintain a sufficiently small airborne computer. Attention is also directed at the increased computational requirements (e.g., sensor mixing via Kalman Filtering) which are being imposed on airborne Guidance and Control computers and the techniques of computer organization which would facilitate their implementation. Many of these techniques were originally developed for high performance ground-based computers but have not yet been accepted by airborne computer designers for reasons of economy, or apparent economy.

By assimilating results of previous system development experience and predicting the trends in airborne computer hardware development, characteristics are proposed for future airborne Guidance and Control computers. The proposed computer characteristics would minimize significant programming difficulties while retaining desireable hardware features (e.g., protected memory, small size, etc.). The features covered include word length tradeoffs, type of arithmetic, addressing techniques, subprogram linkages, etc., and in summary represent a functional specification for a class of future airborne computers using MSI or LSI techniques.

Finally, the specific characteristics are presented of an airborne computer designed to meet these functional specifications. Although, this implementation does not uniquely satisfy the above functional specification (i.e., other designs might provide equal software facilities), it does provide all the desired programming characteristics. Consequently, it is a useful vehicle for illustrating the desireability and practicability of the proposed computer organization. The simplifying effect of the expanded CPU capability on the software development cycle is then covered in some detail. Of particular interest is the fact that high level languages (FORTRAN, JOVIAL, etc.) are naturally efficient in the specified machine while attempts at an efficient compiler language using a less sophisticated CPU are probably doomed to failure. This dependence of compiler efficiency on the organization of the airborne computer is a factor which is usually overlooked in recommendations for the use of problem oriented language to "solve" the airborne computer programming problem. The computer architecture must be carefully chosen to permit a compiler to generate efficient object programs, as Kearfott learned as early as 1962 when we developed a NELIAC compiler for the L90 airborne computer.

Hopefully, the voice of the software system designers, as articulated in this paper will not continue to go unheeded but will result in a series of airborne computers in the 1970's which provide true programming flexibility, and the enumerated advantages to the total system.

1. INTRODUCTION

In order to discuss future Guidance and Control computers, it is necessary to conjecture what significant hardware changes will be experienced in airborne computers in the period of interest. This is necessary since most changes in computer architecture arise in response to a change in the capability of the basic hardware components. Although the time scale might be subject to some debate, there is general agreement among airborne computer designers that the next major hardware change will be precipitated by advances in MSI and LSI circuitry. This will permit the realization of complex logic within progressively less size and weight. This increase in logic density can be exploited in two rather distinct fashions. First it can be envisioned as the accomplishment of the same logic functions within less size, weight and power. With the possible exception of an increase of execution speed, there is no increase in organizational performance. This is the more customary utilization of improved circuitry. Alternatively however, these technology advances could be equally effective in providing a significant increase in the capability of the CPU while retaining the same size and weight. For example, new, more powerful instructions might be added to the computer's repertoire to provide improved performance.

It is the purpose of this paper to advocate a path somewhere between these two extremes wherein the conventional austere instruction set of the airborne computer is strengthened by the addition of some powerful new capabilities but these are carefully structured and limited to also permit the size of the processor unit to be decreased.

Since my personal background lies in the area of software development rather than logic design or circuit design, the principal motivation for suggested organizational changes will be to facilitate the rapid development of accurate, operational software and to enhance the flexibility of that software once it is developed. The opinions expressed on the hardware implications of the suggested computer structure were reached through consultation with several airborne computer design engineers and reflect a rather non-controversial concensus of opinion on the expected direction of airborne computer hardware development.

2. AIRBORNE COMPUTER PROGRAMMING

The primary reasons for using digital airborne computers in G&C applications are their accuracy and their flexibility. The latter is ostensibly accomplished through modification of the computer program. However, making changes to airborne computer programs has proven to be significantly more difficult in practice than it is in concept. Consequently, much of the intended flexibility is either lost or achieved at a high price. Let us review the principal contributors to this "practical" problem as a prelude to searching for potential solutions.

Since virtually all airborne computers perform purely integer (or fractional) arithmetic, the value of the LSB (or MSB) must be chosen by the programmer for every data word including intermediate results. This process is commonly referred to as scaling. Proper scaling further requires that all arithmetic operations conform to the scaling rules appropriate to the operation. For example, if two numbers are to be added, their scale values must be the same. If two numbers are to be divided, the numerator integer must be larger than the denominator; otherwise, a divide overflow condition will be encountered. Finally, since the choice of scale factors and their manipulation can have serious implications on the numerical accuracy of the result, great care must be taken to adjust the scaling and/or the algorithm to minimize the numerical errors so induced. In short, scaling is a tedious, exacting job which is accomplished manually and is probably the greatest single contributor to aerospace programming problems.

Most assemblers for airborne computers generate absolute coding rather than relocatable coding. This again puts a burden on the programmer which might be alleviated. Upon closer inspection, it is usually found that the absence of relocatable coding in the assembler can be traced to characteristics in the airborne computer which make automatic relocation difficult or impossible to achieve. The most prominent of these is the presence of fixed memory pages in the computer organization.

The documentation problems associated with an assembly language (or machine language) program are usually more severe than if a higher level language were used. This is especially true for airborne computer applications which frequently imply time constraints, accuracy limits, etc. which are difficult to document even in a problem oriented language.

Finally, the compiler for a high level language (FORTRAN, ALGOL, JOVIAL, PL/I, NELIAC, SPL, etc.) is not available due to its almost unavoidable inefficiency, an inefficiency which is rooted as much in the architecture of the airborne computer as it is in any other source.

In short it would appear that the airborne computer software field is stalled at the technology level achieved by ground based machines in 1957. Hopefully, the expected advances in hardware technology, coupled with reflection on the problems noted herein, will result in a generation of airborne computers and support software vastly superior to that presently available.

3. HARDWARE CONSIDERATIONS

In order to develop a meaningful model for future G&C computers, it is necessary to review the hardware technology and to predict the general direction of its development if not its specific character. Toward that end, the most dominant factor is the continued rapid development of logic circuitry. In both the commercial and military field, logic circuitry continues to increase in speed and decrease in size.

Improvements have been much slower in memory technology, particularly for the military airborne environment. Although progress is being made toward memories based on LSI circuitry, these are generally limited to read-only memories (ROM) or volatile read/write memories. Since the most common airborne application calls for a non-volatile but electrically alterable memory, core memories will probably dominate in this area for many years to come. Advances in core memory technology for airborne environments has been gradual in the recent past and the future holds little promise for a dramatic improvement.

Consequently, since the cost of the memory and its electronics is already greater than that of the arithmetic unit, this disparity is almost certain to increase. This effect is compounded by the fact that the applications for airborne digital computers continue to increase in size (i.e., memory requirements).

It follows, then, that attempts to minimize the cost of an airborne computer for virtually any application should concentrate on minimizing the memory requirements. In fact, it is likely that a certain amount of increased complexity in the logic unit could reduce the cost of the computer if it promoted more efficient use of the memory. This factor serves as one of the bases for the recommendations made below on future computer architecture.

4. RECOMMENDED COMPUTER CHARACTERISTICS

Based on the above considerations some specific characteristics are now recommended for future airborne digital computers. We do not intend to discuss the minimum size machine used for special sensor processing or small missile guidance since these tend to have very special organizations which depend largely on the specific application. Rather we shall discuss the medium to large scale machines with correspondingly more severe software problems. Any synthesis of a machine organization must necessarily be an iterative process since design goals are usually (as in this case) somewhat in conflict with each other. For example, a powerful instruction set is desirable and a long instruction word is its simplest embodiment. Yet, a long instruction word is at variance with the goal of minimum memory. A full treatment of the iterative synthesis process is beyond the scope of this paper. However, we shall devote the following few paragraphs to a presentation of the computer characteristics which are felt to be of greatest value in aggregate. In addition, a brief indication of major interrelations between these characteristics and the design goals is presented to at least in part substantiate the efficiency of the resulting organization.

For most Guidance and Control applications, a long basic data word is advocated (say 24 to 32 bits). This assures sufficient accuracy for most of the critical G&C computations. Although it significantly increases the data area memory, this usually represents less than 10% of the computer's full memory and thus the overall impact is slight. Moreover machines with short data word length (14 to 18 bits) require that two words be used to represent critical variables. In fact, for most cases, the apparent increase in memory caused by implementing a long data word will be more than offset by savings in the storage of critical variables and by economies in the instruction memory due to:

a) elimination of extended precision requirement and its associated instruction increase.

b) simplification of complex algorithms employed to avoid error propagation due to short data word length.

Since some data can be represented in short data words, it is tempting to suggest that both long and short options be available. However, this would then require the ability to address half words which would require a longer address field in each instruction. Consequently, purely long data words are recommended with the realization that any substantial quantity of short data words (e.g., ballistic data tables) can be accommodated by appropriate packing and/or unpacking of data through subroutines or macro operations.

Two major innovations in ground-based scientific computers were introduced in 1957 which revolutionized the art of software development: the FORTRAN compiler and floating point arithmetic hardware. The floating point hardware provided the solution to many numerical analysis problems but more importantly it assured a substantial increase in the efficiency of FORTRAN programs. This same revolution is long overdue in airborne computers. Consequently, the incorporation of floating point arithmetic hardware is strongly recommended based in part on the realization that compilers for airborne computers will be severely limited without fast floating point operations. Even before compilers are available, however, the requirement to scale all arithmetic operations will be eliminated. This will not only facilitate the development of airborne computer programs but will greatly simplify their documentation thereby facilitating the expeditious preparation of program changes. The recent advances in mathematical techniques for optimum sensor data mixing

for guidance and navigation (i.e., Kalman Filtering) have placed even greater demands on the programmer's ability to scale fixed point operations without losing accuracy. These operations become straightforward with the availability of fast floating point arithmetic.

Furthermore, the cost of this additional capability should be minimal if proper advantage is taken of new MSI and LSI circuitry. It should also prove convenient in this implementation that a long basic data word length was chosen. For example, based on a 32 bit data word, the floating point word could consist of an 8 bit exponent field and a 24 bit mantissa or magnitude field.

Although we have specified long data words for our computer, it is almost imperative that short instruction words be used. This is based on the desire to minimize the amount of expensive memory that is required and the knowledge that 80 or 90% of the memory consists of instructions. For example, if a 32 bit data word was chosen, a 16 bit instruction word would be desirable with either 16 bit or 32 bit parallel access to a memory word. However, later considerations will indicate a requirement for instructions longer than 16 bits. A suitable compromise would be the availability of both short (16 bit) and long (32 bit) instructions where the operation code would determine the length of the instruction. Note that our earlier decision to limit the machine to long (32 bit) data words now results in the availability of an additional bit in the address field of short instructions which access data, thus doubling its range of applicability. One design goal for the machine should be a sufficiently powerful set of short instructions to reduce the need for long instructions to a minimum.

A similar design goal obtains in the architecture of third generation ground-based computers. Whereas second generation computers (IBM 7094, Univac 1107, etc.) permitted the address field of each instruction to designate any word in the computer's memory (32 or 65K words), this luxury could not be afforded in third generation machines whose memory might consist of several million words, requiring an address field of 21 bits or more. Consequently, designs were developed which make extensive use of pointer registers to permit short instructions with limited address fields to access the entire memory. These techniques are directly applicable to airborne computer design.

While the use of pointer registers permits a short instruction to obtain operands from any region of memory, a similar problem must be solved for all jump or branch instructions. The ability to jump to any location in memory is most conveniently provided via the use of an optional long instruction format. A short jump instruction (with an m bit address field) may be implemented in one of two common ways:

1) Add or subtract the m bit address field to the instruction counter. This provides a jump range of $\pm 2^m$ around the present value of the instruction counter.

2) Replace the m least significant bits of the instruction counter with the m bit address field of the instruction. This permits a jump to any location within the block of 2^m instructions containing the jump instruction.

The second approach is more common in present airborne computers since the replacement operation is simpler to implement than the addition or subtraction called for in the first approach. However, the first approach is recommended here; again hoping that the use of advanced circuitry will virtually eliminate the potential hardware penalty. This recommendation is based on still another software design goal: that it be possible to assemble relocatable subroutines without constraint. A relocatable subroutine is a sequence of instructions which can be loaded into any portion of the computer's memory either without change or with the addition of an address displacement to the address field of certain instructions. However, no structural changes are permitted in the instruction sequence. The second technique for implementing a short jump instruction makes it impossible to determine whether a short jump instruction can be used unless it can be guaranteed that both the jump instruction and its target location lie in the same block of 2^m instructions. Note that the boundaries of this block are fixed memory locations at addresses which form a multiple of 2^m. Consequently, the condition for using short jump instructions cannot be guaranteed independent of the relocation address. This method should therefore be discarded in favor of the first approach.

Still another constraint on the design of our advanced G&C computer is the desire that the program consist entirely of "pure code". In other words, the instruction portion of memory is constant and cannot be altered during the execution of the program (except indirectly via the use of index registers). There are several motivations for this constraint, namely:

1) To permit the use of a ROM (read-only memory) to assure the integrity of the program.

2) To permit verification of the integrity of the program in a read/write memory via check-sum tests, etc.

3) To avoid the catastrophic errors which can arise through instruction modification and to avoid the overly complex program structures which it fosters.

4) To provide the necessary basis for the development of reentrant subroutines which can be "simultaneously" used by several processors or on several interrupt levels.

Some common instructions inherently violate the concept of a constant instruction memory and therefore must be avoided. The most common of these is the subroutine jump which stores the return address at the first location in the subroutine. Other instructions which facilitate violations of pure code (depending upon how they are utilized) should either be avoided or their usage carefully controlled. One such instruction is the memory increment instruction which provides a tempting alternate to the use of an index register or pointer for address modification. However, it can be safely used to increment counters and other variables in the data area.

Once provisions have been made for the creation of pure code, reentrant subroutines can be developed simply by careful allocation of the variable data areas. Reentrant routines have an obvious advantage when a program involves several interrupt levels or multiple processors. They permit the various independent programs to share common subroutines rather than requiring multiple copies of the same subroutine for each independent program. In addition to this obvious memory economy, Kearfott uncovered a more obscure but more generally useful feature during a study of reentrant subroutine structures. One memory economy which is introduced manually into airborne computer programs is the sharing of temporary (scratch) data storage between subroutines on the same level. This sharing is accomplished automatically via the memory allocation process when reentrant routines are used. In fact, more extensive sharing is possible under the automatic process since no opportunities for sharing are overlooked by the memory allocation algorithm. When the sharing is accomplished manually (or more precisely, mentally,via the thought processes of the programmer), full sharing is rarely accomplished. Note that this advantage is obtained even though the "reentrant" routines are never in fact reentered, since it is a characteristic of the memory allocation process itself.

To summarize the characteristics of the recommended future airborne G&C computer we recall that it shall:

1) Employ a long (32 bit) data word

2) Provide fast floating point arithmetic in hardware

3) Permit both short (16 bit) and long (32 bit) instructions

4) Include many index registers or pointers

5) Eliminate fixed instruction block boundaries

6) Employ strictly pure code and reentrant routines if possible

where certain specific parameters were chosen for simplicity. It is urged that the advances in modern circuitry be used to develop a family of future computers with the recommended performance rather than to continue to pursue the unimaginative course of simply shrinking the CPU (Central Processor Unit) beyond the point of significant payback.

5. THE ROLE OF COMPILER LANGUAGES

Several organizations have recently been studying the use of one or more high level (compiler) languages for the development of airborne computer programs. These studies have been motivated by the well known difficulties encountered in developing and controlling airborne computer programs in machine language (or assembly language), coupled with the knowledge that much less difficulty is encountered in developing rather complex programs for land based machines using compiler languages. Many of these efforts have narrowed their scope to a search for the optimum language (e.g., the specification of SPL by SDC). Although these efforts are meritorious, it is erroneous to assume that the identification of an appropriate language will alone improve the development cycle for airborne computer programs. The more serious problem is the fact that the object code created in the compilation process is invariably inefficient and an improvement in the language will not substantially alter this process.

In 1962, Kearfott developed a NELIAC compiler for its L90 airborne computer. The language was then used to develop an inertial navigation program and its use uncovered the two problems which continue to limit the use of compilers for airborne computers, namely:

1) The continued dependence on the programmer to scale the fixed-point arithmetic operations required. This process is error prone, difficult to document, and is the dominant problem in processing requests for program revisions.

2) The inefficiency of the object code. Although this depends somewhat on the structure of the compiler program, it is primarily dependent upon the architecture of the airborne computer.

The recognition of these important considerations had a significant effect on the choice of computer characteristics recommended earlier in this paper. For example, the inclusion of floating point arithmetic hardware is imperative for the effective implementation of FORTRAN or any other algebraic language. Attempts to implement an algebraic language without this hardware can only be accomplished in one of two ways:

1) Use subroutines to accomplish the basic floating point arithmetic operations. The speed of these subroutines will be approximately ten times slower than the same operation accomplished via hardware. Although this speed penalty may be acceptable for a few applications, it generally cannot be tolerated.

2) Eliminate floating point operations from the algebraic language. Since scaling for integer (or fixed-point) arithmetic is the major problem in G&C applications, the resulting language will not provide a suitable solution to this problem.

Other aspects of machine architecture should also receive careful attention if efficient compiled object code is a design goal. For example, relocation of program segments should be readily accomplished without regularly resorting to long instructions. The elimination of fixed memory blocks or pages for short jump instructions is one step in this direction. Also, the hardware/software convention for calling subroutines should be carefully chosen, in particular the technique for transmitting arguments (either by name or by address or by value).

The provisions for pure code, which permits the development of reentrant subroutines, can contribute to efficient object code. In fact, if reentrant subroutines are widely used, substantial economies will be realized in temporary (scratch) data allocation. This feature might well be copied by the algebraic compilers on large ground-based machines which largely ignore this possibility (one exception appears to be the FORTRAN compiler for the XDS Sigma 7 computer).

Even if all these precautions are taken, machine (or assembly) language programming will still have its advocates, just as it did in 1957 when FORTRAN was first introduced. They will claim that any compiler is somewhat inefficient and that this inefficiency cannot be tolerated in their application. This objection, which will frequently be emotional but just as frequently be valid, can easily be circumvented by designing the compiler program to accept in-line machine code in symbolic assembly language. Consequently, should a critical computation be compiled "inefficiently", it can be replaced by hand written "efficient" coding. I'm sure many of you will recall that this technique was widely used in early FORTRAN compilers for precisely the same reason. It should not be necessary, then, to implement automatic optimization algorithms such as those used in the 360 FORTRAN H compiler. However, these could also be developed whenever it is justified economically.

In brief, then, algebraic compilers should see widespread application to G&C systems once the appropriate airborne computers are developed. If future airborne computer designers do not provide the facilities necessary for efficient compiled object code, algebraic compilers will continue to see limited service in the G&C field.

6. CONCLUSION

Although time did not permit explicit consideration of many alternatives in airborne computer architecture (microprogramming, parallel processors, stack operations, multiprocessing, etc.), the conclusions of this paper should not be considered tentative or limited. The principal message lies in the considerations upon which the recommended computer structure is based. This had largely to do with the flexibility of the computer from the software viewpoint and the attempts to conserve memory via appropriate structuring of the central processor unit (CPU). The specific structure discussed in paragraph five should be considered more an example than as a specification. Any alternative configuration that satisfies the same goals would be equally acceptable.

In this spirit, then, it is appropriate to conclude by summarizing the principal design goals proposed, and to recommend their careful consideration in the synthesis of any new computer design. The principal design goals include:

a) elimination of the scaling problem (via floating point hardware)

b) efficient use of memory (via short instructions and reentrant routines which share scratch area)

c) use of protected memory (via pure code which also facilitates reentrancy)

d) subroutine relocation facilities (via pointer registers and relative jump instructions)

e) facilitate efficient compilation (via efficient subroutine calling sequence, floating point hardware, reentrancy facilities).

Based on the hope that these factors receive due consideration, the computers developed in the 1970's will significantly improve the problems heretofore encountered in developing software for airborne G&C computers. A likely bonus is the proliferation of efficient compilers for high level languages and their subsequent widespread use in developing airborne computer programs.

AEROSPACE COMPUTER WORD LENGTH CONSIDERATIONS

by

G. W. Braudaway

IBM Corporation,
Federal Systems Division,
Electronics Systems Center,
Owego, New York 13827, USA.

SUMMARY

This paper describes a number of data-word and instruction-format factors that must be considered in selecting a common word length in which both are stored. Examples drawn from avionic navigation computations are used to illustrate the premise that variable word length arithmetic can be used to produce numeric results equivalent to those produced by uniform word length arithmetic while reducing storage requirements substantially. Additional storage savings can result from use of variable length instruction formats. It is urged that additional emphasis be placed on developing techniques to use variable word length arithmetic and instruction formats efficiently.

AEROSPACE COMPUTER WORD LENGTH CONSIDERATIONS

G. W. Braudaway
IBM Electronics Systems Center
Owego, New York 13827

INTRODUCTION

One of the objectives of aerospace mission analysis is to identify and quantify the required onboard computer performance characteristics. Typical characteristics include instruction execution times, data-word lengths, and storage capacity. Such characteristics have a fundamental influence on hardware design. All too often, however, these characteristics are determined by oversimplified and extremely subjective analyses. This has been true especially in the determination of data-word length, where results of simulation studies have been tabulated as a "word-length" requirement for various mission functions. (1) Such simplified studies, usually based on a single inviolable numerical implementation, imply that a uniform data-word length is required throughout all calculation. * This is not true. Also, program storage estimates generally do not use the fact that many current generation aerospace computers employ instructions that are not of uniform length, and thus can store two or more instructions in the space required for one operand. (2) As a result, little emphasis has been placed on developing techniques that exploit variable precision arithmetic and multiple length instruction formats to increase storage and execution efficiency. Because improvement in storage efficiency potentially reduces hardware requirements, improves reliability, reduces power dissipation, or provides growth for new functions, adequate word length and instruction format selection techniques must be developed.

DATA-WORD LENGTH

In the conceptual phase of design, the systems analyst must decide the degree of preciseness required of the various measured or computed parameters so that overall system effecriveness is not compromised. Parameter errors are attributable to two sources, (1) environment measuring devices and (2) numerical computation required to process raw measurement data into useful results. The source over which control can most easily be exerted is the numerical processing of measurement data. The software implementor's principal objective is, then, to implement the required computational procedures so that the preciseness of computed parameters is not degraded. Generally, the error contribution from numerical processing should be of the order of one percent of the error budgeted to any specific parameter.

All too frequently, statements such as the one which follows are heard: "To do inertial navigation, a numerical word length of 24 bits is required." Such statements are inaccurate and misleading, to say the least. Just how misleading they are becomes apparent when one recalls that a Turing machine with a data word length of one bit is capable of computing any quantity which is computable. What the statement should say is: "The way in which the numerical process for inertial navigation has been implemented requires a data word length of 24 bits to insure a negligible error contribution from compuation." The key word in this statement is "implemented." Any method of implementation (and there generally are many) which meets precision requirements can claim no more than to have established an upper limit on the number of instructions and data-word length required. It is extremely risky to claim that any method of implementation has established a lower limit on either the number of instructions or data-word length.

One fact which is usually overlooked when "word-length requirements" are specified is that, in general, the numerical precision which can be obtained by almost any programmable digital computer is unlimited, regardless of what basic data-word length is used. This can be demonstrated easily by merely appending successive data words together, with the understanding that the magnitude of the first bit of the second word is half that of the last bit of the first word, and that the magnitude of the first bit of the third word is half that of the last bit of the second word, and so on.

Representing a value by two and three data-words is commonly referred to as double-precision and triple-precision respectively. All that is needed to complete the unlimited precision argument is the formulation of algorithms for addition, subtraction, multiplication, and division of multiple-precision operands in terms of single-precision operations. This can be done in a straight forward manner, but the algorithms for multiple and especially divide become quite cumbersome for operands of more than two basic word-lengths.

Another facet of implementing a numerical process is the effective use which can be made of varying the precision of arithmetic computation within the process. Nearly all precision analyses made to date have required the data-word length to be uniform. However, it will be shown in a subsequent example of inertial navigation implementation that a basic word length of 15 to 16 bits (including sign), when coupled with a very insignificant amount of double precision arithmetic used exclusively in the accumulation of velocity and position integrals, produces results which are as precise as those produced by an implementation which uses a uniform data-word length of 28 to 30 bits (including sign). Therefore, since the implementation using the shorter operands and very limited double precision has established an upper

I wish to acknowledge the useful comments concerning variable format instructions made by William Patzer, Senior Engineer, IBM Corporation, Federal Systems Division, Owego, New York.

* The discussion given here is based on the use of fixed-point arithmetic typical of nearly all present day aerospace computers rather than on hardware-implemented floating-point arithmetic.

bound on required complexity, any implementation which requires a longer data-word length is wasteful of the computing resource.

A digression is in order to demonstrate the variable precision arithmetic technique. In the evaluation of expressions which define variables whose resultion must exceed that of the basic word length being used, the use of multiple precision is required. However, in most cases all terms in the expressions defining these variables need not be computed with multiple precision arithmetic. Consider, for example, the expression for the approximate radius of the geoid as a function of latitude. This expression can be written in series form, and ignoring terms of order e^3 and greater is:

$$R_e = R_o \left\{ 1 - (e - 5e^2/2) \sin^2 \phi - (5e^2/2) \sin^4 \phi \right\}.$$

where R_o is the equatorial radius and e is the ellipticity of the geoid, and ϕ is latitude. The maximum error in the truncated series is about 0.428 feet at a latitude of 59°. The equatorial radius of the geoid is about 2.09×10^7 feet , so if an overall resolution of ±2 feet is specified, a numerical word length of at least 24 bits is needed. However, the maximum magnitude of the term containing e is about 69,600 feet and the term containing only e^2 is about 588 feet. Hence, a word length of only 16 bits is required to contain the term containing e, and a word length of 9 bits is required to contain the term containing only e^2. If the computer has a basic data-word length of 16 bits, it would be necessary to represent the radius of the geoid in double precision; that is, using 32 bits. It is evident, however, that each of the variable terms can be computed in single precision and added to a constant term, stored in double precision, after an appropriate right shift to align their binary points. The only double precision instruction required is "Add".

To show the effectiveness of variable precision arithmetic, a typical north-slaved, altitude-damped aircraft inertial navigator was implemented using fixed-point binary arithmetic. Double precision arithmetic was used only for the accumulation of the velocity and position integrals. Data-word lengths of 15 and 16 bits (including sign) were used. The results of this analysis are shown in subsequent graphs for the various word lengths on a typical 90-minute great circle flight at constant speed and altitude. See Figures 1 and 2. The errors shown are due solely to computation; the inertial platform was carefully simulated so that its error contribution was negligible.

To contrast these, a second implementation of the same navigator was made in which all calculations were performed with a uniform data-word length. Results for word lengths of 28 and 30 bits are shown in Figures 3 and 4. It can be seen that the preciseness of the 16 bit implementation with double precision accumulation is equivalent to the preciseness of the 30 bit solution with no double precision. (The rapid variations in speed and velocity-heading are due to the resolution of platform velocity-meter inputs, which was 0.03 feet per second.) It is obvious that significant storage savings can be made using the 16 bit word length with limited double precision. An even more important result is that only 16 bit arithmetic need be implemented in hardware.

Results of this nature are typical for nearly all aircraft and spacecraft problems including navigation, platform alignment, calibration, etc. Even for Loran navigation, where the quality of raw measurement data would require a data-word length of about 24 bits, extensive 24 bit arithmetic is still not generally required, but is used in critical calculations only. It is estimated that between 80 and 90 percent of all aircraft and spacecraft data processing computation can be performed with data-word lengths no longer than 15-16 bits. Shorter word length would require much more extensive use of double or multiple precision arithmetic; longer word lengths would squander computing hardware.

INSTRUCTION WORD LENGTH

When selecting word lengths one must also consider instruction formats and the possibility of using variable length formats, since instructions must generally be stored in the same length words as data. However, instruction format selection is not without its pitfalls. If the instruction format is too short, operand addressing limitations are imposed. To circumvent those limitations increased use must be made of indexed addressing, which is usually accompanied by a performance loss. A format that provides little used addressing capability is a poor choice because it is wasteful of instruction storage space. With the number of logic circuits held constant, a trade-off exists between variable instruction format lengths with improved storage efficiency or uniform instruction length with higher performance. A small investment in additional circuits can often maximize improvement in storage efficiency and minimize degradation of performance.

A word length too short to permit storage of two or more instructions per word, yet longer than that required for a single instruction, would also be a poor choice. The word length must be selected to minimize total storage requirements, instruction and data, without unduly sacrificing necessary processing speed.

STORAGE EFFICIENCY

Figures 5 and 6 represent the storage and execution "efficiency" of a fixed-point computer as a function of basic data-word length when applied to a typical aerospace data processing job. It is assumed that the magnitude of the data processing job is invariant. It is also assumed that each implementation produces results of sufficient or more than sufficient preciseness to meet system requirements. It is assumed that the same basic instruction set is used throughout; only the data-word length changes. It is also assumed that more instruction executions are required to perform double-precision operations than single-precision, but that the instruction storage required to invoke each double-precision arithmetic subroutine is, at worst, no more than twice that required for a single-precision arithmetic operation. The rather modest amount of instruction storage required for the reusable multiple-precision arithmetic subroutines is assumed to be small compared with total program storage.

Figure 5 shows the number of instruction executions required as a function of the basic data-word length. When the data-word length is 1 bit, the implementation is that of the Turing machine. This implementation is least efficient in terms of the number of instruction executions required to do a specific job. As a word length increases from 1 bit, the number of instruction executions required decreases. This decrease continues until the data word length is large enough so that no double precision operations are necessary and the number of precision-motivated scaling shifts has decreased to zero. For most aerospace applications the number of double precision operations required will have dropped to zero with a data-word length of 30 bits or greater. The number of precision-motivated scaling shifts may decrease slightly beyond that point but not significantly.

Figure 6 represents the data storage required as a function of the basic data-word length. For basic data-word lengths greater than 30 bits, storage size increases linearly with the word length to accommodate the longer data words. The significant upward increment in storage capacity near the 30-bit word length represents the necessity of including more double-precision operands to maintain the required degree of preciseness. From 30 bits, the curve varies roughly linearly as the word length decreases to approximately 15 or 16 bits. Below 15 bits, the curve expands upward due to the use of multiple precision arithmetic. As the data-word length decreases to one bit, the storage required for data approaches the theoretical minimum typical of the Turing machine. But as the word length approach this limit, execution efficiency greatly decreases because of the excessive number of storage references and data-word manipulations required, as mentioned above.

CONCLUSIONS

In selecting a data-word length for a particular computer, one must study the application to determine the relative importance of computing speed versus storage efficiency. Long data-word lengths can be a poor choice since their manipulation requires excessive computer logic circuitry; also they can produce numerical results of superfluous preciseness and hence use storage inefficiently. Shorter data-word lengths may require use of double-precision arithmetic algorithms thereby reducing computing speed. Too short a word length can require extensive use of double- or multiple-precision arithmetic with an excessive penalty in computing speed. The optimal data-word length must consider all of the factors mentioned and include consideration of the efficient storage of computer instructions within the same memory.

Double- or half-precision arithmetic operations must be considered in determining a nearly optimal data-word length. Highly subjective "precision-analyses" employing a single numerical implementation of the problem in question and utilizing a uniform data-word length will always result in an excessive specification for the required data-word length, and should therefore be regarded as nothing more than a means of establishing an upper limit on requirements.

To insure efficient utilization of non-uniform data-word lengths and instruction formats, programming aids must be developed to optimize format selection for operational programs.

REFERENCES

1. "Horizons in Guidance Computer Component Technology", Maurer, Harold E. and Ricci, Robert C., IEEE Transactions on Computers, Vol. C-17, No. 7, July 1968, pp. 621-634.

2. "Structure of An Aerospace Computer Family", Patzer, William J., Airborne and Spaceborne Computer Computer Lecture Series, published in 1969 under the sponsorship of the Advisory Group for Aerospace Research and Development - NATO.

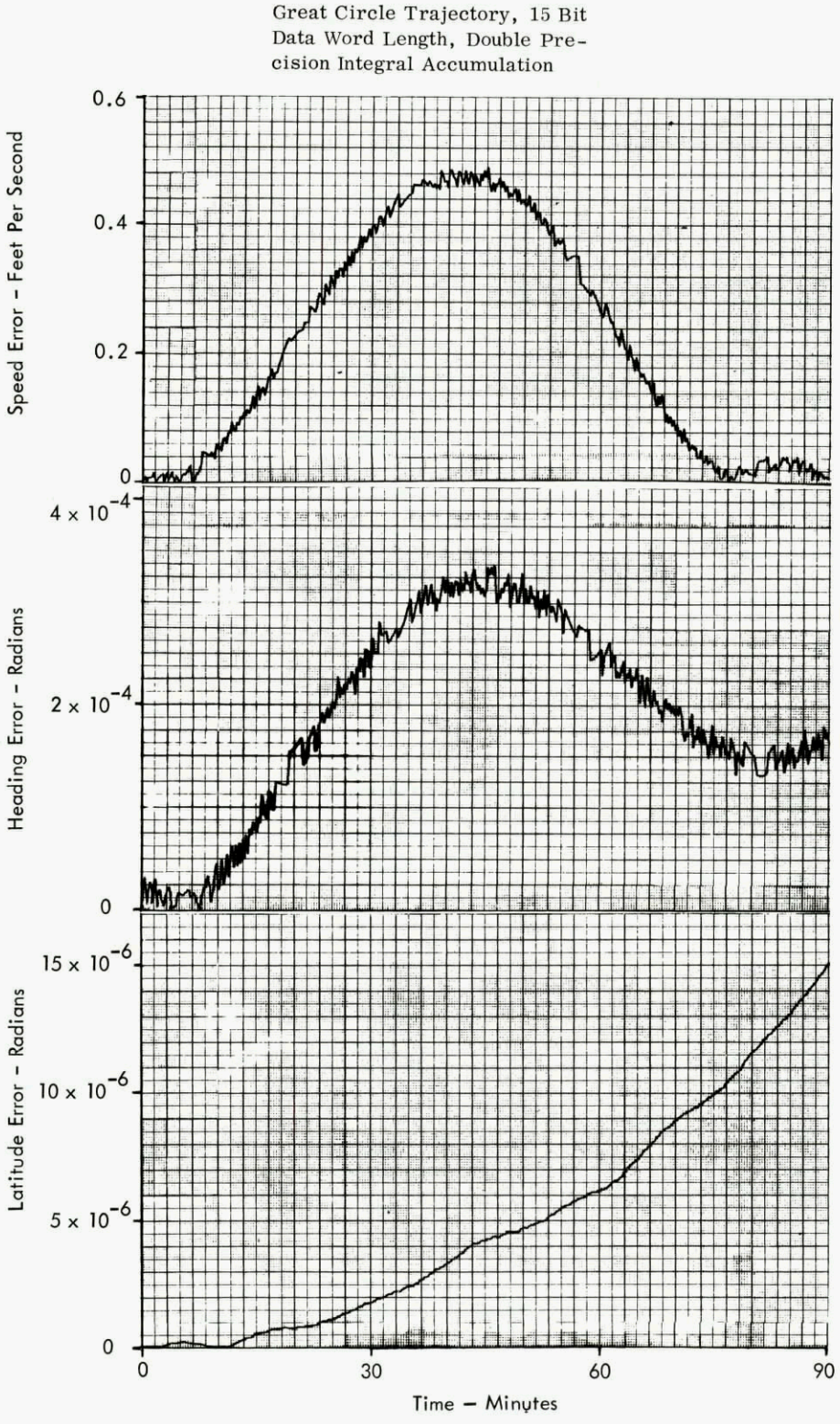


Figure 1. Errors From Computation in a North-Slaved Inertial Navigator

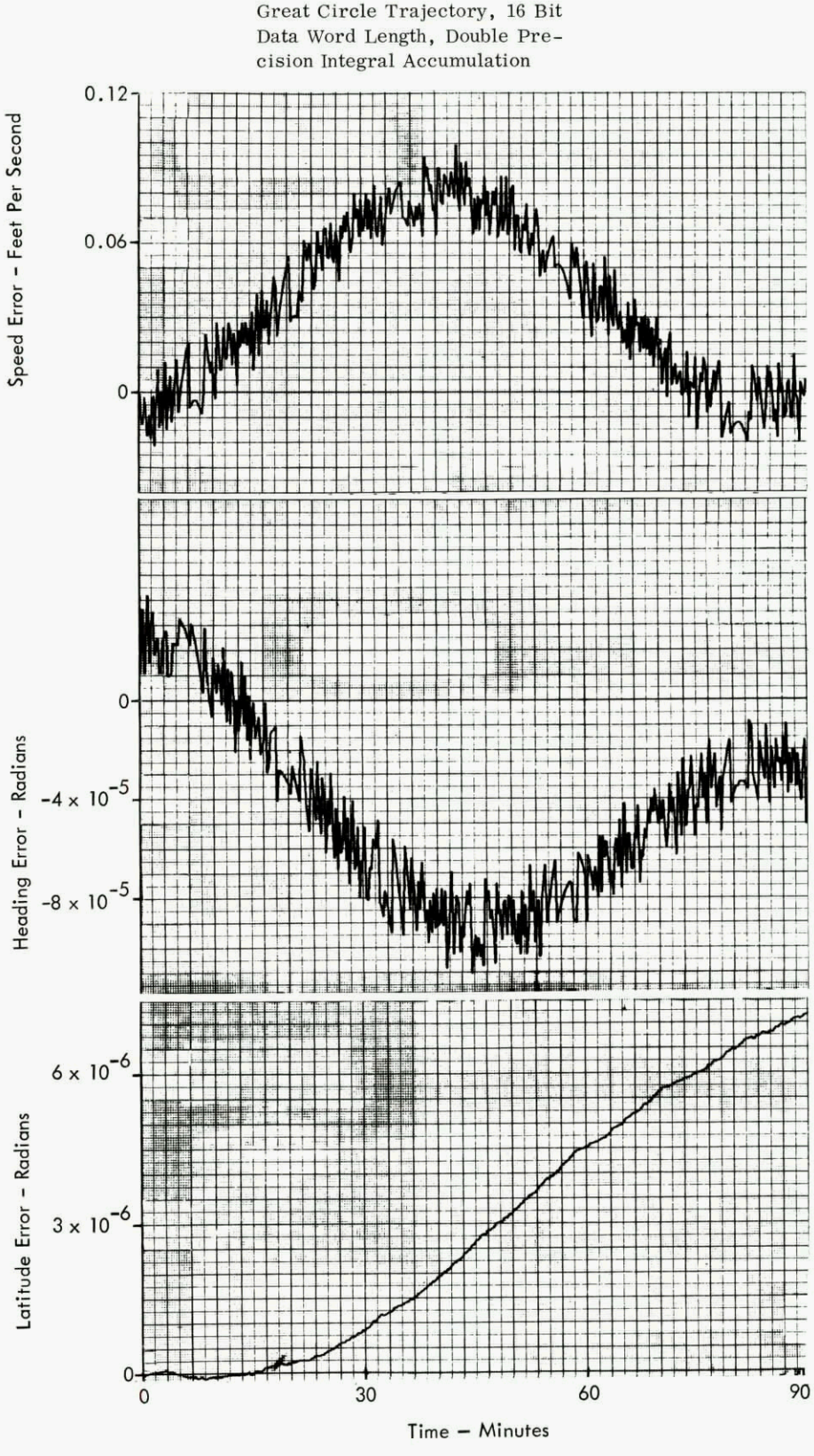


Figure 2. Errors From Computation in a North-Slaved Inertial Navigator

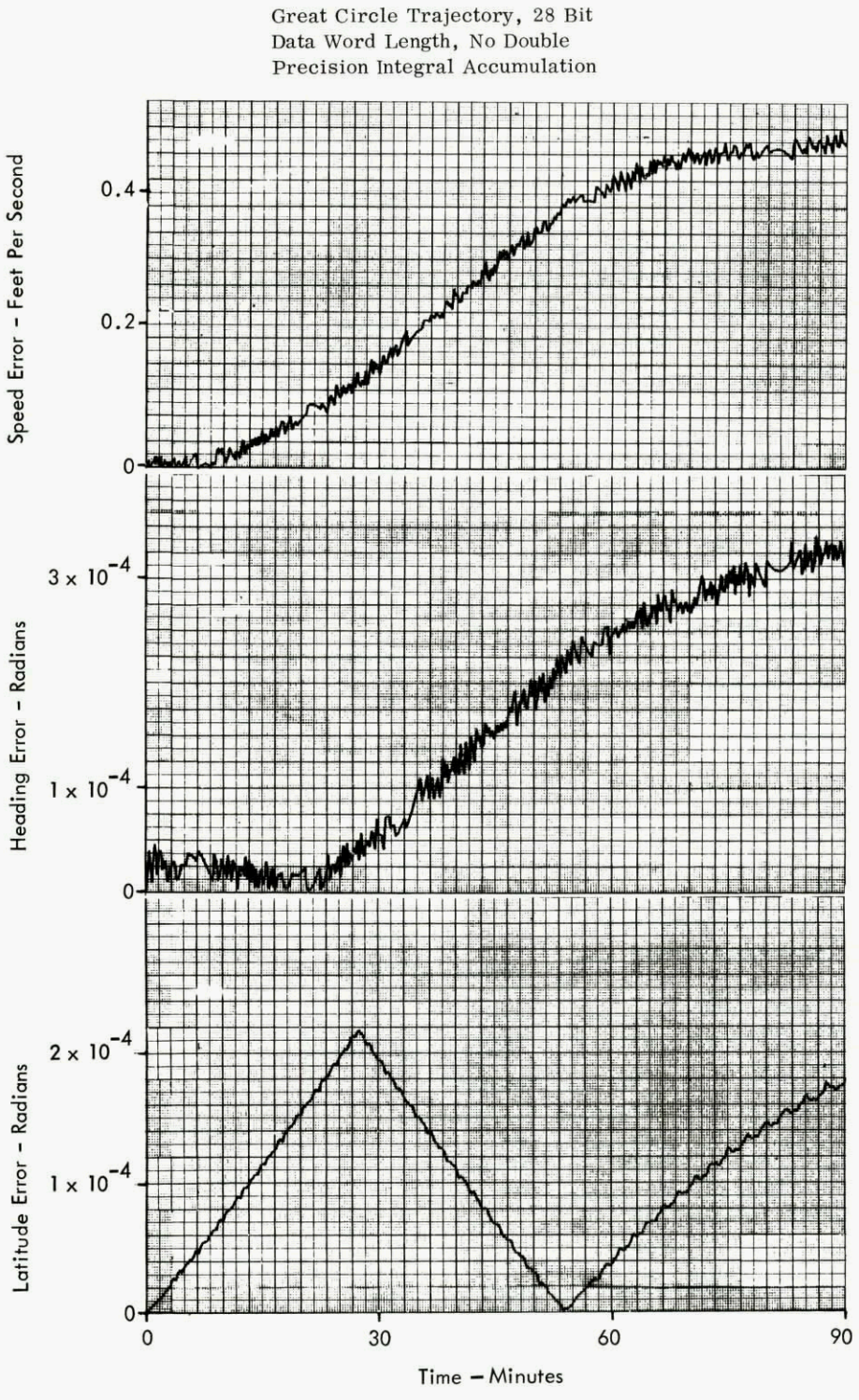


Figure 3. Errors From Computation in a North-Slaved Inertial Navigator

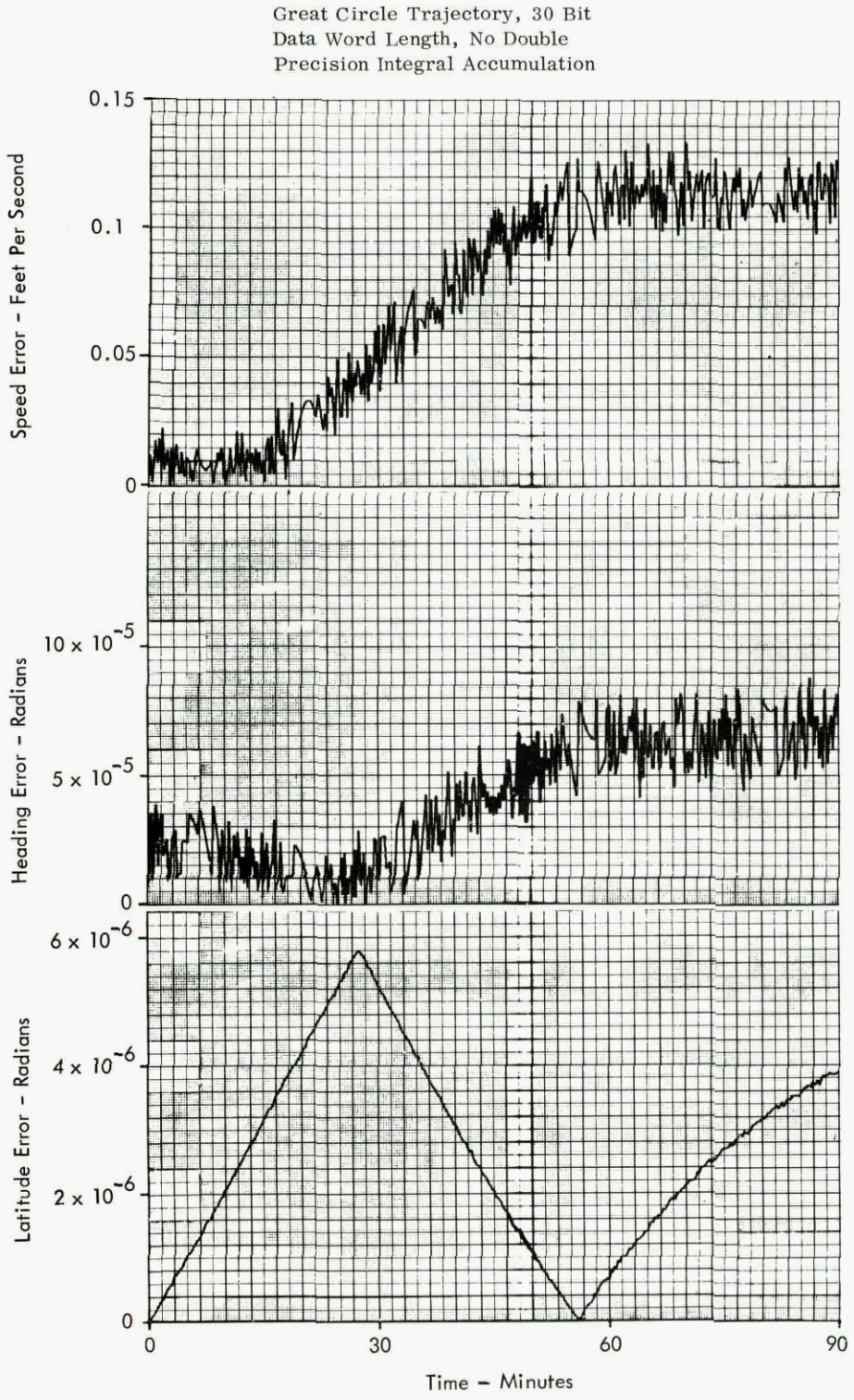


Figure 4. Errors From Computation in a North-Slaved Inertial Navigator

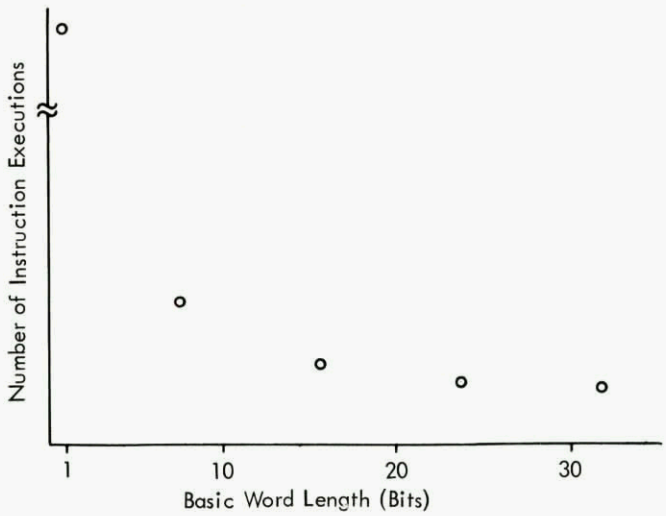


Figure 5. Number of Instruction Executions Required as a Function of Data Word Length (Aerospace Data Processing)

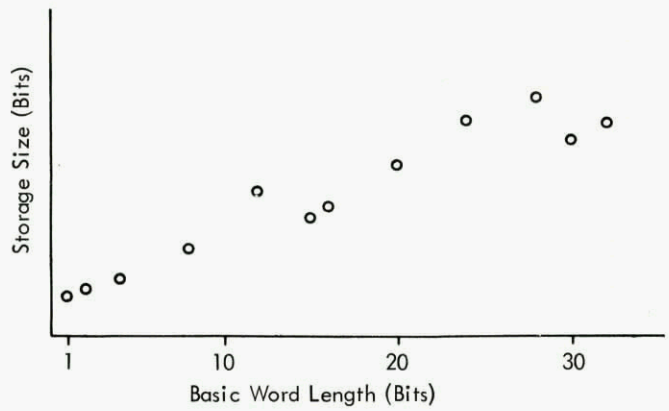


Figure 6. Storage Required for Data as a Function of Data Word Length (Aerospace Data Processing)

THE CASE FOR SPECIALISED SYSTEM PROCESSORS IN AIRBORNE INSTALLATIONS

by

P. A. Hearne

ELLIOTT FLIGHT AUTOMATION,
Airport Works,
Rochester, Kent.

SUMMARY

This paper examines the favourable changes made possible in avionic systems by the ready availability in the 1970's of low-cost, single or limited task processors. However, such availability has certain dangers and the need to optimise the avionics with regard to the functional requirements of particular systems, e.g. flight control, navigation, etc., rather than attempting to treat the avionics primarily as a computing system, is emphasised.

Also examined are the system requirements of airborne data processors in the areas of navigation, weapon delivery, flight control, engine control, air data and displays. From this examination, three different generic types of processor are derived.

One of these generic types, a small control processor, is examined in detail and the various analyses and trade-off studies carried out to determine its structure are summarised. The hardware implementation of this machine is then discussed.

Finally, a comparison between a 'distributed computer' avionics system using a number of these machines on a large, centralised multiprocessor system is made. From this comparison it is shown that the distributed system has a number of important advantages and is a more flexible and cost-effective solution for avionics systems.

THE CASE FOR SPECIALISED SYSTEM PROCESSORS IN AIRBORNE INSTALLATIONS

SECTION 1 – GENERAL.

The high rate of technological advance in digital techniques makes it unwise to be too dogmatic about absolutes in choice of system mechanisation, technology, order code and so on. Many of these are a matter of near religious conviction although it must be admitted that the religious fashion changes often abruptly. In the belief that we are now in a permissive era I would like to get down to some plain basic facts which relate the processors characteristics to the specific requirements of airborne systems and which relegate the computer from the role of an all demanding all worshipped god to an orderly well behaved moronic hand–servant to the aircraft designer.

Historically, airborne digital computers have been associated with the concept of a centralised installation in which a large number if not the entirety of the major computational processors required in the aircraft were performed in one computer or one computing complex.

In the early days of airborne computers some 12 – 14 years previously, the reason for this addiction towards a centralised complex was primarily an economic one. Early airborne computers using discrete semi–conductors or even vacuum tubes, reflected the then high cost of scientific and industrial machines. Consequently, if one particular problem for example inertial navigation, required the application of digital techniques and the use of a processor there was inevitably a tendency to apply the processor to as many different tasks as possible in order to get as much economic benefit as possible from what was undoubtedly a very expensive piece of equipment.

However, the concept of the centralised computer performing a multi–task role offended both practically and theoretically against the well tried concepts of ensuring aircraft safety and integrity by means of either similar or dissimilar redundancy. Many aircraft designers faced with the proposal for a single centralised processor for all major aircraft tasks quite rightly rejected it on flight safety grounds.

The proponents of the centralised large computer approach countered this rejection by the proposal for a multiplexed centralised computer complex in which the computer is still used in a multi–task role, but integrity is allegedly ensured by use of a multiplexed redundant computer and interface complex.

Such approaches seem to us at Elliott to fail to recognise that the purpose of the avionics in the aircraft is to serve the needs of the aircraft and its systems rather than in the case of the centralised computer complex to bias the aircraft and system requirements in favour of a data processing philosophy. It is impossible to over–emphasise this point. Many people nowadays, particularly the specialised system engineering groups refer to the concept of computing systems or the computer network as though the prime requirement was to obtain a completely rationalised and idealised data processing system.

I would stress as strongly as possible that the prime requirement is to achieve the most cost effective aircraft and its associated navigation, flight control, air data, display and other sub–systems. The data processing capability of the digital computer is merely a tool by which the specialist system designer be it for navigation, flight control or whatever achieves the computational requirements of his particular systems.

This necessary emphasis upon the requirements of the system negates many of the arguments for the centralised computer complex. As shown in Section 2 different systems have very different computational requirements some such as navigation requiring relaxed high precision working whereas others such as displays require relatively low precision at a high work rate. A centralised computer which would have to meet the most demanding accuracy as well as the most demanding speed requirements would in many instances over–kill the computational requirement for particular individual systems. This fact and others such as differing integrity requirements for different systems which are examined in more detail later, indicates the dangers of approaching the mechanisation of aircraft computational requirements from a data processing system viewpoint rather than from the viewpoint of the system requirements themselves.

Unfortunately, in the early stages at least airborne computer design groups tended to spring from engineering groups who had previously been associated with the design of scientific or other ground–borne machines rather than from designers of airborne equipment. This ground equipment ancestry has in some instances led to the apostolic attitude in relation to the data processing system as an end in itself rather than as one facet of the several means to that end. Furthermore, the initial scarcity of aeronautical equipment designers in the early airborne computer groups led to an emphasis on the design of the airborne computer as a computer with a large 'C' rather than as a system element which emphasis has in some instances continued until fairly recent times.

Fortunately, today the widespread availability at low cost of digital components has enabled a much more realistic view to be taken of the design of airborne processors. First and foremost it has proved possible to build individual digital single task processor units which are directly cost and weight compatible with their individual analogue counterparts. Thus there is no longer the necessity to inject multi-tasks into a computer to justify its cost which is now directly comparable with the analogue computation element which it replaces.

Furthermore, the increasingly widespread availability of such techniques as computer aided design and simulation of integrated circuits, logic and stores has enabled a lot of mystique to be removed from the design of digital processors. It is now quite feasible to consider the design of specialised processors in accordance with the particular requirements of individual systems rather than the steam age approach to trying to fit one processor to many different applications. Of course, in a situation in which a number of individual single task computers replace the single or multiplexed central computer, the smaller computers, whilst not identical would have many familiar similarities in terms of the type of micro-circuit and constructional standards employed. There will also be some form of resemblance, though not necessarily compatibility in their order code. As shown later airborne computers which may be produced in large numbers for one single defined task differ greatly in the hardware and software trade-off from ground-borne computers which must be capable of being readily programmed for a wide variety of different applications for which they are usually supplied in relatively small numbers.

This note, therefore, examines firstly the requirements of the various avionic systems and secondly one of the generic type of these new specialised digital processors which tend to meet these requirements. Some indication is also given of the relative cost and system effectiveness of the specialised system processor approach. Note though that the specialised system processor will probably still be a general performance machine rather than the centralised computer approach.

SECTION 2 – AVIONICS SUB-SYSTEM REQUIREMENTS

Considering first of all the general requirements for any avionic system we have in general to specify requirements of accuracy and resolution, speed or bandwidth, reliability and integrity and in addition we must decide upon the most economical way in which these requirements can be met. In addition, to these basic requirements, with a digital system in particular, we must examine carefully the interfacing requirements and the programming task involved in establishing the software for the computer.

Considering the main system blocks in an advanced military aircraft we find the following.

2.1 Navigation

Navigation sub-system requirements for single task processors are usually related to the provision of individual special processors for individual sensors requiring extensive computation. The best known example is, of course, the inertial navigation requirement where the accuracy of digital computing is required to maintain the accuracy of open loop integration and the various processor terms.

Two choices of processor are available. When the processor is intimately involved with the inner platform control loops including the actual torquing pulses for gyro and accelerometers there is a high premium on speed coupled with an 18 to 24 bit word length requirement for accuracy. Storage requirements are also increased somewhat. On the credit side there is a reduction in the intimate platform electronics and an overall simplification of the system. The other choice is to have a separate and relatively self-contained intimate platform electronic unit which will probably also house the analogue digital convertor and the appreciably slower computer which has the same word length accuracy requirements and a slightly lower storage requirement. Although this gives a slightly higher system cost it has the advantage of allowing independent operation of the platform as an attitude reference in the event of computer failure and it also can offer simplification of the navigation processor. In each case this processor calculates present position from accelerometer and heading inputs, provides the necessary platform precession terms, carries out the necessary system mode management and switching sequences, executes the gyro compassing phase and provides overall sub-system self-check. Additional modes which may often be included are waypoint and target storage and associated steering calculations and sometimes Kalman Filtering mechanisation. Generally it seems cost effective to put steering and waypoint storage in the special navigation processor but to reserve Kalman Filtering for the overall main system integration computer.

The inertial navigation processor with the desirable characteristics would have an 18 to 24 bit word length and would have between 4,000 and 6,000 words of storage depending upon the elaborations or modes and outputs over and beyond the basic present position and steering modes which can usually be contained in a basic 4096 word block.

Note, however, that read only memories rather dilute the importance of binary power intervals of store capacity.

Speed requirements vary according to the system mechanisation chosen. If a parallel computer is used including the intimate platform functions a one micro–second store access time with better than three micro–second add time seems necessary.

The less demanding speed requirement can in fact be met by a serial computer with much slower speeds. If, however, a parallel computer were used in the interests of commonality the store access time would be in the six to ten micro–second region since reduction below this figure would not give any further cost advantages at 1970 technology.

A processor of this type is very suitable for such complex radio navigation aids as Omega involving timing and hyperbolic transformations. The lower level of complexity processor, probably serial, is also directly applicable to Doppler VOR area nav. TACAN and similar radio navigation aids.

The correlation, compounding, filtering etc, of the various individual navigation sensors is carried out by the main systems integration computer referred to below which receives processed data from the processors of the various sensors. This computer also acts as a central storehouse and data distribution centre for all of the processed data from the various sensors. However, if the main computer should fail the crew still have directly available to them the individual processed data from individual sensors albeit in an uncombined or unfiltered form.

Because of this variety of navigational information normally available in a modern aircraft failure of a part of the navigation sub–system does not, in general, have any immediate effect of the safety of the aeroplane so that it is not necessary to have complete redundancy in the sensor/processors sub–systems for navigation. Such redundancy as is required is given by the alternative navigation sensors available and by increasing the crew workload in the event of a failure in the main computer system.

2.2 Weapon Delivery

Turning now to weapon delivery for military aircraft this sub–system requires solution of the ballistic equations for a wide variety of different weapons and computation of the release point based upon the range information from such sensors as radar, laser etc.

The accuracy requirements in general are not as great as navigation due to the more limited range involved in the weapon delivery computations so that satisfactory accuracy can be achieved with shorter word length. In general weapon delivery accuracy can be achieved with 12 – 16 bit word lengths albeit with some double length working.

The speed requirements, however, are very much greater and whereas with navigation an iteration rate of about five times per second is generally adequate, the iteration rate for weapon delivery increases to about 40 – 50 times per second.

The integrity requirements on the overall weapon delivery system are, of course, stringent since the possibility of accidental weapon release must be minimised. This integrity is normally achieved by having safety interlocks so that the system is only operational when armed by positive crew action immediately prior to an attack.

2.3 Flight Control

The requirements of flight control systems are covered in detail in another paper. In general, however, the accuracy required is limited to approximately 12 bits. For the outer loop control functions the speed requirements are moderate (approximately 20 – 30 iterations per second is the highest rate required) whilst for inner loop control, extremely high iteration rates are required (possibly even up to several hundred iterations per second). The integrity requirements vary depending upon the part of the flight control system concerned. It may be satisfactory to have a single lane for some of the outer loop control functions whilst the stability auggmentation and any terrain following systems require multi–lane fail safe or fail operative redundant solutions. The Flight Control system requirements are thus complex, and must be considered in detail. This is done in Mr. R.W. Howard's paper given at this meeting to which I refer you.

2.4 Engine Control

Engine Control has broadly the same problems as flight control with similar accuracies and similar integrity requirements. When used in an outer loop mode or organisational control the computer tends to require lower speed but larger store capacity and when used in an inner loop control such as for re–heat or acceleration control the speed requirement is increased and storage requirement reduced somewhat.

2.5 Air Data

Air Data computation calls for the solution of complex functions at reasonably high iteration rates at between 20 and 50 per second. The accuracy requirements again are of the order of 12 bits since

once again this is the accuracy with which we are able to extract the basic sensor information and it is obviously not worthwhile computing to any higher degree of accuracy than that sensor information.

The integrity requirements of air data are again complex. For display purposes the integrity is usually provided by having standby instruments and having, within the basic flight instruments normally driven by the air data computer, alternative means of driving that instrument. Redundancy may, however, be required to satisfy stability augmentation requirements.

2.6 Electronic Displays

The computing requirements for displays again usually require high–speed computation, for example on a Head–Up Display 50 iterations per second are required to avoid flicker on the cathode ray tube, and as such displays are more and more becoming the primary flying instruments in the aircraft the amount of data required is increasing. Accuracy again is equivalent to a 12 bit word length, the accuracy being dictated in this case by the accuracy with which the data can be positioned on the face of a cathode ray tube. Integrity once again is usually satisfied by having alternative display configurations rather than by redundancy of the basic display.

2.7 Main Computer

Finally, I must consider the computer for the function which is generally known in today's circles as the main computer, formerly known as the central computer, when the computer tasks envisaged did not extend into flight control and other areas. This computer generally performs what is known as the Nav. Attack function and its main purpose is to integrate the outputs of the other sub–systems such as navigation and displays in such a way that large amounts of data can be transferred, correlated, compared etc, between the various sub–systems within the aircraft. The function of this Nav. Attack computer, whether in an aircraft with primarily analogue or primarily digital computation for the sensors and control functions, is to provide the centralised source of the best possible information derived from these different sub–systems to be used in the navigation and attack functions. Thus, for example, the Nav. Attack computer would produce the ground stabilisation and pointing signals for the Low Light Television and Laser Ranger which it would in turn derive from the inertial navigation and possible tracking radar sub–systems. Similarly, the Nav. Attack computer would probably organise the display format mode change which must necessarily take place when moving from the en route phase to the attack phase of the mission and would also generate much of the data in terms of navigation, position, weapon state etc, which would be passed to the display processors for generation as display formats on the different surfaces.

This type of integration and management task requires computer word lengths of the order of 18 – 24 bits with a fairly large store and, because of the timing requirements associated with a multi–task type function in this instance, the speed of the computer is more comparable with that of the control processor than the navigation computer previously considered.

It is important to note that although this larger main computer is carrying out an integration task, failure of the computer does not affect the integrity of the aeroplane since all of the information which it uses is still available in a semi–processed form from the other sub–systems – navigation, air data etc, in the aircraft. The crew are, therefore, still able to use this semi–processed information, albeit with a higher degree of crew workload even though the integration and correlation function on the main computer may be lost due to computer failure. This gives a welcome form of dissimilar redundancy which may be further improved in later systems by a degree of task sharing between the different processors using the serial data transmission systems which nowadays is a common feature of advanced military aircraft.

We have so far considered relatively slow, high–precision computers for navigation purposes, high–speed and lower precision computers for weapon delivery, flight control and air data, and displays where the accuracy in a computation is necessarily limited by the requisite accuracy of the inputs and outputs and the larger main computer for overall system integration and management where both high accuracy and high–speed working is necessary to cater for its multi–task role. The problem of attempting to meet all these requirements in an efficient manner in one or more large centralised computers is obviously an extreme one. Instead, rationalising these needs to basic aircraft functional requirements leads to three major types of data processor which could be used in future aircraft systems as shown in Figure 1. These are characterised as Management, Control and Sensor processors.

The Management processor is essentially a large, fast, powerful computer useful for carrying out combined navigation/attack/management computing, or as a tactical processor.

The Control processors are smaller, less accurate machines and include as examples small inertial navigation processors, display processors, flight and engine control processors and weapon delivery processors.

Sensor processors include air data processors, fuel measurement system processors etc.

I would now like to concentrate upon the control processor application to indicate the

machine thinking behind one of these types of specialised single task processors.

Figure 1

FUNCTIONAL REQUIREMENTS

Parameter	Management Processor	Control Processor	Sensor Processor
Store Capacity	8K - 64K words	2K - 8K	< 4K
Store Type	Variable	Fixed or Variable	Fixed + data store
Data Word Length (bits)	18 - 24	12 - 18	12
Add time	< 3μsec	< 3μsec	Variable may be either serial or parallel computing
Multiply time	< 9μsec	< 9μsec	
Direct store Access Requirements	To whole store from large number of input channels	To restricted. Section of store from limited number of input channels	To data store only

SECTION 3 – DESIGN REQUIREMENTS FOR A CONTROL PROCESSOR

The basic requirements for an airborne processor can be shown in Figure 2.

Figure 2

OPERATIONAL REQUIREMENTS

- FUNCTIONAL – Speed, capacity, technical suitability for task
- ENVIRONMENT – Suitability for airborne role
- RELIABILITY & MAINTAINABILITY – Overall equipment availability, testability, ease of repair
- INSTALLATION – Size, weight, cooling requirements, power requirements
- COST – Minimum first cost and cost of ownership

Considering first of all the functional requirements, the speed of the machine is obviously of importance and the order of speeds required can be illustrated by the following simple example.

If we assume that we have a programme length of about 4,000 words which is fairly typical for a control processor task and this programme must be cycled at 50 times per second, this means that in every second 200,000 instructions must be obeyed. Making the further assumption that 70% of the instructions are additions or the equivalent to the additions in terms of time and the remaining 30% are multiplications or their equivalent and that a multiply takes four times as long as an addition then the resulting addition time is 2·6 micro–seconds and the multiply time is 11 micro–seconds. This is a relatively trivial example and, in fact, much more detailed analyses are carried out to establish the requirement. For the purposes of this example, however, the numbers generated are suitable.

This speed implies a fairly fast computer. Analysis of the word length requirements indicates that in general with the types of sensors available a word length of 12 bits is suitable for accuracy and resolution requirements. This is not to say that 12 bit words are necessarily the most suitable for instructions. The instruction requirements will be discussed later.

The remaining requirements are largely self–explanatory. Obviously, the equipment must operate in the aircraft environment and the reliability and maintainability requirements have to be met.

These requirements are, in fact, becoming far more stringent and a great deal of attention must be paid to achieving them during the design phase.

It goes without saying that we are extremely concerned with the size and weight and cost of the basic computer.

SECTION 4 – CHOICE OF INSTRUCTION SET

Having decided upon the basic requirements of the machine it is necessary to examine the instruction sets required. Since this choice is an important factor in determining the cost of the complete machine very detailed trade–off studies must be carried out.

One of the first trade–off studies which has to be done is a trade–off between hardware and software. Basically, if the processor has an extremely powerful order code with higher level languages and other features programming becomes easier, but more hardware is required to implement that order code. If we consider for example a general purpose computer for scientific usage with a wide variety of different tasks during the lifetime of the computer then software is an extremely important factor in the total cost of ownership of that machine. It is thus extremely important to be able to programme the machine in an economical manner. With an airborne computer, however, used in a large number of identical systems in the production aircraft series the programming need only be carried out once during the system development phase. For airborne computers, therefore, there are appreciable cost savings by economising on hardware and order code complexity and using skilled system engineer/programmers for the one time non–recurring programming task. This is shown in Figure 3 in which the system unit cost for large numbers of identical systems greatly benefits from the reduced amount of hardware required even though initial programming was more costly.

Because of this software costs are much less important than is normally the case and a different mix of hardware and software is required. Figure 4. With the increased hardware percentage content of the total cost of ownership we are concerned with reducing the amount of actual hardware involved and to do so it is necessary to restrict the instruction set.

Considering the instruction set in some detail, an examination of a typical computer instruction would indicate that this is broken down into three parts. Figure 5. First of all we have a function code which is typically 4, 5 or 6 bits depending upon the extent of that code. The second part is the mode of operation and can consist of 1, 2 or 3 bits.

The final part is the address or displacement field and this can be anything from 7 bits upwards depending upon the type of addressing used.

Figure 3
HARDWARE v SOFTWARE

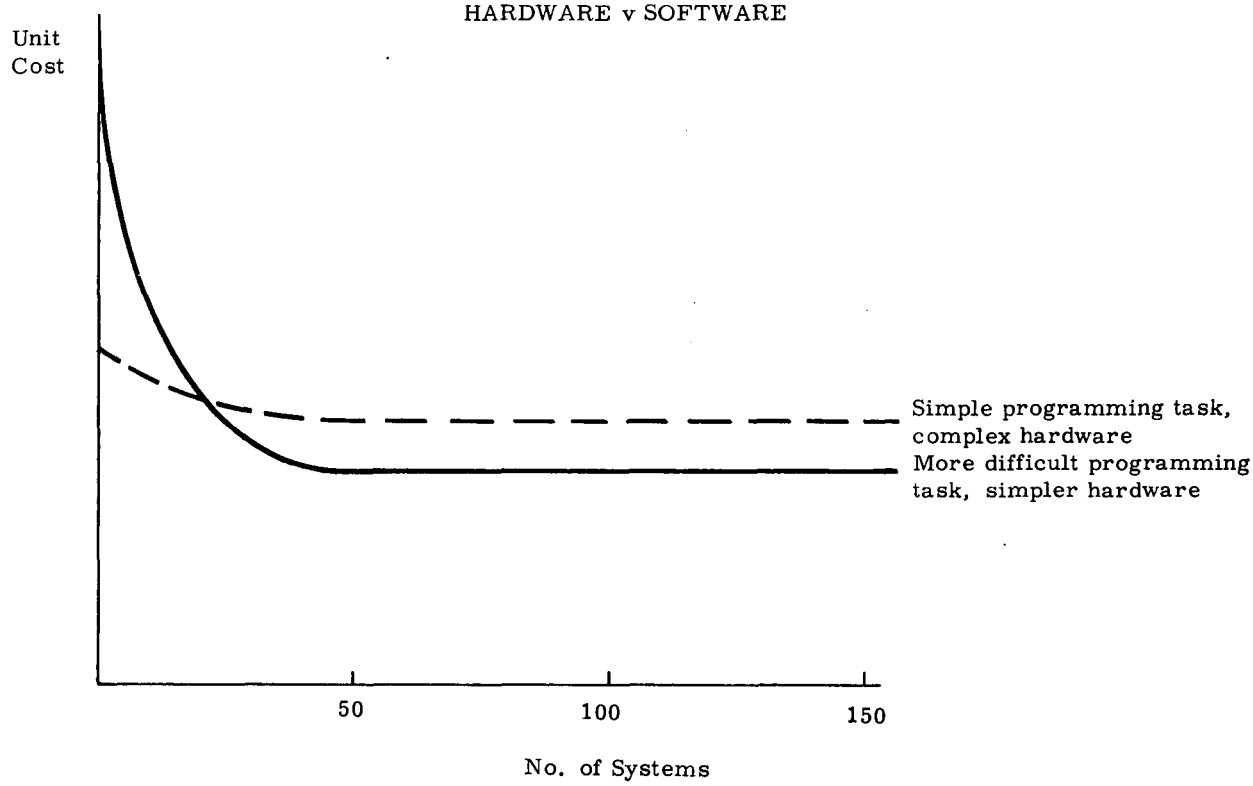


Figure 4

HARDWARE v SOFTWARE COMPARISON

Cost of Ownership Breakdown

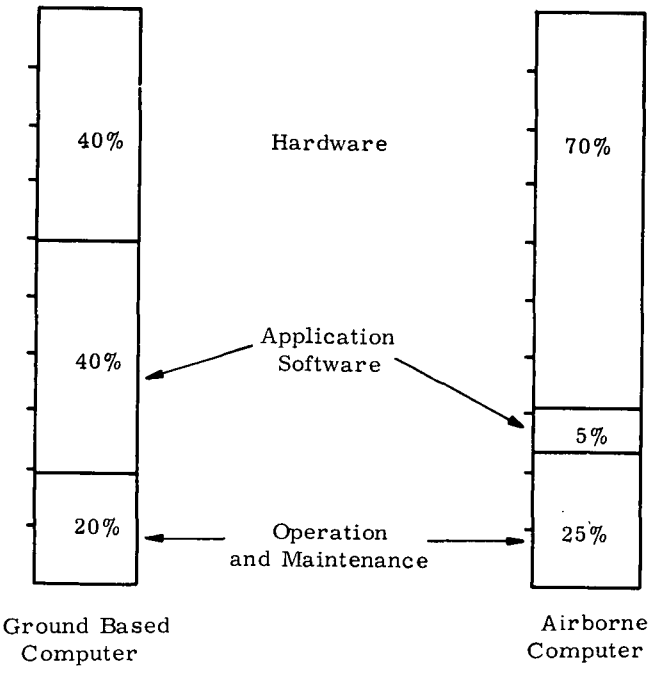


Figure 5

BASIC WORD FORMAT

F = Function	4,5 or 6 bits
M = Mode	1,2 or 3 bits
D = Address	7 or 8 bits for relative addressing or displacement 13 to 16 bits for direct addressing field

Minimum word length is 12 bits

ie 4 bit function
1 bit mode
7 bit displacement

Therefore 12 bit instructions should be used to give minimum hardware provided the various parts (ie FMD) of the instruction are sufficiently flexible for airborne use.

An examination of the minimum number of bits required for each of the three parts gives a word length of 12 bits and since as I have already indicated we are concerned with minimisation of hardware this is the word length which should be used provided it is sufficiently flexible to carry out the tasks envisaged.

Considering first of all the 4 bit instruction code this gives us a minimum of 16 instructions and this can be expanded by the use of the other parts of the instruction word to give additional 'address-less' instructions.

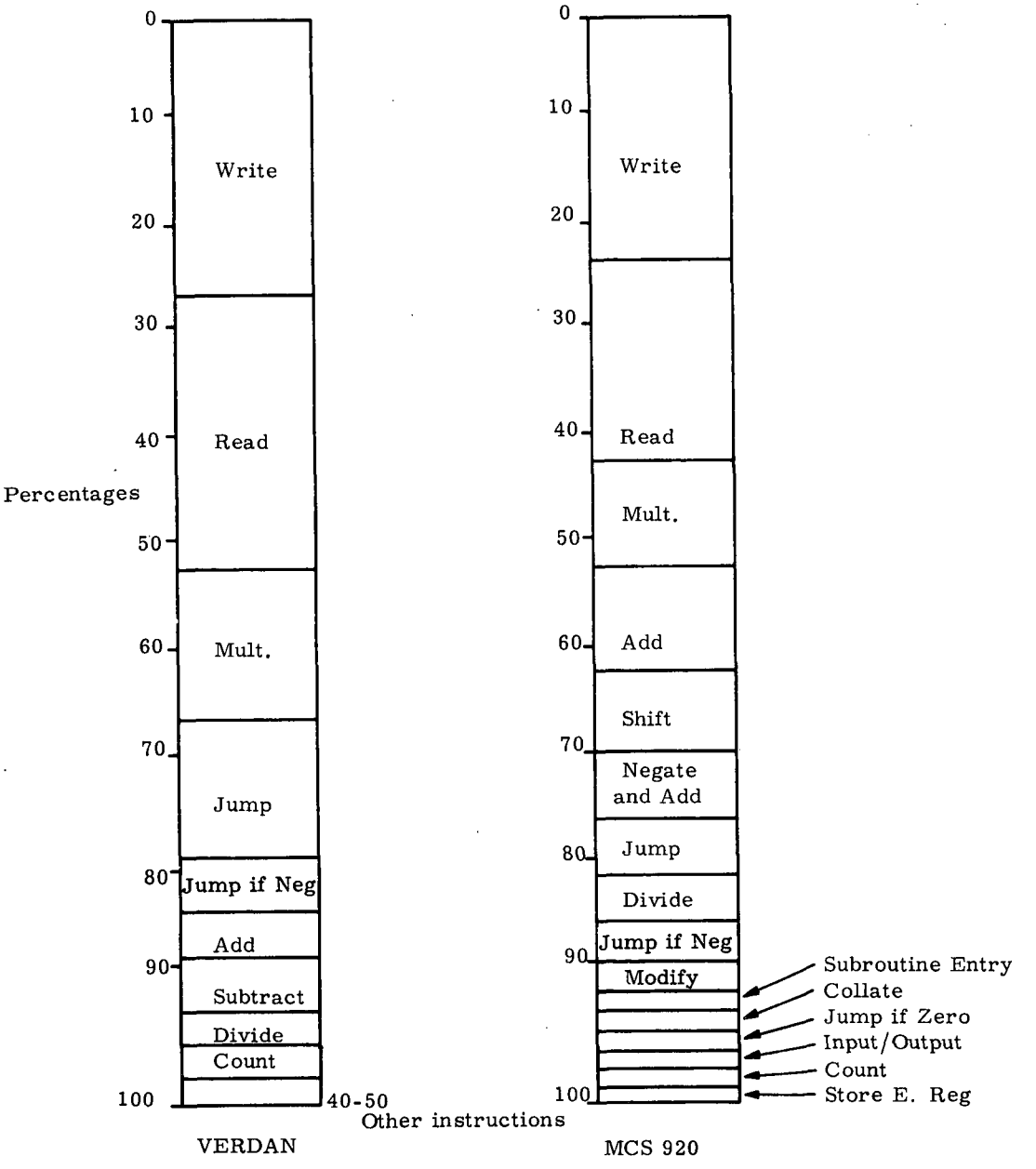
Analysis of a wide variety of airborne programmes for various computers indicates that the instructions used in Figure 6 are the most widely used and that any additional instruction would be used less than 1% of the time. This would indicate that for airborne programmes a 4 bit instruction code is sufficiently flexible.

Examining now the mode section of the word, one of the chief uses of this is to provide multi-accumulator and index register capability. Again an examination of the utilisation of instructions

Figure 6

CHOICE OF ORDER CODE - 4

Utilization



in airborne programmes indicates that a large proportion of the time is taken up by load and store instructions. This is, in fact, typical of single accumulator machines and the method of reducing it is to go to multi-accumulator operation.

However, a detailed analysis of the control processor tasks for this type of computer indicates that only approximately 15% of storage locations are saved by having multi-accumulator operation. Since multi-accumulator operation would require the word length and hence the storage cost to be increased by more than 15%, it does not seem worthwhile for these special purpose applications and a single accumulator system is, therefore, chosen.

The final part of the instruction word is the address field and with a seven bit field a page of 128 words can be addressed. Using a mode bit enables two separate 128 word field to be addressed.

Each additional bit added enables the size of the address page to be doubled so that for example with a 13 bit address 8,192 words are available on the page.

To utilise the complete storage capability of the machine a pointer register indicating the page address must be used and this must be set to a new value every time the page changes.

To decide upon the effectiveness of a 128 word page a detailed analysis of airborne programmes using this size of address field in comparison with the same programmes written for a machine capable of directly addressing 8,192 words of store, it was found that the programme size increased by only a few percent and thus having a restricted field did not seriously affect the flexibility and economy of the machine.

Because of this it becomes apparent that a 12 bit instruction word is sufficiently flexible for the order code for the applications under consideration. This consideration together with the inherent limitations on input data accuracy and the lower accuracy requirements of closed loop systems justifies the choice of 12 bit word length for minimum hardware in low cost specialised control processors.

SECTION 5 – CHOICE OF HARDWARE CONFIGURATION

Having decided on the architecture of the machine, it is necessary to consider its hardware structure. The continuing reduction in logic cost has made the storage media the critical hardware element in the computer.

The most widely used storage media is, of course, ferrite cores. These have been with us for a long time now and are a safe solution to the storage problem. They have, of course, some disadvantages; it is difficult to operate them over the temperature range encountered in an aircraft environment and the price of such a store is high in proportion to the rest of the machine.

Bulk integrated circuit random access stores are now becoming practical as a result of the continued development of M.O.S. technology and such stores are now a serious contender for future applications. The great advantage offered by these devices is, of course, a potential significant price reduction as the learning curve of manufacturing such devices progresses.

Against this, it must be remembered that integrated circuit stores, by their very nature, are volatile and thus unless precautions are taken a power supply interrupt can cause the loss of the programme. These precautions can take the form of either a sustaining standby power supply or an airborne programme loader.

In essence, however, for most airborne applications, only very limited programme changes would be expected during the life of the aeroplane. This, coupled with store integrity requirements, makes it generally preferable to consider fixed storage techniques. In the past such stores have been rope core stores, and now M.O.S. read only memories are the preferred storage medium. Such stores, however, only answer part of the storage requirement and in integrated circuit scratch pad store is also necessary. This can be either bipolar or M.O.S., depending upon the application. To this end it is now our practice to split up the storage requirements into programme store and data store, normally utilising the two types of technology mentioned above. This gives us an immediate increase in flexibility of design and a much greater capability for processor design optimisation to meet a specific requirement.

The introduction of integrated circuit elements and latterly Medium Scale and Large Scale Integration elements has greatly reduced the size and weight of the logic section of the computer. Currently, the best technology to use in the logic section appears to be the use of standard production M.S.I. elements with special hybrid packaging techniques, using standard chip elements, when size penalties might otherwise occur. Because of the relatively small size of the logic section the use of M.S.I. as opposed to the more glamourous L.S.I. technique has very little effect on the overall size and weight of the processor. It does, however, give an appreciable saving in cost compared with the L.S.I. elements particularly when 'discretionary wiring' techniques are used.

Circuit elements are assembled on multilayer boards with multilayer mother board interconnections. Particular care must be given to thermal design and elaborate computer modelling should be used to eliminate high operating temperatures on components and to ensure high reliability design and operation. Figure 7.

Figure 8 shows the achievable targets using 1970 technology for the control and management processors of the type discussed here. Note that the weights and sizes include some interface and also the power supplies. I am sure I am not alone in bemoaning the fact that creative engineers prefer to design advanced technology items and that power supply design being an unglamourous backwater is finding it difficult to attract sufficient talent to keep it in step in size and weight with the startling reductions made in the processor proper.

Figure 7

DISPLAY PROCESSOR SHOWING HEAT SINKING ARRANGEMENTS ON A COMPUTER CARD

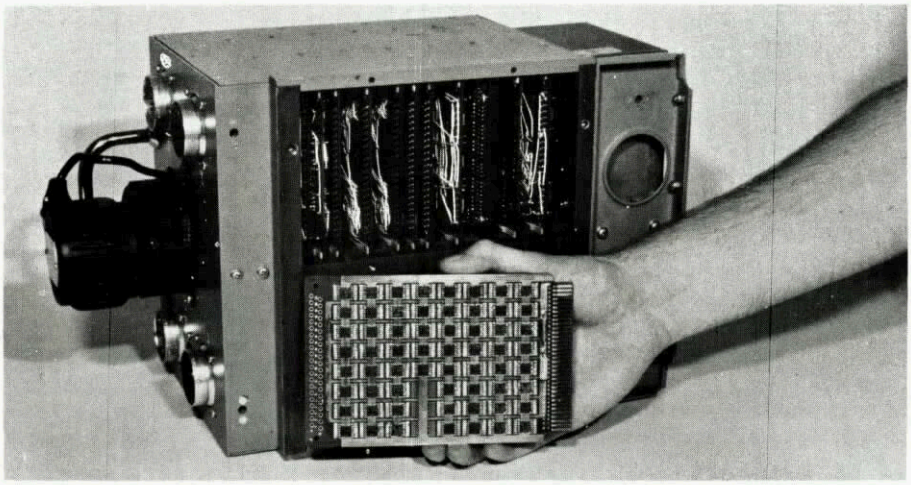


Figure 8

SIZE & WEIGHT

	Management Processor	Control Processor*	Sensor Processor*
Weight including Power Supply store and essential interface	(16K store version) 25 - 30 lb	15 lb	10 - 15 lb
Size	1 ATR short	1/2 ATR short	1/4 - 1/2 ATR short

* NOTE: These sizes and weights are for the processors packaged as L. R. U. s. The design is such that they may be packaged as part of a larger L. R. U.

SECTION 6 - SPECIALISED VERSUS CENTRALISED PROCESSORS

I would like to conclude this paper by considering a case study of an aircraft system using a centralised multi-computer system compared with the same aircraft using distributed specialised processors.

Consider the systems shown in Figures 9 and 10.

Figure 9 shows a hypothetical military aircraft system employing two large management digital computers which are responsible for performing all of the navigation, fuel management, autopilot, air data and weapon delivery functions.

In this system there are 2–8,000 word 18 bit, 1 micro–second cycle time computers together with a display computer of the type previously described, the reasons for which will become obvious in a moment. There are also two quite elaborate interface units associated with these computers.

Figure 10 shows a distributed computing system in which there is only one main computer which works in association with distributed computers performing the navigation, autopilot, air data and display computer functions.

In this system the main computer carries out weapon delivery and fuel management tasks and provides the overall command and control functions for the remainder of the system when the pilot does not wish to exercise individual human control of each computer element.

Figure 9

CENTRALISED COMPUTING SYSTEM

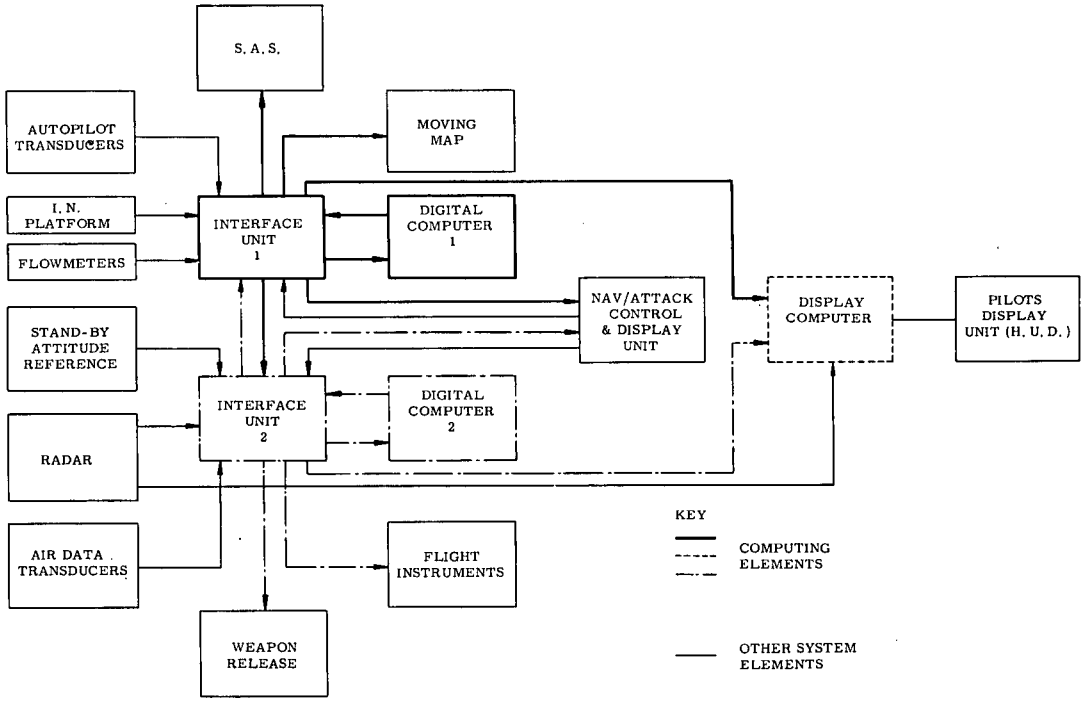
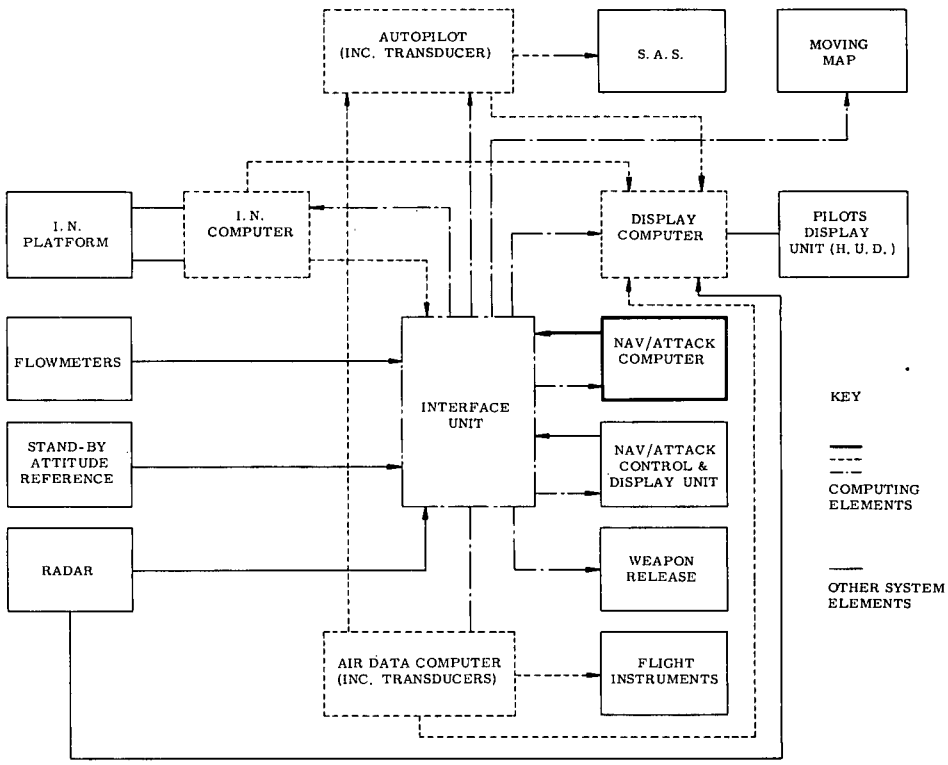


Figure 10

DISTRIBUTED COMPUTING SYSTEM



Comparison between the two systems evokes the following points:

i) Programming

Anybody who has been concerned with the commissioning and operation of advanced digital systems will appreciate the problems involved when a large number of tasks are combined in a single digital computer programme. Changes in individual tasks often demand a complete re–packing of the programme and when changes are introduced sometime after the original programme has been written, the original personnel may not always be available and a considerable hiatus may occur. One or two air defence systems have shown evidence of this.

On the other hand, if the programming load can be distributed between individual computers, the programme is both easier to write and understand since it relates to one task only and it becomes a much simpler matter to carry out the initial programming and maintain adequate software control throughout the life of the project.

Although this may seem a small matter the success of many digital systems is dependent upon the intelligence and understanding of the associated software and anything which can be done to simplify this brings considerable benefits in both the development and operational phases.

ii) Iteration Rate

Although the larger computers can store considerably larger amounts of information in terms of data and tasks to be performed it remains a fact of life or computer engineering that the speed at which they can perform these individual tasks is no faster than the speed of the smaller computers.

Therefore, if the large central computer is performing four tasks say, and a smaller computer performing one, the larger computer can only produce an iteration rate or frequency of solution, 1/4 that of the smaller computer.

Since many of the tasks which we are now considering suitable for digital control such as the Automatic Flight Control System, are dependent upon a high solution rate for stability, e.g.in the terrain following mode, it follows that the central computer can only be used for these high band–width tasks provided it is relieved of its other task functions.

In the dual central computer system illustrated it is obvious that the computing require–ments of the aircraft demand the full time usage of both computers and it is not generally possible to consider a condition where one computer is unloaded merely because it has to concentrate on a high band–width task.

On the other hand, if individual small computers are performing the different high band–width tasks they can allocate the whole of their computing cycle time to their particular function and this very real limitation is removed. The most demanding areas for computing time are the display generation and the autopilot control tasks, and it is noteworthy that in the system considered even the multiplex central computer was unable to perform display calculations at an adequate rate for the system under consideration, and the use of a special additional computer was necessary even though the storage capacity of the central computer was adequate.

In the systems considered on a purely cycle time basis, the dual central computer system was unable to perform anything other than simple heading and height holding autopilot functions and very limited air data information capability, as opposed to the distributed system in which full autopilot functions including terrain following the full air data capability were achieved.

It is interesting to note that recent simulation and analytical work on digital control systems has shown the need for far higher iteration rates for inner and outer loop stability than were at first envisaged and indicate the importance of computing speeds in this particular application.

iii) Weight and Volume

The weight of the computer and interface elements of the centralised computer system are 183 lb. compared with 174 lb.for the distributed system and the volume of the centralised system is 4·6 cu.ft. as opposed to 3·9 cu.ft. in the distributed system.

On this basis neither system has any clear advantage over the other.

iv) System Integrity

The function failure rate, i.e. the mean time between complete failure of a particular function is approximately twice as good on the distributed computing system because the individual distributed computers being less complex have an appreciably higher mean time between failure.

There is, therefore, a significant gain in both the operational capability and the maintenance requirements in favour of the distributed computing system.

v) Cost

The first cost of the computer and interface elements of the centralised computing system is £63,000 compared with the £49,000 of the distributed computing system. There is, therefore, a 30% cost advantage in favour of the distributed computing system and this is also reflected when account is taken of the spares and operational maintenance cost of the system in service. This is probably the single greatest advantage of the distributed system.

Figure 11 summarises the advantages of the distributed as opposed to the centralised computing system. Although the absolute ratios of one system over the other may be queried, I believe there is sufficient indication from the studies carried out to date, which I have not time to produce in detail here, to indicate that at this moment of time in development of airborne digital systems, distributed computing systems offer much more advantageous methods of applying digital control in aircraft than the hitherto favoured centralised system.

Figure 11

CENTRALISED v DISTRIBUTED COMPUTING SYSTEMS

	Centralised	Distributed
Programming	Demanding	Relaxed
Iteration Rate	Restricted	Satisfactory
Weight	183 lb	174 lb
Volume	4·6 cu.ft.	3·9 cu.ft.
Integrity	1·0	2·0
Cost	1·0	0·7

SECTION 7 - ACKNOWLEDGEMENTS

The author wishes to acknowledge the help of the many members of EFA staff who have over a number of years produced the concept discussed in this paper.

ECONOMIES OF INTERFACING WITH SMALL SENSOR COMPUTERS

by

K. R. Brown

Inertial Systems Department
Ferranti Limited
Edinburgh, UK

SUMMARY

With the newer generation of digital processors, it is increasingly obvious that the interface between computer and sensor is the major part of the complete system. This is particularly true in the aircraft navigation field where the trend is to self-contained systems which include a small digital processor - the sensor computer. This must interface with the input sensors and feed outputs to the other aircraft systems by digital data links and by various analogue transmissions.

To make such self-contained systems economical, it is necessary to restrict the size and cost of the interface coupling the computer to the sensors and to the remaining systems. Certainly in the case of avionics equipment an economical solution is the one which leads to a reduction in the material content and the overall size of the equipment. Certain techniques which reduce the cost of storage within the interface and permit the sharing of expensive high accuracy parts of the circuitry are described. These also reduce the bulk of the equipment.

Both output and input of data to the computer must be considered; the data can be of many forms. The data in and out of sensor computers is frequently part of a closed loop control system, in which case very careful attention to the method of signal flow is required to reduce the effects of the control system dynamics on the computer load.

Certain basic design principles and rules are stated in the paper, and the use of these rules is illustrated by two examples - the interface with an inertial platform and a synchro/resolver interface unit.

1. INTRODUCTION

With the newer generation of digital processors, it has become clear that the interface between computer and sensor is now a major part of the complete system. This is particularly true in the aircraft avionics field where the trend is to self-contained systems which include a small digital computer - the sensor computer. This must interface with the input sensors and feed outputs to the other aircraft systems by digital data links and by various analogue transmissions.

To satisfy the need of the commercial market, there are now available very complex logical elements in integrated circuit form, that is the so-called Large and Medium Scale Integration. This has made possible the low cost general-purpose digital computer, and an excellent case can be made for the use of such small computers built in as an integral part of avionics equipment. Such a computer, without program, can be built on a printed circuit card area of 100 sq. in. Amongst other advantages, the economics of using such a small computer are very attractive. The economies possible are not only in direct cost of components. The packaging of equipment is expensive, and hence any reduction in volume is very relevant to the reduction of cost, whilst the saving in weight has a direct bearing on aircraft operating cost and effectiveness. Using high density packing techniques, the sizes quoted can be reduced by a factor of two with relative ease, and there are further reductions possible in the immediate future. Such a computer has sufficient power to service most of the avionic sub-systems within an aircraft. The capacity needed for a selection of typical avionic systems is shown below.

TASK	CAPACITY WORDS	
	PROGRAM	WORKING STORE
Basic Inertial Navigator	1,000	48
IN with Navigation Computation	2,000	64
Air Data Computer	1,500	32
Moving Map, Drive	600	32
Moving Map, Chart Control	1,600	64

COMPUTER SIZE FOR VARIOUS TASKS

The normal task of this type of computer is to serve a group of sensing instruments or sensors, and as such it will be referred to as a sensor computer, that is a computer which deals with a group of sensors, processing data from them, and transmitting data to display sensors as required.

Input sensors are generally analogue instruments, whilst many outputs are still required in analogue form.

The advantages of integrating a computer into the equipment are great. The computer is closely associated with the sensors it serves and the electronics connecting it to the sensors, for example an inertial platform with gyroscopes and accelerometers, or an air data system with pressure transducers and temperature probes. This leads to much simpler maintenance and testing, as a complete system function is contained within one box, compared to the older assemblages of separate boxes providing the different parts of a particular sub-system. The very real danger of faults in the box interconnections can be avoided and, of course, the interface with the rest of the aircraft system becomes much simpler.

The demands of a very large commercial market have made it possible to reduce the computer until it is a very small part of the overall size of the equipment. Unfortunately the interface between the computer and the sensors serving it tends to be of a special nature for each task, and in general no vast market exists to warrant the development of micro-miniature circuit elements to perform the interface function. The size of the equipment is dictated by the complexity of this interface between the sensor and computer, which must be tailored for each particular task. The computer must, of course, communicate with other equipment, either to other sub-systems or to some central computer. This will be by some form of data link which is of sufficiently common nature to justify special micro-miniature circuits being developed, and in general should not present a problem in terms of complexity.

The case for separate sensor computers stands or falls on the size of the interface associated with them. That is whether or not it is economic to have these separate computers, each with its individual interface section, rather than a large scale computer which warrants a complex interface which is shared among a much wider group of sensors.

By careful organisation of the interface to share the accurate analogue circuits which are essential to it and which create most of the bulk, it is possible to hold the size within economic limits. This paper describes some techniques which have proved successful in reducing the size of the interface. The key to the problem is the requirement for buffer storage between the computer and the interface - this must be of both small size and low cost. This buffer need only store data between successive computer cycles - a time ranging between 30 milliseconds and perhaps 300 milliseconds. For this purpose a capacitor proves to have all the necessary properties, and circuits exploiting this method of buffering can lead to great economies. The use of capacitors to buffer a frequently updated DC voltage is, of course, a very obvious example of this technique, and is certainly too obvious to dwell on. However, in combination with other circuit elements, this technique can be enlarged to include the storage and modulation of AC voltages and to buffer direct current pulses. It is in these two areas that the greatest economies have been realised.

2. THE INTERFACE PROBLEM

The problems which must be considered in studying the economics of the interface between sensors and computers fall under the following headings.

(a) It is necessary to share the accurate and bulky analogue-digital conversion circuits as widely as possible in a particular equipment. Every attempt should be made to have only one such conversion unit.

(b) The buffer storage between the computer and the interface can be bulky. The size and complexity of this must be closely controlled.

(c) Does the data from or to the sensor change rapidly? If so, can the conversion circuitry and the buffer storage be arranged to deal with rapidly changing signals?

(d) The interface and the computer may form part of a closed loop control system, in which case the computing speed and properties of the interface affect the stability of the control system.

Although the interface problems of different systems are different, both in the accuracy required and in the format of the data dealt with, the design techniques used to reduce the material content of the interface unit appear to follow similar rules.

The quantities to be dealt with in the interface between the analogue sensors and the computer in the system mentioned in the Table on Page 8-1 can be grouped under the following headings.

Accurate direct current pulses - both input and output

Synchros - both control transmitters and control transformers, including four-wire devices such as resolvers

AC and DC voltages - input and output to and from sensors and aircraft

Self-balancing bridge

The problem of encoding the DC voltage through a multiplex analogue to digital convertor is commonplace, and requires no further comment. LSI packages already exist for this function. Methods of dealing with AC voltages and the self-balancing bridge can be considered as variations of the technique used to provide the more complex synchro angular transmission. The problems of producing a small interface unit can be concentrated in the two remaining areas, that is

(a) The quantizing of direct current. This is required to an accuracy of better than one part in 10^4, both in magnitude and in time; and

(b) Simulating both the functions of a synchro angular transmission, that is the synchro transmitter and the control transformer

These widely separate functions are entirely different and yet the methods to reduce the material content prove to follow very similar rules.

(a) Share the accurate generation of analogue quantities as widely as possible

(b) Avoid the need to interrupt program for input/output functions. This simplifies the buffer storage between the computer and each parameter in the interface.

(c) Organise the flow of data such that the computer is not involved in any closed loop response of high bandwidth

The magnitude of the problem if the buffer storage is in conventional digital form in the case of an output signal is briefly as follows. This is shown in Figure 8.1.

The logical elements to provide this buffer storage are readily available at an economic price. However, the sheer bulk of the circuit elements required makes it necessary to look for something smaller to provide the storage between successive connections to the computer.

The order of magnitude of the storage problem is as follows. The data to be stored will be represented by a digital number of around sixteen bits for each output function. This will be transmitted from the computer, probably in parallel form, and must be stored in a group of latching circuits holding the data until it is updated again. The computer will be connected to the interface at a random time in an interval of between 30 and 300 milliseconds. This requires logic switching to connect the output from the computer to the correct set of latching storage circuits. This switching, together with the latches, provides a continuous output as a digital number.

The digital data is then translated into analogue form in a digital-analogue convertor. The physical size of the circuit needed to provide this function per channel can vary from 18 sq. in. of printed circuit card area downwards, dependent on what special function elements are available.

The digital elements may be separate from the digital to analogue convertors, and the magnitude of the interconnecting circuits needed to transmit parallel digital data on this scale within the system can be considerable.

3. ECONOMIES FROM TIME SHARING INTERFACE FUNCTIONS

How can an economy be effected? Perhaps by closely integrating the circuits using the full power of large scale integration to reduce the bulk of the digital to analogue convertor. However, as has already been said, the interface is the one part of a system which does not lend itself to standardisation. The computer will be common to many projects, and has benefited from this. The commonality of its complex function elements has justified the degree of integrated circuit development needed to make it economic in both size and in direct cost. This large market does not yet exist for the special interface elemenets which must be tailored to the requirements of a particular system.

An alternative method of providing the output interface has been developed which gives a great economy and is being introduced in this paper. The key lies in the buffer storage between the computer and interface. This does not need to be digital in format. Changing to analogue buffer storage can give great economies and, if coupled to the variable resistance properties of field effect transistors, multi-purpose units of great power can be constructed. In all the variants of this technique, the storage element is common, that is a capacitor. The method can be best explained by two examples.

(a) Gyro and Accelerometer Interface

This is an essential part of any Inertial Navigation system (Fig. 8.2). The accelerometer consists of a current balanced pendulum, any displacement of the pendulum being balanced by a current through the balancing coils. The accelerometer reset loop is closed through a capacitors, this charging up to a voltage which is a function of velocity change as sensed by the accelerometer. At a suitable time this is inspected by the computer, and, if above a certain level, a reset current pulse is fed into it from a common current pulse-generating circuit. This pulse represents a standard velocity increment. In this case the condenser is acting as the buffer storage, and is scaled to store the equivalent of several velocity increments - sufficient to carry over any period whilst the computer is committed elsewhere, for example the integration sub-routine which is in real time and cannot be interrupted in a simple fashion.

The gyroscopes are precessed by the same current pulse-generator, each pulse representing a small angle turned through by the platform. Again, a buffer condenser is provided to limit the voltage developed across the comparatively high resistance of the gyro torquer.

The size and cost of the bulky current pulse-generator, which must be accurate to 0.005%, are very effectively reduced by sharing it around these six sensors, that is the three accelerometers and three gyroscopes. The priority of dealing with a particular sensor is allocated within the interface, and not in the computer.

(b) Synchro Interface

This includes the interface units needed to transmit an angle from the computer in the format of a synchro transmitter and that needed to accept a signal from an external transmitter through an equivalent control transformer.

The control transformer function falls under two headings. An angle from a control transmitter has to be converted into digital form and stored within the computer. Alternatively, the computer performs the function of a control transformer, that is adding to the output from a control transmitter to generate an analogue error signal. Both of these conditions must be satisfied. The simpler arrangements for a synchro transmitter are shown in Figure 8.4.

The economy in the interface unit arises from a circuit (Figure 8.3) which could have very wide use in computer interfaces. This may be considered as a modulator which is controlled by the storage element - a condenser. The gate to source voltage set on the condenser determines the source to drain impedance of the FET shown, which provides one arm of a bridge. By controlling the condenser voltage, the output of the bridge as a proportion of its input can be varied, and obviously this can either be an AC or a DC signal.

The great attraction of this arrangement is this proportionality; the supply to the bridge may be a variable - the circuit merely outputs some demanded ratio of it, and it is this property which makes it such a powerful tool. If the circuit is to be used as a precise modulator of an AC signal, the input levels must be kept small due to non-linearity in the FET bridge. Harmonic distortion from this source can easily be reduced to a level less than 1% - a level which causes no deterioration in the performance.

(i) An Equivalent Synchro Transmitter

The modulator can be arranged to provide an equivalent synchro transmitter output which may be shared to provide many outputs. This is shown in schematic form as Figure 8.4. In this arrangement only one set of latching circuits and a single digital to analogue convertor are needed, these being time-shared between channels as required. Switches gated from the computer coincident with the setting of the latches control the updating of the charge stored on the condenser, and thus form the output as a ratio which represents one phase of a synchro transmitter. This output meets the normal requirement for a synchro transmission, in that it may be excited from an external source. Performance of the FET modulator and the accuracy of a typical synthetic synchro transmitter are shown in Figures 8.5 and 8.6.

(ii) Synchro Control Transformer

Using similar circuit elements, the interface may be adapted to act as a control transformer, that is the synchro receiver.

(c) Operation in a Control System

The two examples discussed so far are part of closed loop control systems, and as such their performance affects the servo bandwidth and stability of the overall closed loop. In both cases the method used removes the computer from the actual control loop. The computer merely acts on the data from or to the closed loop, and does not appear as a function in the overall closed loop transfer function. This is a very important consideration as the control loop servos may now be designed without regard to the data rate possible with the computer in circuit.

As a general rule, such a sampling system behaves rather like a transmission line, giving approximately 45° phase shift at a pulsatance of approximately one-third of the rate at which the computer cycles the data. This imposes a very severe limit on the servo bandwidth possible if the control signal is taken into the computer and the error computed digitally before transmission to the control element. By avoiding the need to digitise any part of the control loop, this problem is completely removed. This condition is met in the two examples given.

To give practical examples, the accelerometer capture loop is closed with a bandwidth exceeding 600 c/s and synchro follow-up servos have been operated at bandwidths of around 20 c/s using a 400 c/s carrier. Servo elements designed for normal analogue use have operated without modification in these cases.

4, CONCLUSIONS

The interface of a sensor computer can be so bulky compared to the small computers possible at present that it can dominate the size of the equipment. The cost of such an interface and the cost implications of its bulk prove to be a real limit as to whether such self-contained systems are feasible.

By careful choice of circuit elements, one or two standard circuits which may be time-shared can be arranged to deal with the complete analogue-digital interface of the system associated with such a small sensor computer. This sharing of common functions reduces the size of the interface to a level compatible with that of the computer without development of special integrated circuits. The combination of thin film techniques in conjunction with standard circuit elements on printed circuit boards proves to be an adequate form of construction. The methods chosen to reduce the size of the interface also prove to have the correct properties required for interfacing with the control elements of active sensors and save a vast amount of high speed digital processing if the control loop of the active sensor is to be converted digitally.

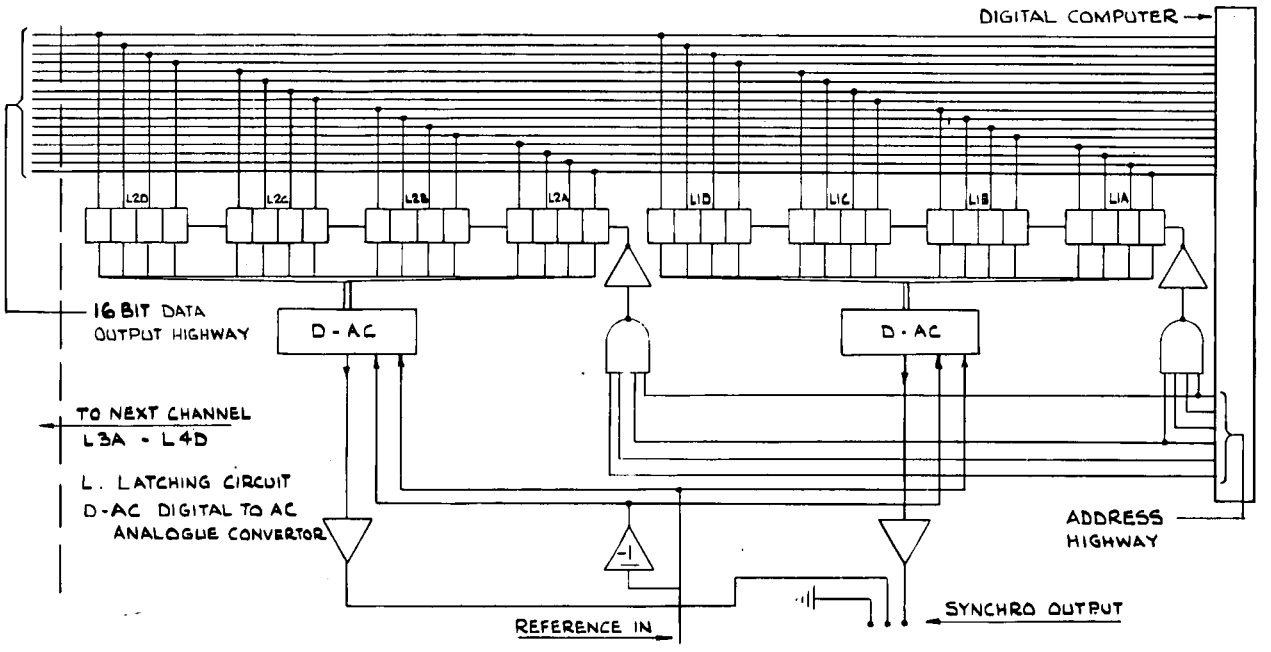


Fig.8.1 Conventional synchro output from digital computer

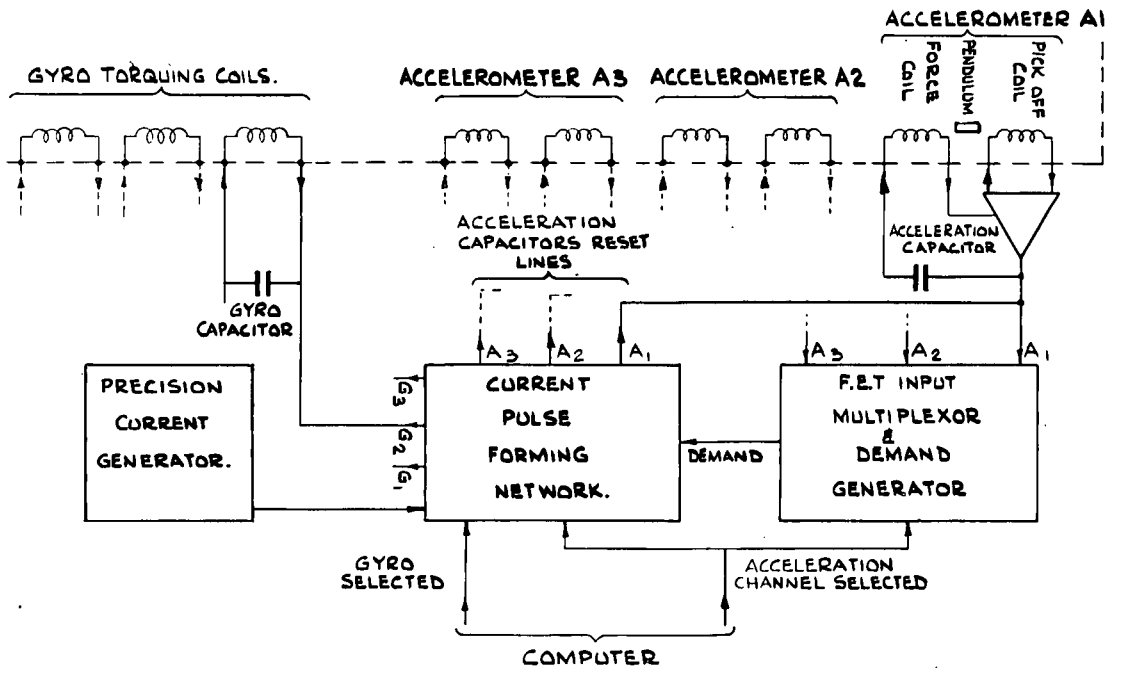


Fig.8.2 Gyro and accelerometer interface

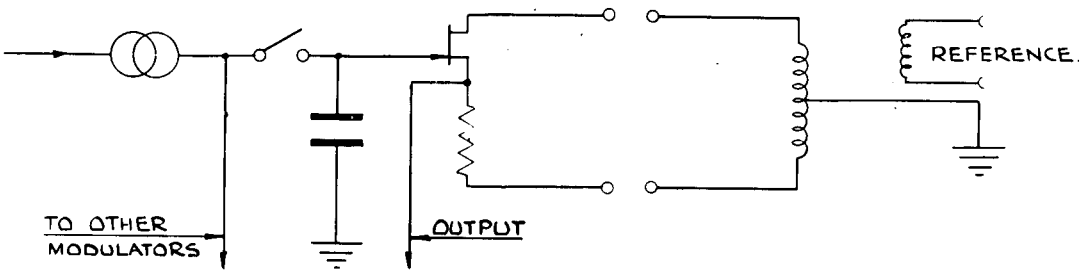


Fig.8.3 Modulator

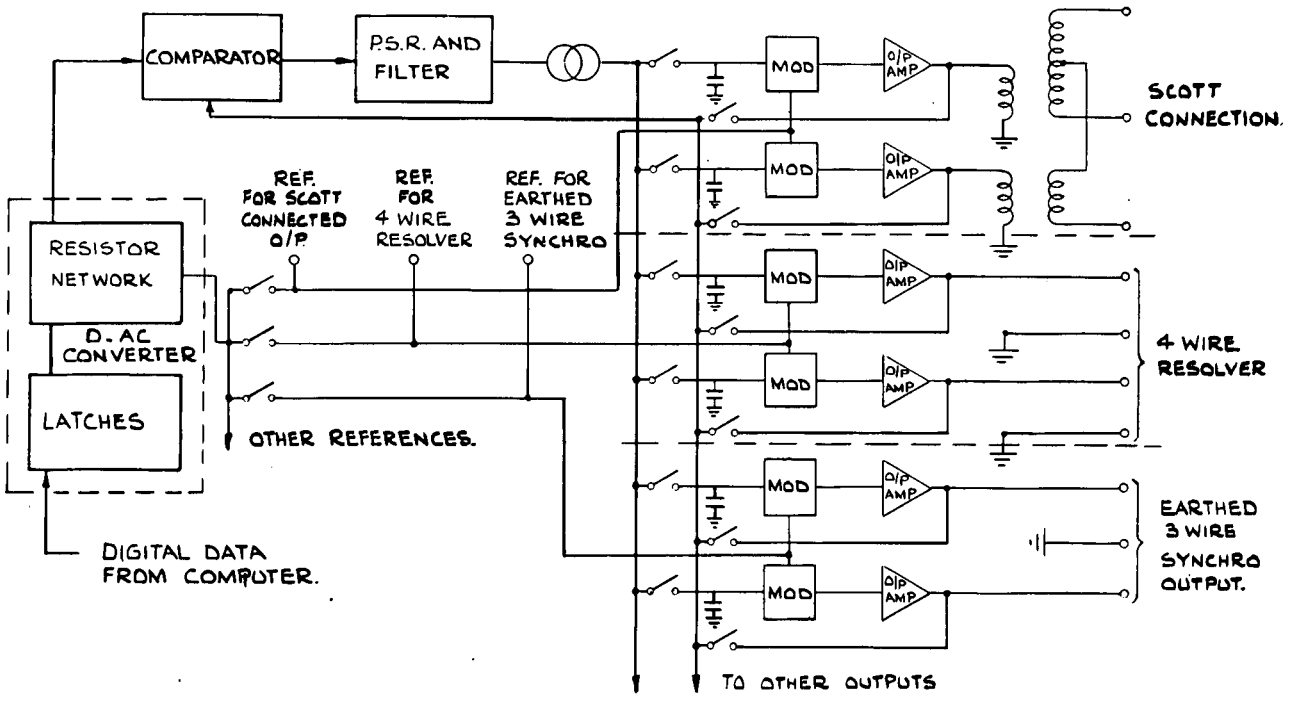


Fig.8.4 Schematic synchro converter

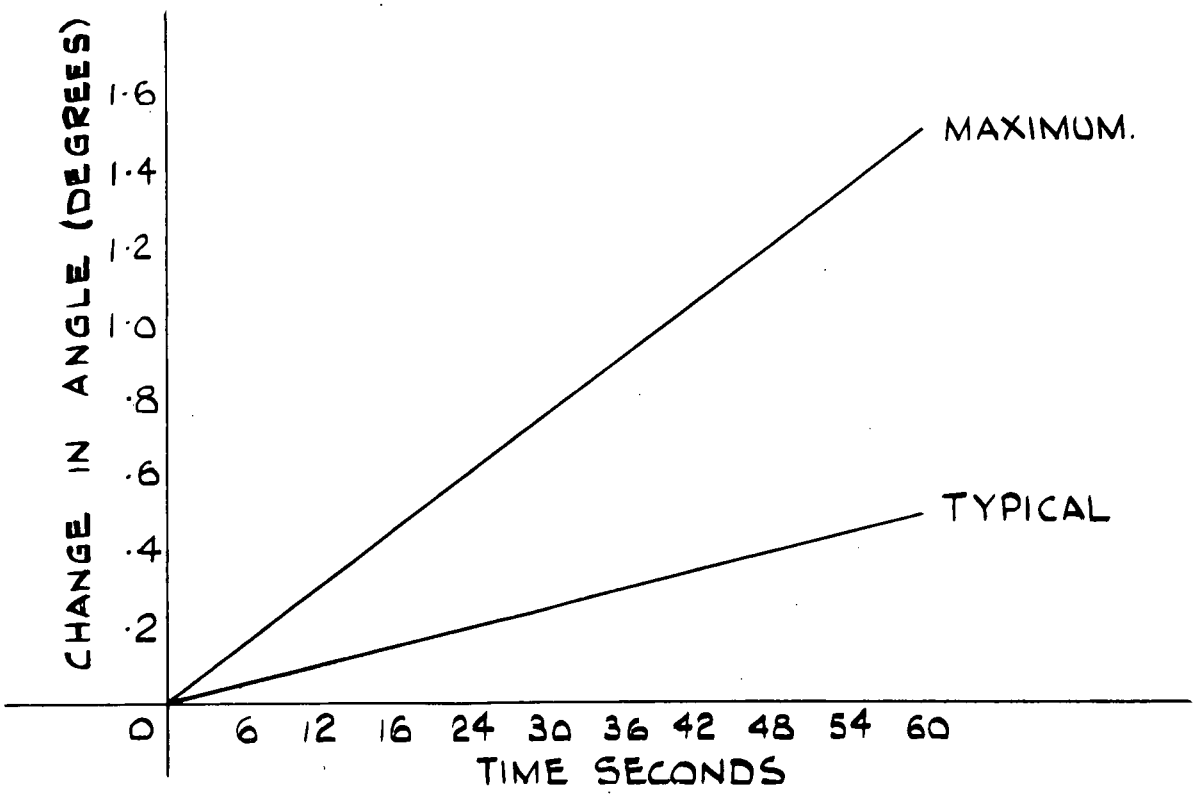


Fig.8.5 Output angle drift

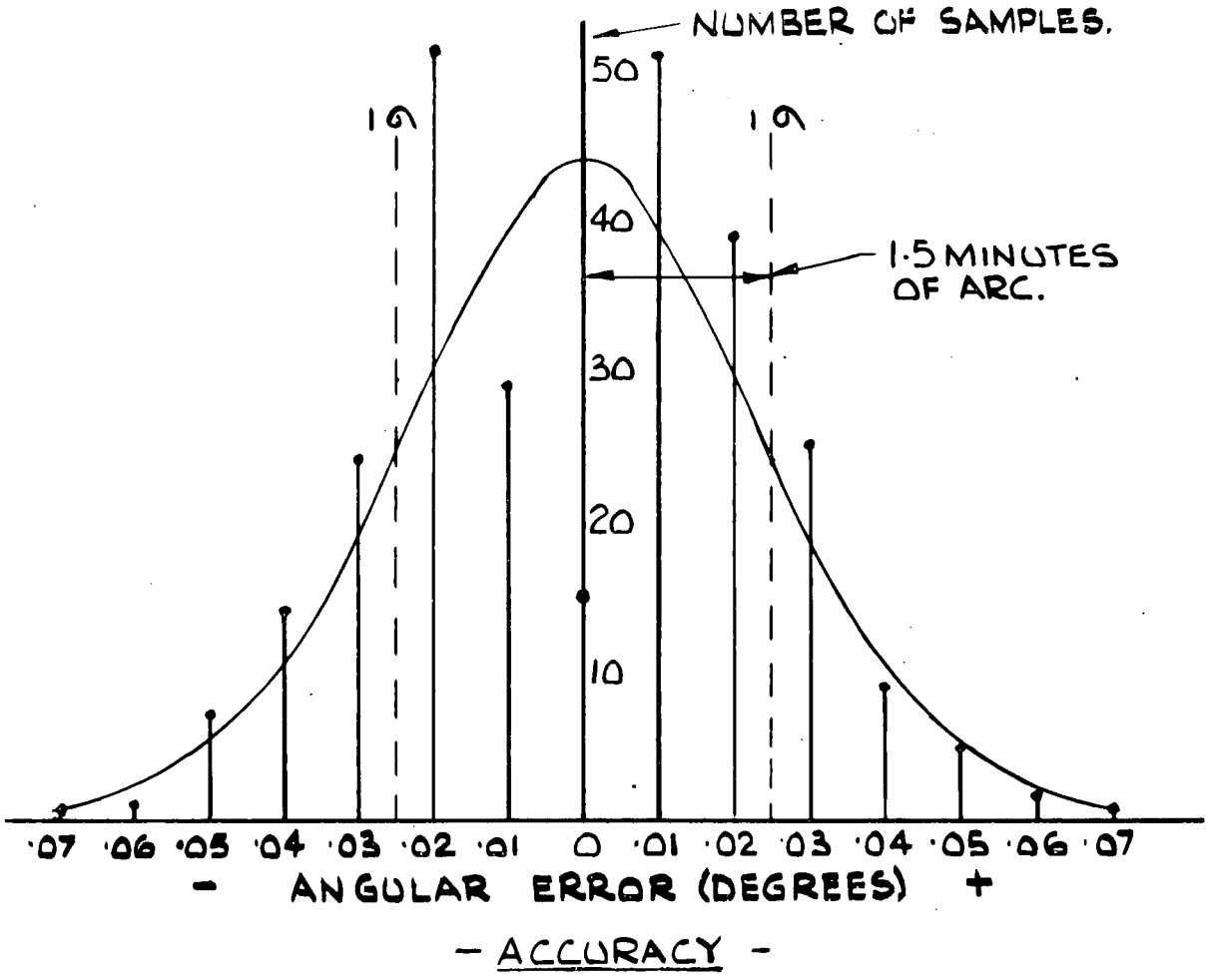


Fig.8.6 D/A resolver output performance

FLIGHT RESEARCH EXPERIENCE WITH GUIDANCE AND CONTROL COMPUTERS RELATED TO GENERAL APPLICATIONS

Melvin E. Burke
NASA Flight Research Center
Edwards, California, USA

SUMMARY

Several guidance and control research programs involving the X-15 and F-104 airplanes are discussed, with the discussion oriented toward airborne digital computer utilization. An analog and a digital systems mechanization are compared, and the performance advantages of the digital system are pointed out. The flexibility of the digital computer as a research tool is indicated, as are advantages of decentralized computers. Application of a general purpose computer to the solution of strapdown system equations was successful in the laboratory in preparation for a flight program. The effects of input-output mechanizations on software complexity are discussed. The utility of a general purpose digital computer is shown by its flexibility in being used for various research tasks. This utility is degraded, however, by the effort required to write programs in machine language for real-time applications.

INTRODUCTION

Flight research activities at the NASA Flight Research Center have provided systems engineers with the opportunity to investigate airborne computer utilization in control and/or guidance systems in both conventional and research aircraft. The computers used have included both analog and digital mechanizations, as indicated in the brief compilation of table 1. Generally, the applications have followed the trends seen in operational vehicles; that is,

TABLE 1.–CONTROL/GUIDANCE COMPUTER USE

Aircraft	Type	Computer		Comments
		Control	Guidance	
X-15	Research	Analog	Analog/digital	2 airplanes with stability augmentation system 1 airplane with adaptive flight control system
JetStar	Production (configured for research)	Analog	----------	Computer for model-following airborne simulation (variable stability)
Lifting bodies (M2-F3, X-24)	Research	Analog	----------	----------
F-104	Production	Analog	Digital	Special configuration for guidance research
PA-30	Production	Analog	----------	Special configuration for general aviation control systems research
F-8A	Research	Analog	----------	Special configuration for supercritical wing evaluation

analog devices in control systems and digital devices in guidance systems. Several of these programs have been specifically oriented toward control and guidance investigations which required the use of airborne computers as integral components in the systems.

The utilization of computers to provide guidance information to the pilots in the X-15 research program, which was concluded in the fall of 1968, will be considered first. This will be followed by a review of current guidance research programs now entering flight phase in an F-104 airplane. The computer applications in these programs are described briefly. On the basis of the experience gained, the discussion emphasizes the following computer application questions: analog versus digital, centralized or decentralized computers, general purpose computers in special purpose applications, and hardware versus software complexity. The advantages of the digital computer for research purposes, because of its flexibility, are pointed out.

Most of the discussion summarizes investigations conducted in the laboratory during systems integration and simulation. Although the laboratory is the proper place for the resolution of systems problems, often computer systems problems occur with simultaneous operation of other systems or are unique to the aircraft environment. These problems are revealed only during flight testing. Such problems are discussed, and some flight data are presented. Conclusions are drawn that may affect or apply to future applications of digital computers as integral components of aircraft systems.

DISCUSSION

X-15 PROGRAM

The X-15 was a rocket-powered research airplane designed to be air-launched from a B-52 carrier aircraft at about 45,000 feet (13,604 meters) altitude and Mach 0.80 (1). The X-15 had a flight envelope extending to speeds above Mach 6 and altitudes above 350,000 feet (106,680 meters). A typical time history of velocity and altitude for a high-altitude mission is shown in figure 1. Because of these extreme ranges in velocity and altitude, an inertial flight-data system (IFDS) was required to provide the pilot with displays of aircraft velocity, altitude, and three-axis Euler angle attitude information. Initially, all three of the X-15 airplanes had analog inertial flight-data systems. In two of the airplanes, the systems were later converted to digital systems because of the poor reliability of the analog system (2). The third airplane retained a redesigned version of the analog system. One of the airplanes that was converted to the use of a digital system was equipped with the self-adaptive flight control system and was later also configured with a high-speed digital computer to provide the computer capability required to conduct guidance investigations.

Analog Versus Digital Mechanization. The conversion of two X-15 airplanes from analog to digital inertial flight-data systems provided a unique opportunity to compare the relative merits of the two system mechanizations operating in the same environment and performing essentially the same functions.

The analog system was mechanized as shown in the simplified flow diagram of figure 2 to provide pitch (θ), roll (φ), and heading (ψ) attitudes, total velocity (V_t), rate of climb ($\dot{h}$), and geometric height above the surface of the earth (h). In this mechanization within the computer, a wide range of electromechanical components was used. These components included choppers, servomotors, tachometers, amplifiers, and potentiometers. They were grouped in three major packages--an erection integrator, a velocity integrator, and a position integrator--with each package processing data for all three axes--vertical, range, and cross range. This mechanization was simplified somewhat because of the bounded range within which the X-15 airplane was designed to operate.

The solution of the guidance equations in the digital system was mechanized as shown in figure 3. This mechanization, while basically quite different from the analog, provided essentially the same information to the pilot. The

mechanization was a combined digital differential analyzer (DDA) and general purpose (GP) computer. The DDA section was used for continuous summing of the accelerometer outputs because of its inherent speed in accomplishing this type of operation, and the GP section was used for the solution of equations which required more precision but less speed. The memory of this computer was a rotating disk with a capacity of 1664 24-bit words. This system was mechanized to provide the reference axis in the north, east, and vertical directions. Gyrocompassing techniques were used for initial alignment of the system.

The vertical loop in an inertial system is normally not mechanized to provide altitude information. When the loop is mechanized, it must be damped from an external source to prevent divergence. When it is switched from damped to undamped as it was in the X-15 during its free flight, the rate of divergence is an early indication of the overall system performance. This was particularly true in the X-15, where pure inertial operation was limited to the relatively short duration (approximately 10 minutes) of the free flight. The analog system was continually erected through interface with systems carried on the B-52 airplane until the X-15 was launched. The digital system, however, operated in the navigational mode with only the vertical loop slaved until launch. Thus, it would be expected that the analog system would outperform the digital system. Data from 10 flights for each system presented in figure 4 indicate that this was not the case. These data are the mean inertial altitude error and its standard deviation as a function of time from launch of the X-15. The data were obtained by comparing the recorded output of the system with a reference curve obtained by smoothing data from ground based radar systems, pressure altitude systems, and independent acceleration measurements. The noteworthy points in these data are the standard deviations. These deviations are high for the analog system and consistently lower for the digital system, which indicates a more consistent performance of the digital system.

The analog system was marginally adequate for the X-15 program when aspects such as reliability and performance are considered. The factors that made the digital system stand out were more simplified operating procedures, consistent performance, and flexibility. The flexibility aspect of the digital system was especially significant when the airplane carried experiments which required computer-supplied information and processed forms of this information. On the basis of the experiences with these two system mechanizations in this program, the digital system would be the clear choice for use in similar applications.

Guidance Experiments. Two experiments were included in the later phases of the X-15 program that were specifically oriented toward the investigation of guidance concepts. The first, a manual boost guidance study (3), generated fly-to-null pitch attitude commands as a function of altitude, altitude rate, and total velocity. These commands were updated at the rate of 10 times each second and provided the pilot with guidance commands which placed him successfully on a preplanned trajectory at burnout. The second, an energy management (EM) study, did not reach flight status but was extensively evaluated on ground based computers and simulators (4). Since energy management required far more computing capability than did boost guidance, it will be described further to illustrate the complex program that a digital computer is capable of handling. Additional computing capacity was required on the X-15 airplane to conduct these studies.

System Description: The basic aircraft systems configuration for the X-15 airplane used in these experiments is shown in the simplified block diagram of figure 5. This configuration is analogous to a decentralized computer concept; that is, each system operated independently, so that a failure in one system had little effect on the performance of any other system. This philosophy was followed throughout the X-15 program partially because of pilot insistence on system independence. No experiment was added to the X-15 which, if a failure occurred in that experiment, would affect the operation of the basic system. The addition of the digital computer to perform the necessary computations for the guidance experiments resulted in the systems configuration shown in figure 6. This was a prototype computer that used a 4096 24-bit word, random access, non-destructive read out, micro-biax memory. In this application, the computer interfaced with the air-data system analog computer, the IFDS digital computer, and the pilot's displays. Also, through the IFDS digital computer, it was interfaced with the adaptive flight control system.

Each system retained its operational independence except the computer added for guidance experiments. This computer was dependent on the other computers for its input data. In the limited operational experience with this computer, no failures occurred in other systems that affected its operation. However, during the overall X-15 program, several system failures did occur, and the pilot was able to recover the vehicle and often complete the mission successfully because the systems were sufficiently isolated that he received guidance information from either the air-data system or the IFDS. The integration of the additional computer with the other systems on the X-15 airplane presented no significant problems.

One problem which did occur was not directly related to the centralized or decentralized computer concept but is directly related to the application of digital computers in aircraft. This problem concerns the susceptibility of digital computers to power transients. On most aircraft, there is usually a rather large amount of noise on the power buss, and spike transients occur when systems are turned off. Computers are equipped with transient protection circuitry so the contents of the working registers can be stored for later recall when the transients are no longer present. The sensory circuits in the computers work well, the data are stored, and, when the power source is quiet, processing is restarted. The problem is especially apparent when this occurs under dynamic conditions and valuable sensor data are lost. If this happens often, the solutions are no longer valid. This problem occurred on the X-15 airplane from time to time and was sufficiently severe to justify a major effort to rectify it by rewiring the airplane and isolating the noise-generating systems to a separate buss.

Energy Management Study: The energy management study planned for the X-15 airplane was the flight evaluation of a concept developed by Bell Aerospace Systems, Buffalo, New York, for maneuvering lifting reentry vehicles. This concept was specifically tailored to the X-15 for evaluation. It utilized a predictor concept designed to compute the extreme dimensions of the physical area within reach of the X-15 airplane, which determined the ground area attainable (GAA). In addition, from these predictions, the anticipated maximum values for temperature, dynamic pressure, and normal acceleration were obtained. This was accomplished by repeated integrations of the differential equations of motion for a "model" vehicle and by the use of various energy-distance relationships. Aerodynamic coefficients and flight criticalities for the model were stored within the computer.

As shown in figure 7, when the vehicle approached the reentry phase, the energy management computations were initiated. Starting with the initial conditions of the "real" vehicle, the model was rapidly "flown" through a maximum range, a minimum range, and a maximum cross range flight, using the flight techniques and vehicle constraints

applicable to each. During any given prediction flight, the stepwise integrations could be discontinued once the model reached equilibrium conditions, after which simplified energy-distance relations were sufficient.

The energy management system also generated commands for both display and flight control. By selecting one of several pre-stored landing sites, the energy management program would determine the location of the site within the ground area attainable, and, based on this location, would generate both pitch and roll commands necessary to direct the "real" vehicle to the high-key point for the site selected. In addition, the commands necessary for phugoid damping were superimposed on the guidance commands so that any trajectory oscillations were eliminated. Vehicle constraints were fully respected during the generation of all commands.

This concept was evaluated on the X-15 ground based simulator by using a hybrid analog and digital computer facility. The simulation was later interfaced with the ALERT computer so that the ALERT program could be dynamically checked out prior to flight.

Utilization of a high-speed digital computer interfaced with the X-15 systems proved to be a feasible method of conducting research investigations when the added computing capacity was required. These two programs, manual boost guidance and energy management, illustrate the great degree of flexibility that a digital computer provides, particularly in a research application.

GUIDANCE RESEARCH PROGRAM

An F-104 airplane is being used at the NASA Flight Research Center as a test vehicle to conduct research studies on strapdown guidance and real-time flight-path optimization. A general purpose digital computer is being used in both of these studies. This has provided an opportunity to investigate the use of this type of computer in an application normally handled by a special purpose (SP) computer and to investigate the trade off between hardware and software complexity. These investigations were conducted as required during the strapdown system study. The flight-path optimization study represents an investigation into the use of a general purpose computer for real-time applications. Although flight results are not yet available, much of the information gained in the laboratory is pertinent to airborne computer utilization and is further described.

Strapdown Guidance System. Briefly, the basic difference between a strapdown and a gimbaled inertial guidance system is the mechanization to establish a reference coordinate system within which acceleration measurements may be integrated for navigation purposes. In a gimbaled system this is accomplished through the gimbals and gyro feedback to position the gimbals, whereas in a strapdown system it is accomplished within the computer, as indicated in the following equation:

$$\begin{bmatrix} \int_0^t a_E\,dt \\ \int_0^t a_N\,dt \\ \int_0^t a_V\,dt \end{bmatrix} = \underbrace{\begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix}}_{C(T)} \begin{bmatrix} \int_0^t a_x\,dt \\ \int_0^t a_y\,dt \\ \int_0^t a_z\,dt \end{bmatrix}$$

In the gimbaled local vertical system the coordinate transformation matrix C(T) is unity or some preselected value for convenience of mechanization. In the strapdown system C(T) must be continually updated from a measured set of initial conditions. Strapdown systems to date have utilized computers specifically designed for them, often with the coordinate transformation matrix hard-wired for speed of solution. Little has been reported on the use of a purely general purpose type of computer in this application.

The system assembled for this investigation is shown in a simplified block diagram in figure 8. This system, as indicated, consists of the inertial measurement unit (IMU), an input-output unit, control panel, a power supply, a pilot's display, and the ALERT digital computer. The design of the system was not optimized; that is, major components such as the IMU and the computer were obtained from other programs. This meant that certain hardware and software conditions were set. These conditions included allowable interconnect cable length, maximum allowable angular rates, the format of the IMU data word, and the IMU sample interval.

The IMU consisted of three pulse-torqued gyros, three pulse-rebalance accelerometers, and the necessary electronics for closing the loop about each of these sensors. The maximum allowable angular rate that the gyro torquers were capable of maintaining was 20.8 deg/sec (0.363 radian/sec). The computer used had a 2-microsecond access core memory of 16,384 24-bit words. The central processor used fixed-point arithmetic.

Normally, complete data acquisition systems are installed in each airplane for flight research programs. However, because of manpower limitations, it has been necessary to proceed on this program with a limited data acquisition system. To supplement this system, some 8,192 words of the computer's 16,384 words of memory were reserved for the storage of important parameters generated in the computer that would normally have been recorded on the data acquisition system. Data will be stored as 12-bit words, giving a total capability of some 16,000 words.

Strapdown System Coordinate Transformation Solution With a General Purpose Computer: Two tasks were required to accomplish the solution of the strapdown system coordinate transformation problem. First, provisions had to be made to transfer data from the IMU to the computer. Six channels of digital data, with each channel representing a velocity change (ΔV) along a reference axis or an angular change ($\Delta\theta$) about a reference axis, had to be input to the computer. Second, an algorithm had to be selected to perform the transformation. This algorithm had to provide a sufficiently accurate solution and yet not be so cumbersome as to consume too much of the computer's computational cycle.

The computer has two modes of inputing data: a direct memory access (DMA) under peripheral control, and a direct input-output (DIO) under program control. The DIO mode was selected to retain program control. All data

must be input in a whole-word parallel format. This required an input-output (I/O) unit that was capable of accumulating and storing data in registers for inputing on computer command. Since the IMU had such registers, this could have been accomplished by direct interface with the computer and IMU. However, a finite amount of time is required for the computer to transfer these data (about 175 μsec), and the registers must be inhibited while that transfer takes place. This would cause a loss of data during each transfer, with the accumulative effects causing considerable errors. This problem was circumvented by using registers in the input-output unit which tracked the registers in the IMU. When the computer was ready to transfer data, the registers in the input-output unit were inhibited. Following transfer, the I/O registers were again updated by the IMU registers.

It was arbitrarily decided early in the program that no more than 25 percent of the computer's memory capacity and computation cycle would be allocated to the strapdown solution, thereby leaving some 75 percent for other uses. To insure this, the ALERT simulation available on the Flight Research Center's data processing computer was used. A first-order algorithm and a second-order algorithm were evaluated with the results shown in table 2. The

TABLE 2.– STRAPDOWN ALGORITHM EVALUATION FOR ALERT GENERAL PURPOSE COMPUTER

	First order	Second order
Iteration rate	800/second	50/second
Interval	1250 μseconds	20 milliseconds
Execution time	950 μseconds	3000 μseconds
Percentage	76	15
Memory space	145 words	125 words

first-order algorithm required a high iteration rate (800 per sec) to maintain a reasonable accuracy. At this high iteration rate a large percentage (76 percent) of the computer computation cycle was required. The second-order algorithm with its higher accuracy solution required a much lower update rate (50 per sec). This resulted in a much lower percentage of the computer cycle being used for this computation. The data transfer from the IMU was thus established at about once every 14 milliseconds. Further investigation showed that a complete solution of the algorithm was required for about every fourth data transfer, or every 56 milliseconds, to achieve sufficient navigation accuracy.

The algorithm selected was as follows:

$$[C(T + \Delta T)] = [C(T)][C(\Delta T)]$$

where

$$[C(\Delta T)] = \begin{bmatrix} 1 - \frac{h_2^2}{2} - \frac{h_3^2}{2} & -h_3 + \frac{h_1 h_2}{2} & h_2 + \frac{h_1 h_3}{2} \\ h_2 + \frac{h_1 h_2}{2} & 1 - \frac{h_1^2}{2} - \frac{h_3^2}{2} & -h_1 + \frac{h_2 h_3}{2} \\ -h_2 + \frac{h_1 h_3}{2} & h_1 + \frac{h_2 h_3}{2} & 1 - \frac{h_2^2}{2} - \frac{h_1^2}{2} \end{bmatrix}$$

and

$$h_i = (n_i)(S_{F_i})$$

n_i = number of pulses

S_{F_i} = gyro scale factor

i = gyro 1, 2, 3

The matrix C(T) represents the previously computed value of the solution to the algorithm; h_i is the measured angular change for that particular gyro for a given measurement period ΔT; $C(\Delta T)$ indicates the second-order solution within the matrix.

The success with which this general purpose computer has been applied to the solution of the strapdown system coordinate transformation problem indicates that general purpose computers may be serious contenders for applications previously restricted to special purpose computers. This is particularly true in research programs or other applications where additional computer capability may be required and yet physical constraints limit the system to one computer. This could, of course, be resolved by using a combined SP/GP computer, but the pure general purpose computer gives greater flexibility for overall hardware utilization.

Hardware Versus Software Complexity: The analog to digital (A/D) conversion in the input-output unit represented an interesting situation relevant to the question of hardware versus software complexity. Rather than undertake the extra costs of developing new circuitry for performing the A/D conversion, it was decided to utilize circuitry already developed by the computer contractor for a strapdown system computer. The problem was that it placed the burden of conversion on the computer program. When a 12-bit word A/D read-in was called, the last value stored in memory was compared with the newest value. If the two differed, the difference was added to the old value in memory and the computer was required to recycle through the conversion again, performing sufficient iterations until there were no changes. There was no provision for stopping the cycle until there were no changes. Each iteration took approximately 160 microseconds, and no two A/D channels could be serviced at the same time.

It was soon apparent that valuable computer time was being used servicing the A/D conversion in the input-output unit. This was no problem in the previous application of this mechanization because the computer was used only for the solution of the navigation equations and thus had ample idle time for servicing the I/O unit. In this application, however, the coordinate transformation and navigation solutions were but small portions of the computer use. Most of the computer cycle was to be used for the solution of the flight-path optimization problem, and, if this were to be accomplished in real time, servicing of the I/O unit had to be minimized.

A second I/O unit was ordered with adequate A/D conversion capabilities. In order to utilize the previously described unit, a change in programing was initiated which eliminated some of the iterations previously required. A test was applied which required agreement only to the fifth least significant bit before proceeding to other functions.

This experience with the I/O unit clearly demonstrates how the unwary may seriously compromise the software effort and indeed the overall computer capability through attempts to simplify the hardware. Fortunately, a solution was readily available through software manipulation that was not overly complex. The question of software complexity is still a serious problem to the general purpose computer user. The problems involved with machine language programing are significant when the computer is used in applications involving real-time solutions. Attempts to use more simplified programing techniques, such as Fortran, have proved to be too costly in machine efficiency to be practical.

Flight-Path Optimization. Techniques for optimizing the flight path of a vehicle that would provide savings in either time or fuel in maneuvering from one flight condition to another have been under investigation by researchers for some time. A flight evaluation of the gradient method, with the pilot using a placard of various cross-referenced Mach number and angle-of-attack commands, demonstrated a considerable savings in time required to get to cruise conditions (5). The major drawback of the techniques proposed to date has been that the computational requirements have been too great for real-time solutions on an airborne computer.

Real-time solutions for flight-path optimization offer considerable advantages when variables such as atmospheric temperature, excess thrust, and fuel flow are measured and the measurements are used in the optimization solution. In view of these advantages, optimization techniques, which include dynamic programing and Balakrishnan's epsilon method, for possible real-time solutions are being investigated and have shown considerable promise (6)(7). Figure 9 shows three Mach/altitude curves for an F-104 airplane to climb and accelerate to the end conditions of 40,000 feet (12,192 meters) altitude and Mach 1.9. Curve A represents the profile obtained by using the manufacturer's handbook procedures. Curve B was obtained by using the gradient optimization method. Curve C was obtained from Balakrishnan's epsilon method (7). Previous studies have shown a possible savings of 22 percent in time with the gradient method over the manufacturer's handbook procedures. The differences between curves B and C appear significant; however, the net time to reach the end conditions differs by only a fraction of a percent (6).

The dynamic programing technique has been programed for the ALERT computer and will be the first technique to be evaluated in flight. This technique, however, has significant limitations in the number of variables that can be handled. Analytical and simulation investigations of other techniques are continuing, and it is planned to phase these techniques into the flight program.

CONCLUDING REMARKS

A comparison of a digital and an analog inertial flight data system operating in essentially the same environment and performing the same function was made during the X-15 research airplane program. The advantages of the digital system were significantly greater than those of the analog system. On the basis of the experience with these two systems, it would be inconceivable to undertake another research airplane program similar to the X-15 program without requiring a digital inertial flight data system as one of the basic airplane systems.

The question of which concept to use, centralized or decentralized computer systems, is difficult to answer. Interfacing of computer with computer presented no significant problem. However, decentralized computers will probably be used on future research airplanes as a result of pilot objections to the routing of all display parameters through a single electronic component. This objection may be overcome with proper redundancy and demonstrated reliability.

Use of a true general purpose computer in applications normally handled by a special purpose computer can present some significant problems. These problems can be solved, however, and, from the standpoint of having extra computer capacity available for other problem solutions, it is often advantageous to use a general purpose computer. This is particularly true when the computer is being used in research programs.

The investigation of hardware versus software complexity was limited in this paper to the computer input-output unit. It was shown that attempts to simplify hardware mechanizations resulted not only in more complex software but also in consumption of the computer computation cycle. This, of course, was unsatisfactory when the computer was being applied to real-time solutions that required high utilization of the computation cycle.

The various research programs conducted with the same general purpose digital computer point out the high degree of flexibility of this computer. The biggest drawback to the use of digital computers in research programs is the high cost both in manpower and programing time. This is particularly true when near-real-time solutions and the most efficient machine language programing are required. This would not be nearly as great a consideration in operational use where the effort could be amortized over a number of aircraft and a period of time.

The digital computer has been widely used in guidance but has had only limited use in flight control systems for aircraft. It is expected that more digital logic circuits will be used in flight control systems as new sensors, such as the laser gyro, gain acceptance. Also, the acceptance of fly-by-wire controls will provide easier applications of digital mechanization. Certainly solutions of equations in statistical and nonlinear control theory are more suited for digital computers. It is anticipated that digitally mechanized flight control systems will evolve during the next decade. However, the takeover of flight control systems will probably not be as thorough as the takeover of guidance systems has been in the past decade.

REFERENCES

1. Stillwell, Wendell H.: X-15 Research Results With a Selected Bibliography. NASA SP-60, 1965.

2. Burke, Melvin E.: X-15 Analog and Digital Inertial Systems Flight Experience. NASA TN D-4642, 1968.

3. Lopez, Armando E.; Hardy, Gordon H.; and Gerdes, Ronald M.: A Study of Display Systems for Manual Guidance of Large Launch Vehicles Utilizing the X-15 Airplane and Flight Simulators. NASA TN D-5632, 1970.

4. Cockayne, William G.: Description of an Energy Management System for the X-15. NASA CR-96006, 1968.

5. Bryson, Arthur E., Jr.: Applications of Optimal Control Theory in Aerospace Engineering. 10th Minta Martin Lecture, published by Massachusetts Institute of Technology with the AIAA, 1966.

6. Taylor, Lawrence W., Jr.; Smith, Harriet J.; and Iliff, Kenneth W.: Experience Using Balakrishnan's Espilon Technique to Compute Optimum Flight Profiles. AIAA Paper No. 69-75, 1969.

7. Taylor, Lawrence W., Jr.; Smith, Harriet J.; and Iliff, Kenneth W.: A Comparison of Minimum Time Profiles for the F-104 Using Balakrishnan's Epsilon Technique and the Energy Method. Symposium on Optimization, Nice, France, June 29-July 5, 1969.

SYMBOLS

a_E, a_N, a_V	acceleration in east, north, vertical axis
a_x, a_y, a_z	acceleration in aircraft axis
[C(T)]	coordinate transformation matrix measured axis to computing axis, last computed value
[C(ΔT)]	coordinate transformation matrix, second-order update for interval of measurement
[C(T + ΔT)]	coordinate transformation matrix, last computed value plus second-order update
ΔE	change in east-west position, positive east
h	geometric height, feet (meters)
$\dot{h}$	velocity along the vertical axis (analog system)
$\ddot{h}$	acceleration along the vertical axis
h_o	initial height position
Δh	change in vertical position, positive up
L	longitude
ΔL	change in longitude
ΔN	change in north-south position, positive north
q	impact pressure
$\dot{R}$	velocity along the range axis
$\ddot{R}$	acceleration along the range axis
R_{max}	maximum range capability
R_{min}	minimum range capability
R_o	initial range position
T	temperature
t	time, seconds
V_E	velocity along the east-west axis, positive east (digital system)
V_N	velocity along the north-south axis, positive north (digital system)
V_o	initial velocity
V_t	total velocity, $\sqrt{\dot{R}^2 + \dot{x}^2 + \dot{h}^2}$ (analog), $\sqrt{V_N{}^2 + V_E{}^2 + V_V{}^2}$ (digital), feet/second (meters/second)
V_V	vertical velocity (digital system)
ΔV	change in velocity component
ΔV_E	change in east velocity component
ΔV_N	change in north velocity component
ΔV_V	change in vertical velocity
$\dot{x}$	velocity along the cross range axis
$\ddot{x}$	acceleration along the cross range axis
x_o	initial cross range position
α	angle of attack
β	angle of sideslip
γ_c	flight-path angle command
θ	pitch attitude
$\Delta\theta$	angular change in strapdown system
λ	latitude

$\Delta\lambda$	change in latitude
σ	standard deviation
φ	roll attitude
ψ	heading attitude
Ω_e	rotational earth rate

Subscript:

u	uncorrected for coriolis or centrifugal acceleration

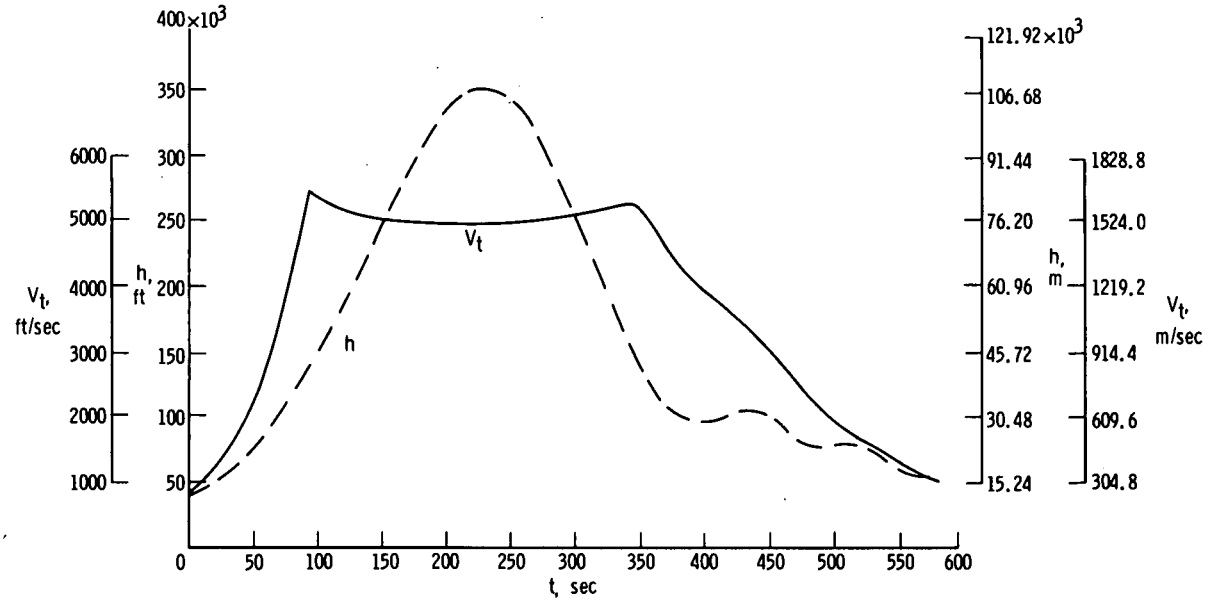


Figure 1. Typical time history of X-15 high-altitude mission.

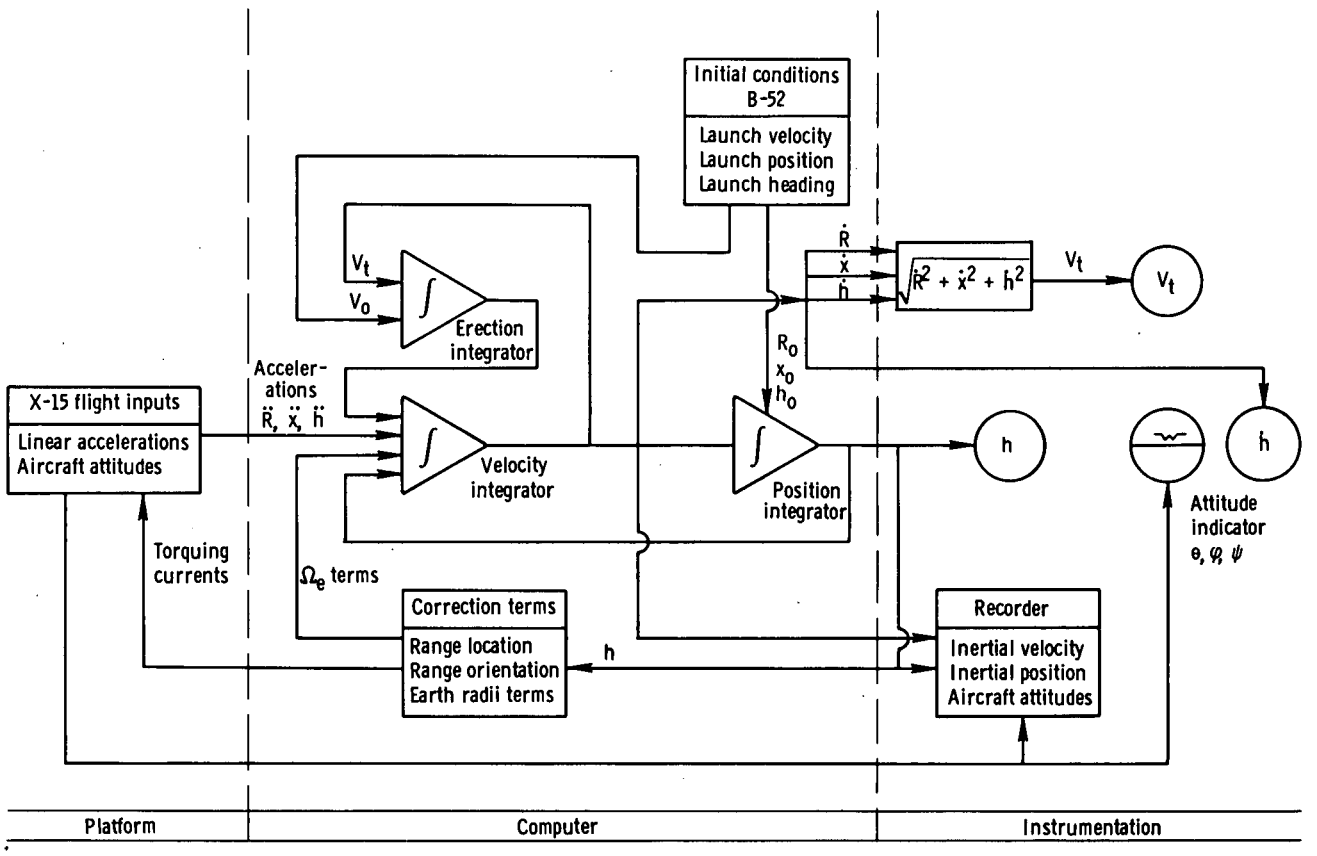


Figure 2. Flow diagram of X-15 analog inertial flight system.

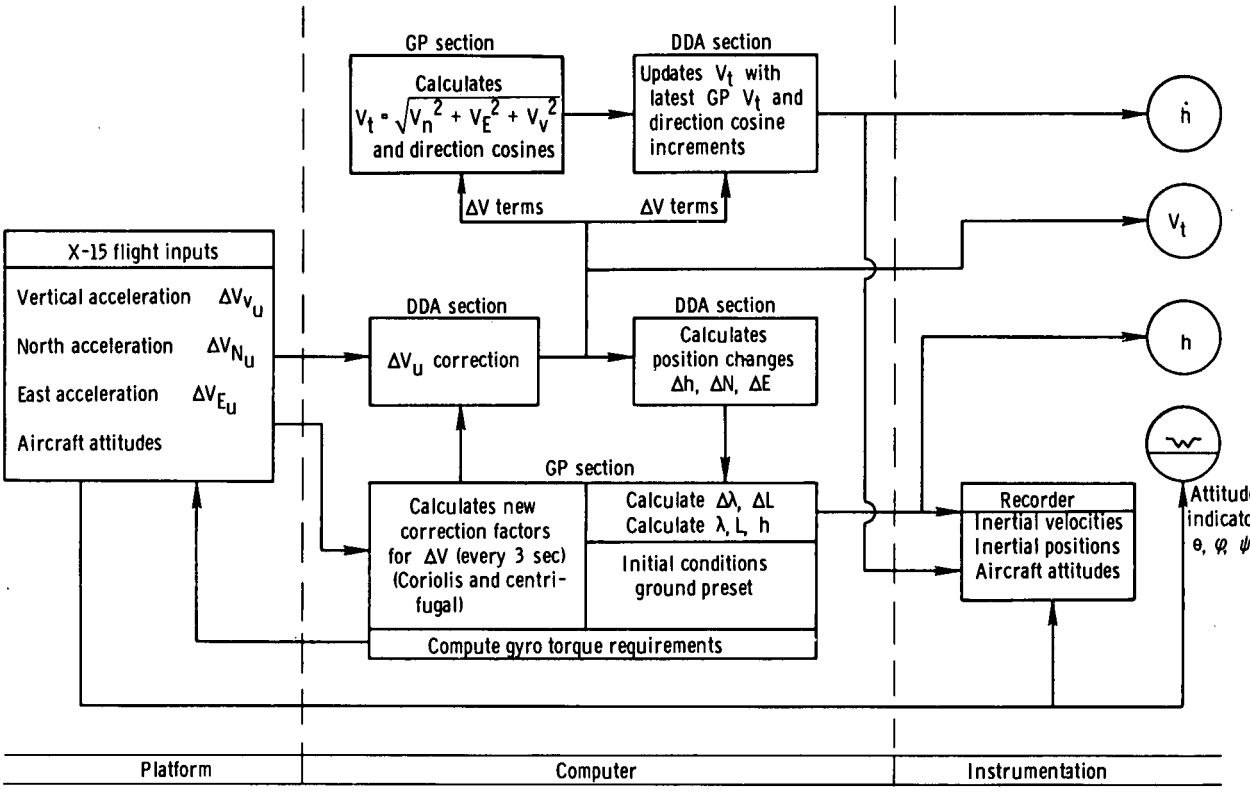
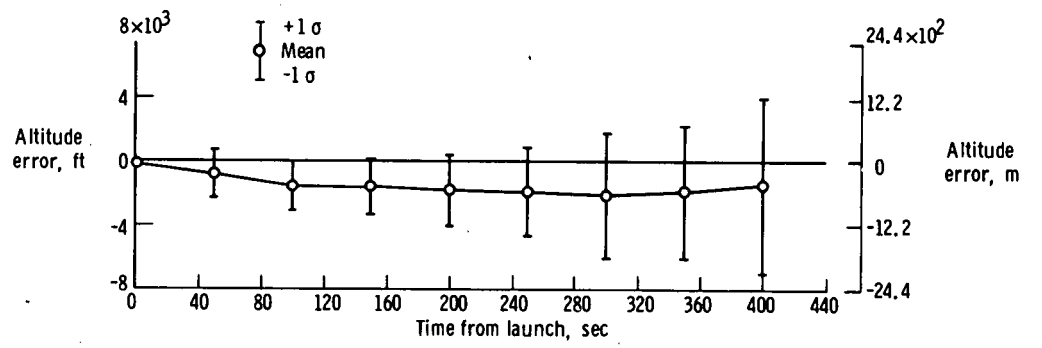
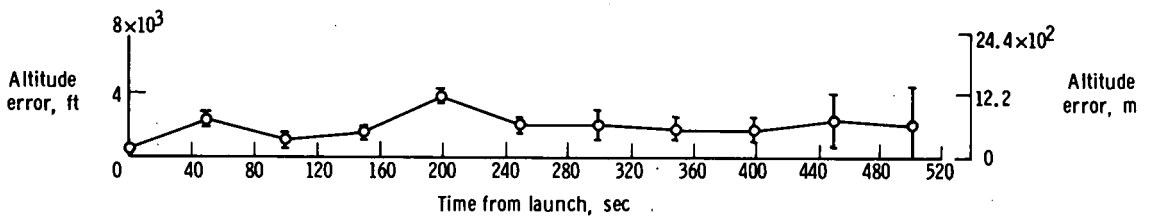


Figure 3. Flow diagram of X-15 digital inertial flight system.



(a) Analog system.



(b) Digital system.

Figure 4. Mean altitude error and standard deviation from 20 X-15 flights (10 each system).

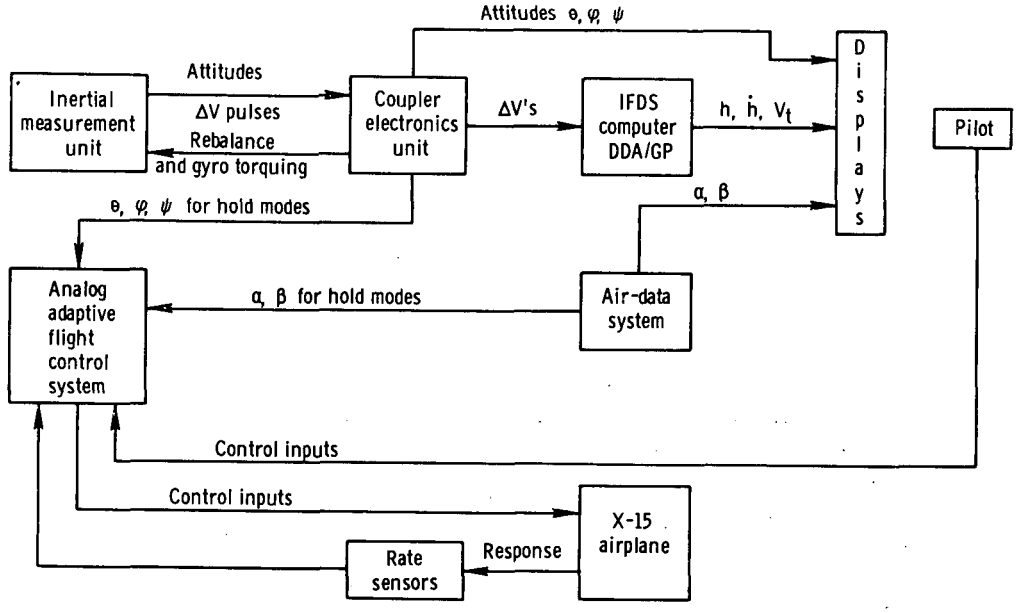


Figure 5. Basic X-15 systems interconnect diagram.

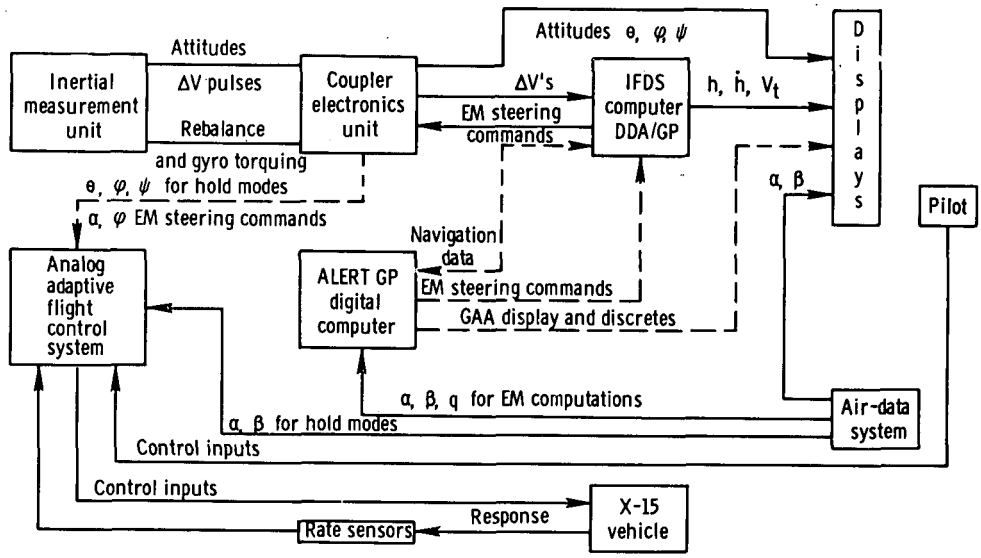


Figure 6. X-15 systems with ALERT computer.

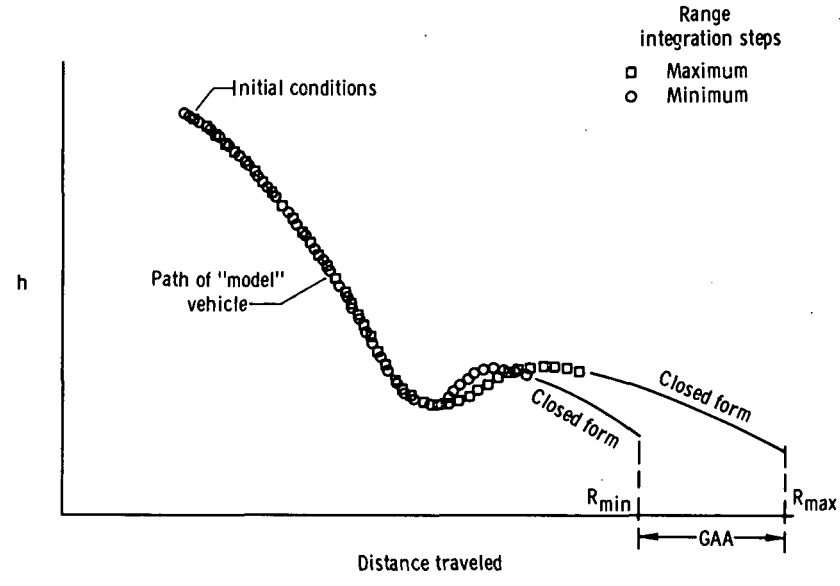


Figure 7. Generation of ground area attainable in the X-15 energy management simulator studies.

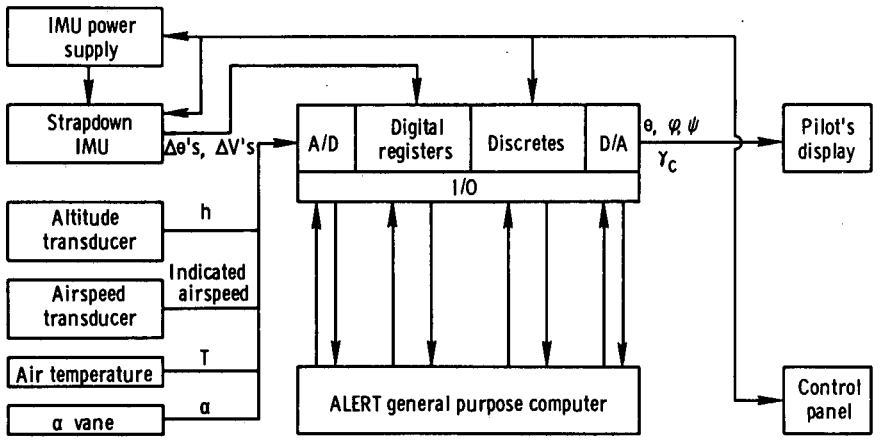


Figure 8. System interconnect diagram for strapdown evaluation.

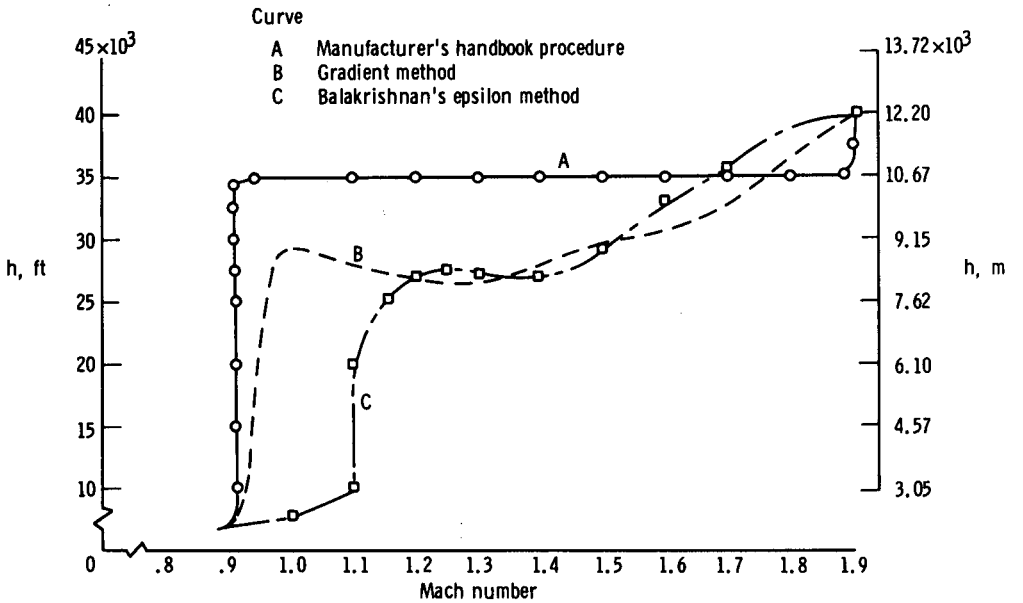


Figure 9. F-104 computer-generated minimum time profiles.

DIGITAL COMPUTING ASPECTS OF THE JAGUAR NAV/ATTACK SYSTEM

BY

D E HUMPHRIES

AVIONICS DEPARTMENT

ROYAL AIRCRAFT ESTABLISHMENT

SUMMARY

The Jaguar nav/attack system performs approximately the same task as that in the Harrier aircraft, which is wholly analogue, but does the job using a digital computer for roughly the same cost. Despite the similarity of Harrier system requirements, every endeavour has been made to prevent the system from becoming simply a digital copy of analogue ideas, but to approach each requirement from a digital viewpoint. This paper describes the lessons that have been learned and the techniques that have been evolved during its development up to its present stage of 'Hack' aircraft flight trials. A brief description of the units which comprise the system and the facilities which it provides is given, with particular emphasis on those areas where the change to a digital computer has allowed much greater design freedom and operational flexibility.

1 INTRODUCTION

The navigation and attack system which is being manufactured by Elliott Automation for the British version of the Anglo-French Jaguar aircraft, represents a particularly important milestone in the application of digital computers into the avionics systems for UK military aircraft. The system for the Jaguar performs approximately the same task as that in the Harrier aircraft, which is wholly analogue but uses a digital computer and yet costs roughly the same amount. Despite the similarity of the requirements however every endeavour has been made to avoid making simply a digital copy of analogue techniques but to approach each requirement from a digital viewpoint.

1.1 System Description

Fig 1 shows a functional block diagram of the system. Interfacing with the remainder of the system is predominantly analogue, via the interface unit, mainly due to the fact that at this stage of system development, the majority of peripheral sub-systems are still analogue. The E3(R) inertial platform is cluster rotated and the major part of the necessary calculations are carried out in the digital computer. One of the few digital links in the system is in the transmission of velocity information from the platform electronics unit into the digital computer in parallel binary form. The interfacing requirements, in particular the signal forms have been dictated largely by the characteristics of the peripheral equipment, but where possible the preferred form is analogue voltages which lend themselves particularly well to simple and accurate digital conversion especially when the voltage comes from a potentiometer energised by the Interface unit. There are however a number of areas where the change to a digital computer has allowed a considerable improvement in both facilities and operational flexibility over that attainable by analogue means and it is proposed to describe three areas in which these advantages have been apparent.

2 MODE SELECTION

In an analogue system complicated relay logic is required to convert a limited number of mode selections available to the pilot, into the necessary switching actions to connect the circuit elements appropriate to the mode. However, consideration of the problem from a digital viewpoint indicates that a fresh approach might well yield dividends. By the use of push buttons representing yes/no answers to a limited number of questions, simple selection of a wide range of combinations is possible. Such a push button control system is illustrated in Fig 2.

During the development of this programme it became clear that a flow diagram for the weapon aiming programme could be constructed whose decision structure was largely controlled by the settings of the push buttons, resulting in a simple yet effective steering programme with straightforward logic.

The Weapon Aiming problem can be effectively divided into two main blocks, micronavigation to determine the relative position of the target to high precision, and ballistics to calculate the flight of the weapon. The weapon aiming problem then reduces to one of bringing the end point of the ballistic flight into coincidence with the target, and initiating a release. There is little variation in the sub-routines necessary for the ballistic calculation, other than to select the necessary ballistic parameters in response to the actual weapons selected, but it is in the choice of method of determining the target position that a great deal of flexibility is possible, and it is largely over the micro navigation part of the weapon aiming programme that the push button panel exercises control. The process of selection may be summarised under the following headings.

2.1 The Initial Starting Conditions

The three relative position coordinates of the target position can either have as their starting conditions the values given by the navigation system, (a planned target), or an arbitrary set of values, based on the most likely relative position for an unexpected target to be sighted. (Target of Opportunity), for example a few miles ahead and the radio altimeter height below. This choice is made by Buttons A and B.

Alternatively, the starting conditions can be derived by adding a precisely known offset in each coordinate, following a preliminary micro-navigation run against an easily visible fixing point, this choice is made by Button C.

2.2 Response to Updating Inputs

Once the pilot identifies the target, he can begin to correct the relative coordinates using the display sight, and in addition inputs from the radio altimeter, or barometric height unit are available if required. The main alternative options open are:

a. Accept the depression angle of the target from the pilot via his display, and scale the triangle using either Barometric or radio height. This selection is made by Button E.

b. Alternatively, start using this method but once the marker has been laid over the actual target, refine the slant range by comparing the sight line spin rate calculated using computed range, with the actual spin rate, and then modifying the range until the spin rates match. A process termed Kinematic Ranging which is selected by Button F.

Of course the whole of the micro-navigation process can be ignored, and the results of the ballistics calculations displayed on the sight giving an impact point predicting mode, allowing the pilot to judge coincidence with the target and initiate release. This mode is selected by Button D.

It is clear even from this brief description the way in which computing routines can be directly controlled by the push buttons, to give a very wide range of options through a simple set of controls.

3 VERTICAL VELOCITY COMPUTATION

Although Barometric height rate, and vertical acceleration from an inertial platform, provide complementary data which can be mixed to provide accurate vertical velocity information, the practical realisation of an effective mix by analogue means has proved difficult.

In level flight accurate height rate can be obtained from barometric data, and this can be used to back off any accelerometer bias and nul the accelerometer output.

In order to take full advantage of the accelerometers ability to detect rapid changes in velocity, this loop must have a long time constant, so that short term information is derived from inertial data, and long term from Baro data. A typical analogue loop for this purpose is shown in Fig 3. This scheme works effectively in level flight and for short rapid climbs and dives, since over short intervals errors in barometric height rate resulting from diving or climbing through a non-standard atmosphere have little effect on the accelerometer bias, and the accelerometer information is believed. In long slow climbs or descents, however, even the long loop time constant is unable to prevent erroneous bias correction terms being applied to the accelerometer output with the result that at the end of a long slow change of height through a non-standard atmosphere a large height rate error can accumulate. Worse still, because of the long time constant of the loop, a long period of level flight is required to recover the correct value.

A simple expedient is to use the vertical accelerometers in an open loop configuration during height changes but since in this arrangement errors are not only unbounded but divergent, over long periods in open loop, large errors still build up, resulting in transients when the loop is closed again, and once again a slow recovery.

Studies were therefore undertaken at RAE to investigate the possibility of using Kalman Filtering techniques for this mixing, since it was clear that a variable gain system was required (Reference 1).

Developing the model, however, led to a much clearer insight into the nature of the problem, and the gain behaviour of the filter indicated clearly the ways in which a simpler system could still be made much more effective than earlier ones. The important point is that since there are two sources of error, accelerometer error, and deviations of the atmosphere from ICAN, this should be recognised in any mixer design, and rather than feed the error signal always to correct the same parameter, to use it to correct the more likely error source, which will vary according to the state of the flight, particularly the aircrafts vertical velocity.

If one considers an aircraft in level flight, clearly the gain of the original systems are of the right order since any steady divergence in vertical velocity must be due to incorrect accelerometer bias, or backing off of the earths "g" components. However, immediately upon entering a dive, any immediate deviation between the two sources, is likely to be due to Barometric error and the correction should be applied therefore to the Baro input, and the loop providing corrections to the accelerometer open circuited. This system still does not prevent errors due to the inertial velocity divergence from building up, but if a high gain is used to force the inertial rate to agree with the barometric vertical rate as soon as level flight is regained, at least the recovery is more rapid, and the magnitude of the resulting transient greatly reduced. A system like this which is shown in Fig 4 represents a considerable improvement over previous analogue loops but it is perhaps worth pointing out the additional advantages which the true Kalman Filter can achieve.

There are two main improvements, firstly it makes use of the baro-corrections estimated in the early stages of the dive, to calculate the percentage deviation of the atmosphere from standard, which it uses later in the descent when otherwise both baro and inertial information are unreliable. Secondly, because of statistical correlation estimates which have been built up about any height errors arising as a result of incorrect velocity information, once level flight is regained, corrections are not only applied to the velocity, but also to the height error, so that an extremely accurate estimate of true height is also available from this system . The expected performance of such a Kalman Filter Baro/inertial loop, derived from the mathematical models, is shown in Figs 5 and 6, which are extracted from Reference 1. It is expected that the sub-optimal system will approach this performance in the computation of vertical velocity, although it will not be able to provide the remarkable height error recovery which is exhibited by the true filter. This, however, can be compensated for by building into the system the facility for height fixing while passing over level ground of known height above sea level. In this process the height of the ground which has been stored along with other mission data in the computer is subtracted from the Barometric height, and the result compared with the value received from the radio altimeter, and when the pilot operates the "Height Fix" button the baro-datum is reset to bring the two into agreement. Thus if the height of target above sea level is also stored in the computer, the difference between computed aircraft height and target height can be used to set the initial value of the vertical relative position coordinate to an acceptable accuracy as described in section 2.

4 IN FLIGHT DATA INSERTION

In the past, the nature of analogue computing in aircraft systems has meant that the main method of altering a particular parameter was by means of "slewing". In this method a variable rate control was provided for the operator, by which he could control the rate of increase or decrease of the quantity until its value as indicated on a dial or counter indicated the desired value. Since in general

changing parameters involves the physical movement of a shaft it is unlikely that there is any possibility of instantaneous change to a new value. With digital computing, however, this limitation no longer applies and the possibility of instant change is available. In the Jaguar system initial thoughts tended towards a digital version of the "slewing" system. In this system a binary digit was added to or subtracted from the existing stored number at a fixed rate whenever the slew control was operated. The digit at which this increment was added depended on the amount of rotation of the slew control, small movements causing the increments to the least significant bit of the number, and increasing rotation causing the increments to be added further up the number towards the most significant end. This resulted in a logarithmic "slew" control which it was thought would be easy to operate. In practice, however, a number of significant disadvantages became apparent, mainly due to the fact that the conversion to decimal took place "downstream" of the slewing operation, so that the slewing process was in binary, while the display was in decimal, and since the conversion to decimal was done at a relatively slow repetition rate, decimal digits which were changing very rapidly appeared as random numbers which were clearly readable and which made assessment of the value reached very difficult. It was obviously possible by the use of software to make the behaviour represent more closely the characteristics of analogue "slewing", but rate "slewing" is not an ideal method for data insertion and therefore advantage was taken of a re-appraisal to look at some alternative schemes which might prove both easier to implement and operationally more effective. The normal method for real time data insertion to a digital computer is by means of a keyboard, and there is no doubt that for rapid transfer of information when the operator can concentrate wholly on the task the keyboard is ideal. In a single seat strike aircraft, however, this is by no means the situation. During high speed, low level, flying, the pilot cannot afford to divert his attention into the cockpit for more than one or two seconds at a time, so that the keyboard, with its demand for visual identification of the keys, for effective operation, becomes less attractive. In addition, its operation by a gloved hand, in conditions of considerable buffet is likely to prove difficult unless some form of hand or wrist support is provided. If an error is made, the whole number has to be reinserted, as it has even if only the last two digits need to be changed. One further constraint which was provided by the Jaguar aircraft, was that as in any single seat aircraft, cockpit space within easy view and reach of the pilot was at a premium and to provide space for a keyboard with sufficient key spacing for easy operation would have been extremely difficult and would have needed quite a radical re-design of the cockpit, particularly if the hand support for buffet conditions was also provided.

The scheme which has been devised requires one extra push button to be provided on a hand controller which is already present for operating the weapon aiming system. A sketch of the modified hand controller is shown in Fig 7, the important controls for this purpose being the "digit advance" push button, operated by the thumb, and the "Phase change trigger" operated by squeezing with the first finger. The hand grasps the barrel of the controller so that it is fully supported in all directions, and the controller is placed at the pilots left hand, just to the rear of the throttles. The sequence for revising any number is the same but for convenience this description will deal with changing the stored latitude and longitude of a target or navigation way-point.

i The present stored value of latitude and longitude are displayed in the two sets of optical indicators by operating the look up control and way-point selector.

ii If it is desired to change either or both of the stored values then a push button under the appropriate window is pressed and the most significant digit of that number begins to flash at a slow rate indicating that the digit advance control is operative on that digit.

iii The first press of the 'digit advance' control sets the digit to 1, the second to 2 etc so that n presses sets the digit to read n.

iv The phase change trigger is then squeezed and control passes to the next most significant digit which then begins to flash, and the process repeated, until the whole number is reset. If it is wished to leave a digit unaltered then a second press of this phase change trigger without operation of the 'digit advance' will pass control to the next digit leaving the previous one unaltered.

v The number is then checked to see that it is correct and if it is, an 'insert' button is pressed and the new value is stored in the computer.

vi This process can then be repeated for the other stored value.

Although when described fully as it is above this process sounds very laborious, practical experiments have indicated that it is very easy to learn, and speedy operation comes naturally, so that the time taken to insert complete data about each way-point is about 20 seconds, but more important it is possible to feed in a complete latitude, longitude, and target height in the air, with the only time spent head down being the checking time necessary to ensure that the right number has been set up before inserting.

It is clear from the above that the requirements for rapid data insertion into the system prior to the mission, are not compatible with the requirements for modification of the data during the flight which can result from changes in the tactical situation during the mission. Removed from the constraints which were imposed by the Jaguar requirements it is possible that a more effective system than that described above could be devised, but even so such a system would have to be orientated towards minimum 'head-down' time, and it is unlikely that this would be compatible with the speed required for mission data insertion. It seems therefore that in future digital systems will need to have an additional means of rapid data insertion. Such a facility could be provided by a simple card reader into which a card which had been prepared at the pre-flight briefing could be inserted which would rapidly enable all the mission data to be inserted in the shortest possible time.

5 CONCLUSION

The three examples quoted above, are by no means the only areas in which the change to a digital system have allowed much greater freedom and flexibility than that attainable with an analogue system and yet at no greater cost for the whole system. But they have been chosen as examples because they represent the dividends that can be reaped by adopting a fresh approach to old problems and starting the design of a digital system from a clean sheet of paper rather than from the stand-point of an existing analogue system even though the system is required to fulfil the same functions. It is hoped that in the same view, when it comes to the next generation of digital avionic system we will not allow schemes devised within the constraints of the Jaguar or other earlier digital systems to influence the design, but to take as much advantage as possible of the extra freedoms that recent advances will allow.

REFERENCES:

Ref 1 Application of Kalman Filtering to Baro/Inertial Height Systems.
Royal Aircraft Establishment Technical Report No 69121.
P M Barham and P Manville.

ACKNOWLEDGEMENT

The work described in this paper was carried out in close collaboration with Elliott Automation (Inertial Navigation Division) whose assistance is gratefully acknowledged.

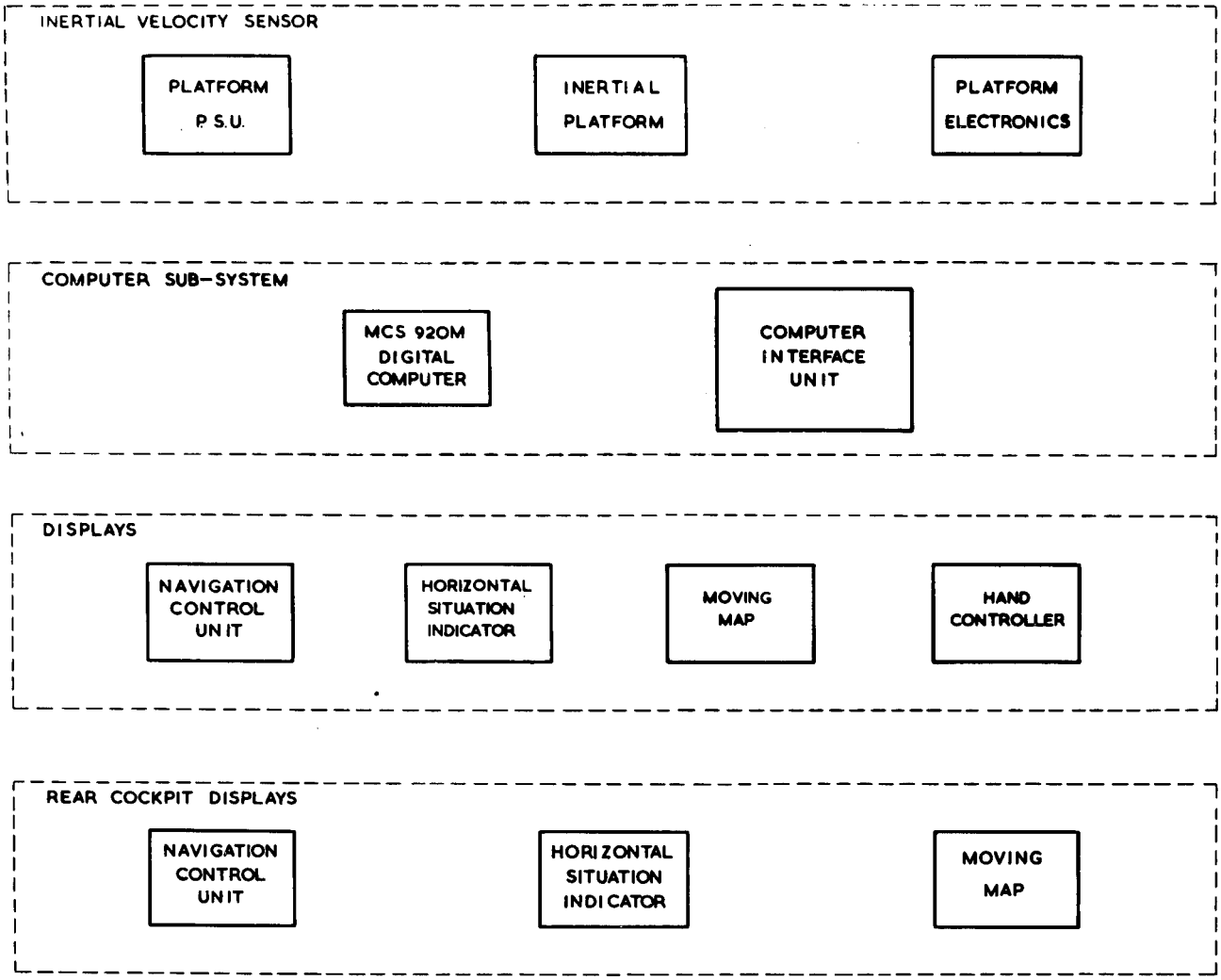


Fig. 10.1 Jaguar navigation and attack system

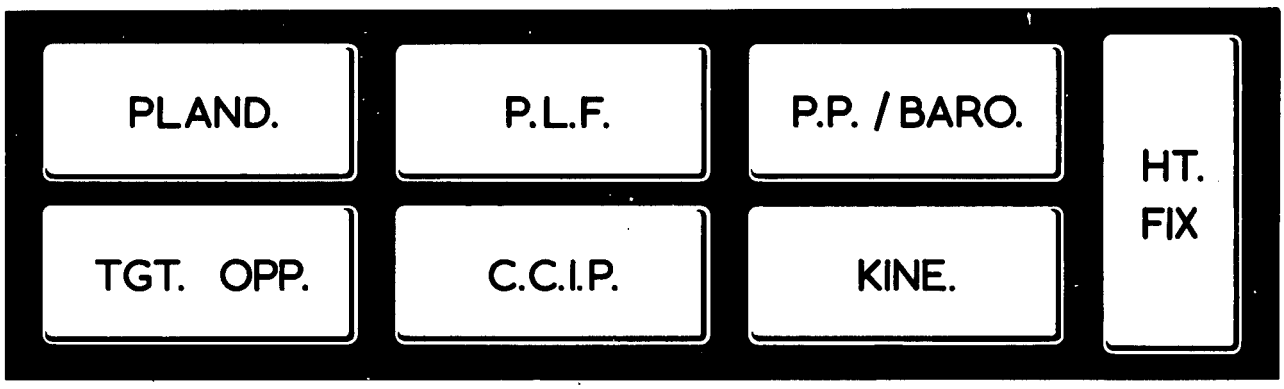


Fig. 10.2 Aiming mode selector

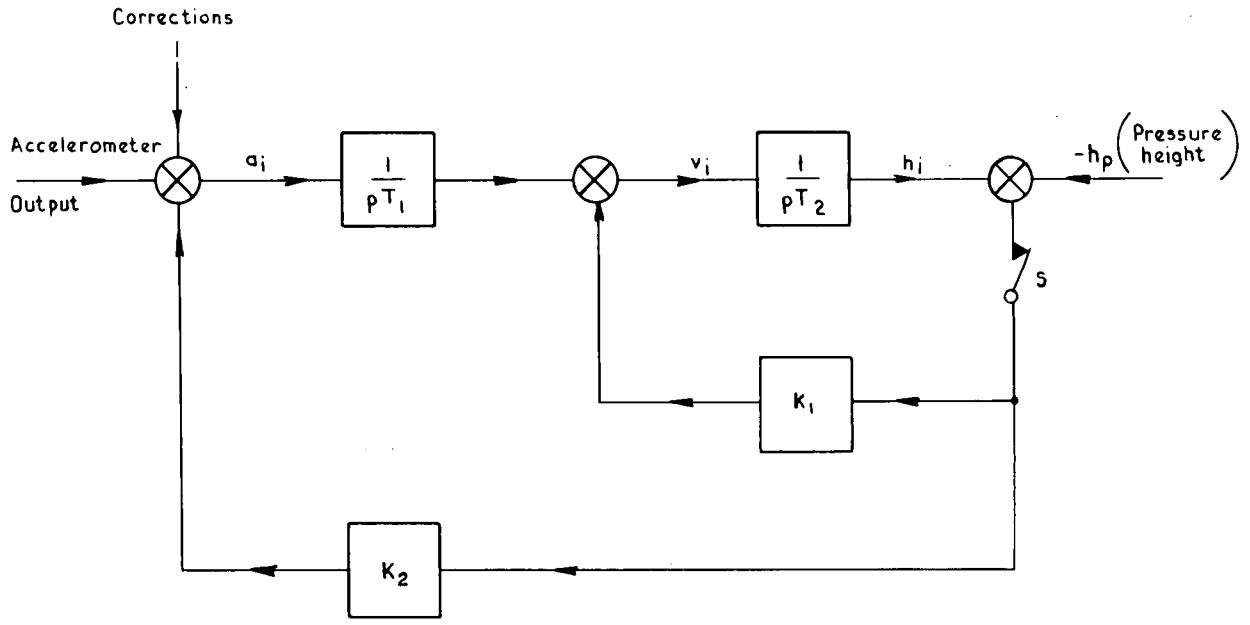


Fig. 10.3 Basic baro/inertial mixing height system

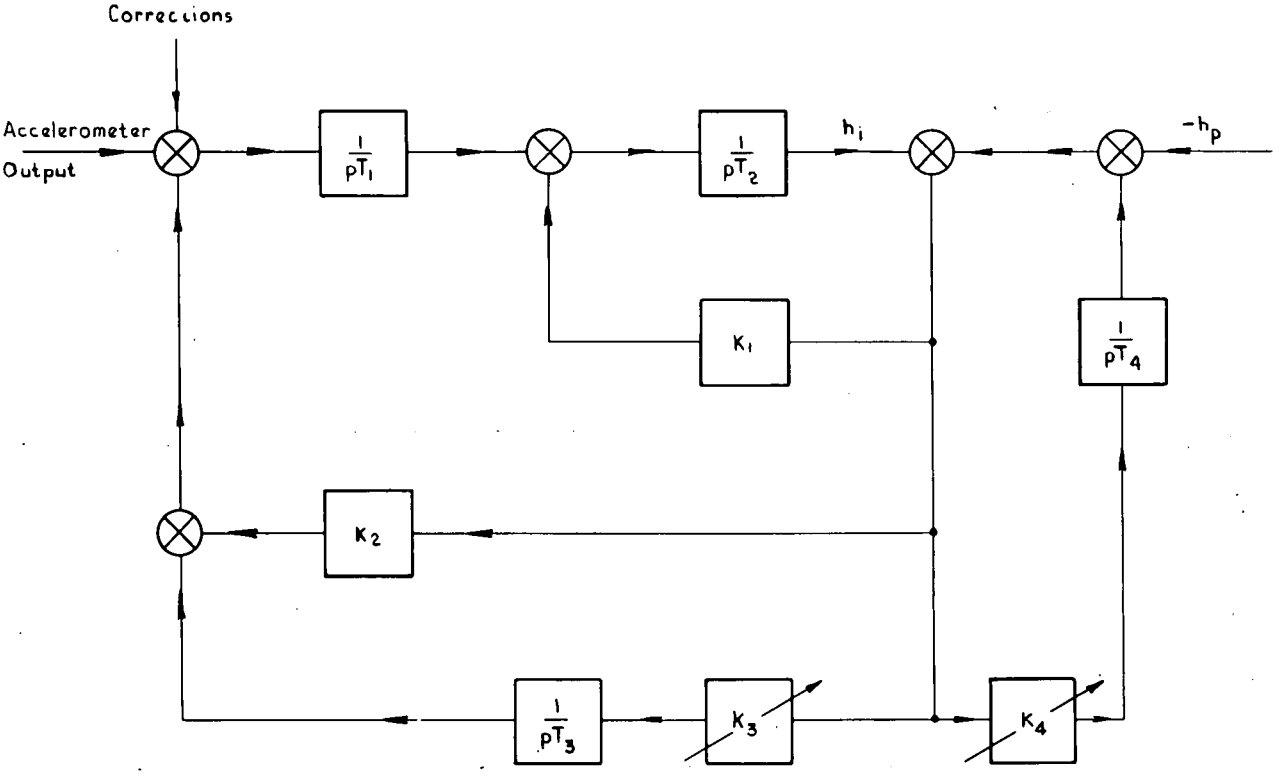


Fig. 10.4 Height system with barometric height correction

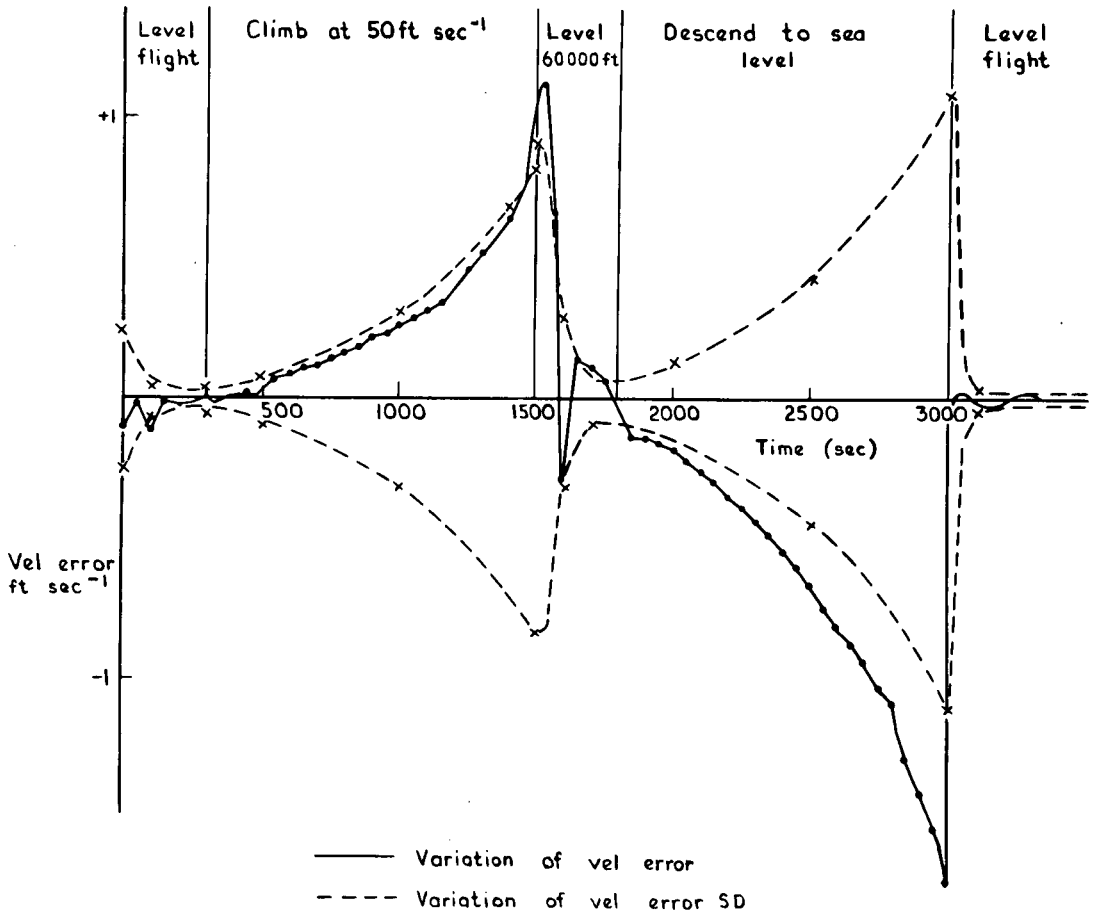


Fig. 10.5 Vertical velocity error during 50 ft/sec climb/descent

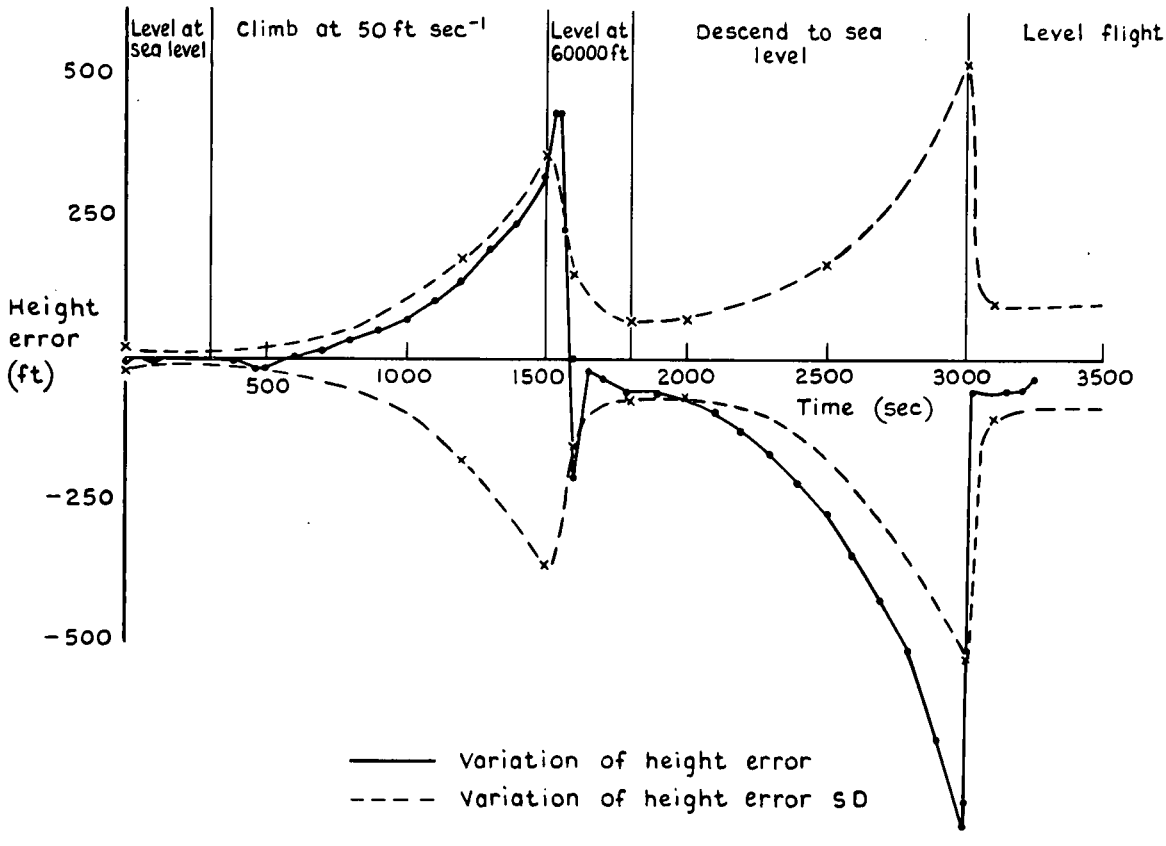


Fig. 10.6 Height error during 50 ft/sec climb/descent

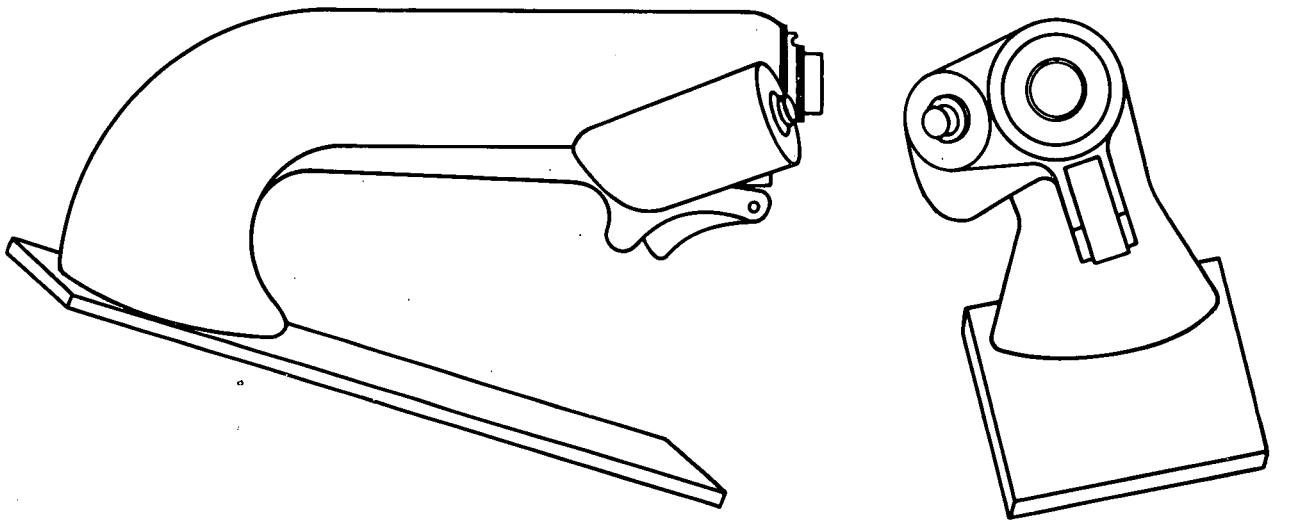


Fig. 10.7 Hand controller for data insertion

The Effect of Digital Computer Limitations and Sensor Errors on the Accuracy of an Automatic Bomb Release

Gerhard Schroeter

Industrieanlagen-Betriebsgesellschaft mbH
Ottobrunn, Germany

Summary

Onboard computer and attitude/air data sensor errors - among others - cause delivery errors of automatically released bombs.

For a retarded bomb it is shown by numerical evaluation that with a uniform distribution of release conditions the computing error (= difference between true value and the estimate, which is won from a polynomial) is approximately normally distributed with mean value zero and standard deviation σ, σ being dependent on the degree of the polynomial and the width of the admissible interval of release conditions.

Besides, formulas are derived for the estimation of the influence of the inaccuracy of weapon computer input data on the impact point. These formulas can be applied to derive the requirements which attitude/air data sensors should satisfy to provide for sufficiently accurate bombing. An example is given.

1. Introduction

If a bomb is to be automatically released, the pilot initiates a command that starts the weapon release computations at the instant when the target symbol on his head-up display matches the target.

Instantaneously, the position of the aircraft relative to the target is determined by means of sensors providing data on slant range and sight angle (angle between line of sight and horizontal). Simultaneously the momentary bomb range x_1 is calculated from the height, velocity and dive angle, and the difference a between the bomb range x_1 and the horizontal distance to the target x_2, i.e.

$$a = x_1 - x_2 \tag{1}$$

is determined. By means of the navigation system, the position of the aircraft is continuously monitored and automatically brought up to date, thus the value a also can be determined continuously. If a equals zero, the bomb is released automatically. A prerequisite that has to be met is that the pilot flies the aircraft in a vertical plane which contains the target.

There are two principal sources of error associated with the nature of the automatic bombing method. These are

- computational errors resulting from an approximated solution of the differential equation of the bomb range because of limited onboard computer storage capacity, and
- sensor errors which cause 1st inaccurate estimates of the release conditions, thus causing differences between actual and calculated bomb range, and 2nd a wrong determination of the horizontal distance between aircraft and target.

The influence of these errors onto the delivery error will be outlined in the next paragraph.

2. Error Analysis

2.1 Computer Limitations

It can be shown that the bomb trajectory in a homogeneous environment (no wind) satisfies a differential equation with release height h, release velocity v and release angle ψ (i. e. the angle between flight direction and a horizontal at release) as initial conditions. The differential equation has to be solved numerically by the onboard computer in sufficiently short time.

Methods to solve this equation in an adequate manner are not to be described here, but it is assumed that the bomb range x_1 can be found by evaluating a polynomial of limited degree with the release conditions as independent variables:

$$x_1 = a_0 + a_1 \cdot h + a_2 \cdot v + a_3 \cdot \psi + a_4 \cdot h^2 + a_5 \cdot v^2 + a_6 \cdot \psi^2 + a_7 \cdot h \cdot v + \ldots \tag{2}$$

The average accuracy should be the better the more terms are included in the polynomial. If a series of points of solution has been calculated previously by some standard routine of numerical analysis, the coefficients a_i can be obtained e.g. by the method of the least sum of squares. Once the coefficients a_i are calculated they can be stored in the onboard computer, and the bomb range can be determined according to Eq.(2).

If we consider e.g. a hypothetical retarded bomb with drag coefficient $c_w = 2$ and cross section/mass ratio $F/m = 0{,}002\ m^2/kg$ we get different polynomials depending on the number n of terms in the sum and the interval of validity of the approximate solution. If e.g. the interval of admissible release conditions is given by

$$\begin{aligned} &100\ m \leqslant h \leqslant 500\ m, \\ &100\ m/sec \leqslant v \leqslant 300\ m/sec, \\ &0^\circ \leqslant \psi \leqslant 30^\circ, \end{aligned}$$

which provides high flexibility in the selection of attack modes we get for $n = 20$ a polynomial of third degree in h, v, ψ with a standard deviation σ of error of about 8 m, if we assume that the release conditions are at random uniformly distributed over that interval. It can be shown, that the error then fits approximately a normal distribution with zero mean and standard deviation σ.

If we only admit release conditions which are confined by

$$\begin{aligned} &100\ m \leqslant h < 300\ m, \\ &150\ m/sec \leqslant v \leqslant 250\ m/sec, \\ &0^\circ < \psi < 15^\circ, \end{aligned}$$

the standard deviation of error is only about 2 m, if the polynomial has 20 terms. On the other hand a standard deviation of error of approximately 8 m is reached with $n = 5$ if only the smaller interval of release conditions is admitted. These figures show clearly how reduction of military flexibility reduces either the error at a given storage capacity of an onboard computer or the requirements as to the storage capacity.

2.2 Sensor Errors

As to the sensor errors, their influence can be evaluated separately for the bomb range and for the distance to the target. If Δh, Δv, and $\Delta\psi$ signify the errors of h, v, and ψ, respectively, we can determine the deviation of the calculated range from the true range approximately according to Eq.(3):

$$\Delta x_1 = \frac{\partial x_1}{\partial h} \cdot \Delta h + \frac{\partial x_1}{\partial v} \cdot \Delta v + \frac{\partial x_1}{\partial \psi} \cdot \Delta \psi \qquad (3)$$

In the same manner we get the error Δx_2 from the horizontal distance x_2 between target and aircraft which is given by

$$x_2 = l \cdot \cos\varphi$$

where l is the slant range, and φ the angle between the line of sight and the horizontal:

$$\Delta x_2 = \Delta l_1 \cdot \cos\varphi - \Delta\varphi \cdot l \cdot \sin\varphi .$$

Here

$$\Delta l_1 = \Delta l + \Delta l_2,$$

Δl being the error of the sensor for slant range, and Δl_2 being an error due to a wrong point of reference resulting from inaccurate determination of the sight angle φ.

If we introduce into Eq. (3)

$$h = - l \cdot \sin\varphi$$

$$v = \frac{v_g}{\cos\psi}$$

where v_g = ground speed, and regard that for the total delivery error because of Eq. (1)

$$\Delta a = \Delta x_1 - \Delta x_2$$

holds, we finally get

$$\Delta a = b_1 \cdot \Delta v_g + b_2 \cdot \Delta\psi + b_3 \cdot \Delta\varphi_1 + b_4 \cdot \Delta l + \varepsilon$$

with

ε = computational error

$$b_1 = \frac{\partial x_1}{\partial v} \cdot \frac{1}{\cos\psi}$$

$$b_2 = \frac{\partial x_1}{\partial \psi} + \frac{\partial x_1}{\partial v} \cdot v \cdot \tan\psi$$

$$b_3 = - \frac{\partial x_1}{\partial h} \left(x_1 + \frac{h^2}{x_1}\right)$$

$$b_4 = \frac{1}{\sqrt{x_1^2+h^2}} \left(\frac{\partial x_1}{\partial h} \cdot h - x_1\right) .$$

Here $\Delta\varphi_1$ is the error of the angle φ_1 between the longitudinal axis of the aircraft and the line of sight.

If the distribution of the release conditions (h, v, ψ) is known,

the distribution of the b_i's and ε can be found by sampling. It can be shown that the variance of Δa is a linear combination of the variances of the sensor and computer errors.

Assuming that the expected values of the sensor errors equal zero, and that

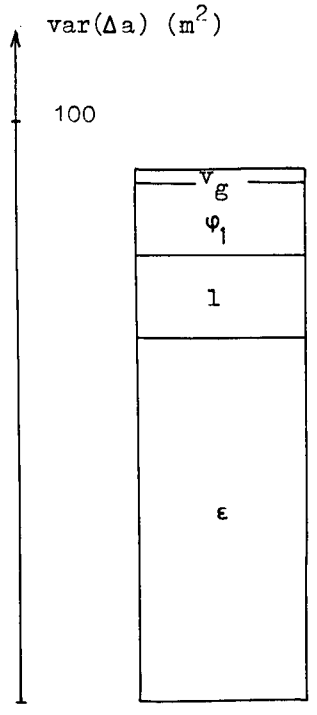
$$\sigma_{\Delta l} = 7{,}5 \text{ m}$$

$$\sigma_{\Delta v_g} = 1 \text{ m/sec}$$

$$\sigma_{\Delta \psi} = 0{,}002$$

$$\sigma_{\Delta \varphi_1} = 0{,}005 \text{ ,}$$

and that the release conditions are distributed uniformly over the larger of the previously mentioned intervals, we get the break down of the variance of Δa into the individual terms which is shown in the graph.



In this graph the height of the blocks represents the size of each influence, showing that the computational error is by far the main source of delivery errors - of course only under the above assumptions as to the storage capacity and the computational method.

This example took into consideration only 4 sensors; if additional ones, such as the air data sensors, are regarded, analogue formulas and results hold.

Symbols and Abbreviations

x_1	horizontal bomb range
x_2	horizontal distance aircraft-target
l	slant range aircraft-target
a	x_1-x_2
h	release height
v	longitudinal velocity at release
ψ	dive angle at release
φ	angle between line of sight and a horizontal at release
φ_1	$\varphi - \psi$
v_g	ground speed
ε	computational error
a_i	coefficients of the polynomial for x_1
σ	standard deviation of ε
n	number of terms in the polynomial for x_1
$\sigma_{\Delta l}$	standard deviation of the error of the sensor for slant range
$\sigma_{\Delta v_g}$	standard deviation of the error of the sensor for ground speed
$\sigma_{\Delta\psi}$	standard deviation of the error of the sensor for dive angle
$\sigma_{\Delta\varphi_1}$	standard deviation of the error of the angle between line of sight and longitudinal axis of the aircraft

RECENT ADVANCES IN SELF-ORGANIZING AND LEARNING CONTROLLERS FOR AERONAUTICAL SYSTEMS

by

Cecil W. Gwinn* and Roger L. Barron†

*Physicist, Avionics Laboratory
Air Force Systems Command, United States Air Force
Wright-Patterson Air Force Base, Ohio, USA

†President, Adaptronics, Inc.
McLean, Virginia, USA

RECENT ADVANCES IN SELF-ORGANIZING AND LEARNING CONTROLLERS FOR AERONAUTICAL SYSTEMS

Cecil W. Gwinn and Roger L. Barron

1. BACKGROUND IN CYBERNETICS

What kind of control system, if hooked up backward, having no *a priori* knowledge of this fact, can discover that it is, rewire itself, and work properly? This is part of the more general question: What kind of a system, if put together more or less at random, can reorganize itself to be stable to some given set of environmental conditions? A system which displays this property has been called variously a self-organizing or learning system. Such systems and some of their aeronautical applications are the subject of this paper.

The only systems we know at present that display this reorganization property to a marked degree are living systems. The study of self-organization and learning, among other aspects of living systems, for the purpose of building non-living machines displaying similar characteristics, is included in a branch of Cybernetics called Bionics. The philosophical background of the science of Bionics is discussed by one of the authors in Reference 1.

The exploration of these ideas properly began with the paper "Behaviour, Purpose, and Teleology" by N. Wiener, W. Rosenbleuth, and J. Bigelow in 1943 (Reference 2). Therein was shown the possibility of treating such teleological notions as, "goal", "purpose", "evolution", etc., in a quantitative manner. Philosophers had been debating these terms for centuries, and it was quite startling in the context of the times to be shown that a quantitative treatment was indeed possible. As an outgrowth of this way of thinking, Wiener in 1948 founded the science of Cybernetics (Reference 3), meaning, in his words, "the study of control and communication in animal and machine".

Wiener's development of this theme was couched in a highly abstract form for the times and not easily accessible to the engineer. His ideas were put at the disposal of the engineer by the British psychiatrist, William Ross Ashby, in 1952 by the publication of the book *Design for a Brain* (Reference 4) and in 1956 with the treatise *Introduction to Cybernetics* (Reference 5). Ashby demonstrated in these books, particularly the first, that it is possible, working from purely cybernetic ideas, to develop machines that would show such peculiar animal-like characteristics as purpose, goal, and survival potential in an *a priori* unknown environment. It is important to note that a brain was not being constructed, but rather a machine which exhibited behaviour which is presumed to be caused, in animals, by a nervous system.

Ashby amplified the definition of Cybernetics as follows: "Many a book has borne the title 'Theory of Machines', but it usually contains information about things, about levers and cogs. Cybernetics, too, is a 'theory of machines' but it treats not things but ways of behaving. It does not ask, 'What is this thing?' but, 'What does it do?' It is thus functional and behavioristic!" In the authors' opinion there has been no end of confusion caused by misunderstanding this point of view. To cite an example, there did not exist at the time Ashby wrote and not to any great degree today an accepted explanation, at the physiological level, of such operationally observed and defined (by experimental psychologists) behavioral parameters as learning, recognition, etc. This is not to say that such knowledge would not be desirable, but the Cyberneticist is forced today to define, to paraphrase Eddington (Reference 6), his "neuron or nerve cell as a conceptual carrier of a set of psychological variates". The point of Cybernetics, then, is the explicit possibility of quantitatively treating such behavioral parameters independently of the mechanisms presumed to give rise to them.

A number of scientific people, including the present authors, were caught up in the ferment of these ideas. The US Department of Defense became interested, and in the period 1958-60, through one of its elements, the US Air Force, initiated the Bionics (another word for applied Cybernetics) program. We will not go into detail on the various types of learning and self-organizing machines studied since this period. One type, known as a probability state variable (PSV) machine, was ultimately selected for intensive development. In June 1960 one of the present authors suggested to the Air Force the use of this particular machine in control applications. It was suggested that a period of five to ten years of theoretical and development work would realize this specific application. Theoretical work was initiated in 1961 and a laboratory prototype of the controller was first constructed in 1964. This paper will deal with the self-organizing controller, some of the results of flight testing of an elementary version of this controller during 1969, and certain aspects of the design for a self-organizing control system for on-line optimization of thrust specific fuel consumption in turbopropulsion systems.

We shall first define some terms as they are now coming into use in Bionics.

To Control:

To force and/or steer toward accomplishment of required actuator outputs. It is implied that there is a set of criteria to be improved or optimized in some sense for efficient control. This optimization may be by incorporation of learning or self-organizing system functions and/or by *a priori* design. A complete control system may involve input and feedback channels that embrace adaptively-trained models for prediction and other functions.

To Adapt:

There are two uses of this word which should be, but are often not, carefully distinguished. Consider a system $f(t)$ with output $c(t)$ and input $r(t)$. Suppose, however that output $c'(t)$ is preferred, given input $r(t)$. There are only two ways that this can be accomplished. (1) It is known that the system $f(t)$ would produce output $c'(t)$ if supplied with some input $r'(t) \neq r(t)$. Therefore the desired output can be obtained by altering the input $r(t)$ until it becomes identical to $r'(t)$, using this changed variable as the input to the system $f(t)$. This changing of the input is usually brought about by a feedback loop generating an error signal steering the system toward the desired response. It is the only case that can occur in linear systems containing feedback loops. (2) It is known that there exists a system $g(t)$ that, given input $r(t)$, will produce the desired output $c'(t)$. Therefore the system $f(t)$ is itself operated upon, causing a change in its transfer function in a way that produces the preferred output $c'(t)$ in the presence of input $r(t)$. All systems to be discussed as adaptive in this paper will, unless otherwise indicated, fall under case (2). On the face of it, this definition seems a rather drastic choice, as it inevitably makes the adaptive system nonlinear. For a full discussion of this distinction and its implications see Reference 7.

Learning:

Given a specified operational situation and specified criteria related to performance in this situation, learning is defined as statistical improvement of performance according to the criteria. A system which exhibits this characteristic can have the additional ability of spontaneously initiating transfer functions. Because memory is an inferred part of this definition, the distinction has arisen between learning systems with very short term memory, called hereafter, self-organizing systems, and those with long term memory. Long term memory can be up-dated with experience to improve system "models" of controlled-process dynamics, optimum operating points, and boundaries. The distinction between terms of memory arises only as a function of the properties, real or inferred, of the specified environmental situation. The bulk of this paper will be in the discussion of applications in which the environmental situation can be approached with short term, or self-organizing, learning systems. However the technique is not limited to only this situation.

Random Generator:

This is a term, while not of the same nature as the preceding terms, which will be so much a part of the following that it is advisable to define it at this point. A random generator is taken to mean a subsystem which generates interrelations in the rest of the system, such that these interrelations cannot be predicted based on information derived solely from parameters of that part of the system which excludes this subsystem.

There are two main areas to be considered in the design of any control system. One of these relates to the characteristics of the environmental situation, called the "plant" in control literature. The other relates to necessary properties of the controller given the environmental situation.

The characteristics of the environmental situation or plant might be summed up in a number of questions:

- Is it linear or nonlinear?
- Is it of low order or high order?
- Is it stationary, quasi-stationary, or non-stationary?
- Does it contain significant transport delays?
- Is there interaction or cross-coupling between the actuators that drive multiple response variables?
- Does the coordinate system of the sensors undergo rotation with respect to the coordinate system of the actuators?*

We must carefully distinguish in all these questions between our verbal and/or analytical description of the environmental situation and the "real" environmental situation. Engineers quite often make linear approximations to situations known to be nonlinear. Such approximations may be useful in one regime of the situation, but when one attempts to extrapolate them to another regime he finds them totally inadequate. Again one often assumes stationarity when at most quasi-stationarity exists. The gist of this is that too often the environmental situation is a construct reflecting limited analytical abilities rather than the more unconstrained actual situation.

---

* "Rotation" has evident physical interpretation in control of vehicles, but the mathematics of rotation are equally applicable to treatment of other multivariable processes.

Presuming however that an environmental situation is given, artificially constrained or not, what characteristics are commonly sought for in the controller? A list of some, by no means complete, would be:

- rapid real time convergence to solutions,
- minimum amount of hardware consistent with environmental situation complexity,
- minimum amount of information processing consistent with environment situation complexity,
- reliability.

Other desirable but generally not sought for characteristics would be:

- minimization of the needed amount of analytical description of the environment situation,
- insensitivity to sensor noise,
- ability to deal with *a priori* unplanned for eventualities,
- search of transfer function space rather than input function space to give desired control response.

With self-organizing and learning control systems, except for the elementary self-organizing controller described in Section 3, the approach is to use an automatic search for the system parameter values that give best performance. This search in parameter space is directed by a means for performance assessment. The performance goal generally is to achieve and maintain the minimum levels of system response error that are consistent with available control resources. Subsidiary goals may relate to observance of operating boundaries (such as the surge boundary in a turbojet compressor).

As discussed in References 9 and 10, the two basic types of parameter search are the *unguided* searches, both systematic and purely random, and the *guided* searches, which use performance trend information to quicken their convergence. The most familiar of the guided searches is the classical steepest descent method. With this method, the gradient of the system performance surface is computed from systematic explorations about the existing operating point and the next step is conducted in this gradient direction. The past decade has seen increasing attention given to *guided random* searches. Here, information acquired during the search is used to control the probability distribution of future steps. Two versions of the guided random method are employed. The simpler of these uses constant step magnitudes, the directions of the steps being statistically governed by performance trend data. This method is applicable to self-organizing (short-term memory) systems. In some cases, the step magnitudes are varied as a deterministic function of error levels.

The other, more general, version of the guided random search is a stochastic process in which the parameter vector, $\underline{a}$, at time t is taken in the form

$$\underline{a}(t) = \underline{a}^*(t) + \underline{\underline{C}}(t)\underline{\xi}(t) , \tag{1}$$

where $\underline{a}^*$ represents the mean of $\underline{a}$, usually the "best to date" value from prior experiments in the search, i.e., $\underline{a}^*$ is the current estimate of the best $\underline{a}$; $\underline{\underline{C}}$ is a transformation matrix that is usually modified as information is gained and system performance changes; and $\underline{\xi}$ is a *normal random vector*, often defined for convenience as having a unit covariance matrix. The algorithm characterized generally by Equation (1), with $\underline{a}^*$ and $\underline{\underline{C}}$ being continually adjusted so as to guide further exploration, is a powerful tool in multidemensional, multimodal surface search problems. This search appears to be particularly well suited to off-line and on-line learning of prediction models in control systems.

2. SELF-ORGANIZING CONTROL

Parameter search and performance assessment processes for control systems were discussed by one of the present authors in an earlier AGARD paper (Reference 10). Design of correlation logic units was treated extensively, and it was shown how modular performance assessment and correlation units are combined in self-organizing controller (SOC) networks. The simpler guided random search method is used in these correlation units, that is, search actions pertain only to the question of step directions.

References 10-12 develop the theory of self-organizing control for plants that can be correctly characterized by the vector-matrix relationship

$$\underline{\Delta x}^{(N)}(t) = \underline{\underline{G}}(t)\underline{\Delta u}(t) , \tag{2}$$

in which $\underline{\underline{G}}$ is the gain matrix that transforms a control vector change (Δu) *into an increment* at the N^{th} (highest non-singular) derivative level of the system response vector, $\underline{x}$. Although more general cases are treated, it is shown how the goal of minimum

$$\int |e_p| \, dt ,$$

where $e_p \equiv e + \tau\dot{e}$ and τ is a positive prediction constant, leads one to the following binary *value function* for performance assessment (provided $N = 2$ and e is a one-dimensional control error):

$$v = - \operatorname{sgn} \ddot{e}_p \operatorname{sgn} e_p \, . \tag{3}$$

Consider, now, an illustrative application of the SOC. Figure 1 shows a hypothetical two-degree-of-freedom plant. The goal of the SOC is to position a mass, m, in x-y space. Two coplanar, mutually-orthogonal forces, F_1 and F_2, act on the mass. These forces are produced by the SOC working through fixed actuator gains, K_1 and K_2, respectively. Thus:

$$F_1 = K_1 u_1 \, , \qquad F_2 = K_2 u_2 \, . \tag{4}$$

The most important feature of the plant is that the angle, φ, between F_1 and the x axis is arbitrary and unknown; furthermore, it is assumed that φ cannot be measured by any direct means.

If the mass moves without friction or spring restraint, one has the differential equations

$$m\ddot{x} = F_1 \cos\varphi - F_2 \sin\varphi$$

$$m\ddot{y} = F_1 \sin\varphi + F_2 \cos\varphi \, ,$$

or

$$\underline{\ddot{x}} = \underline{\underline{G}}\,\underline{F} \, , \tag{5}$$

where

$$\underline{\ddot{x}} \equiv \begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} \tag{6}$$

$$\underline{F} \equiv \begin{pmatrix} F_1 \\ F_2 \end{pmatrix} \tag{7}$$

and

$$G \equiv \frac{1}{m} \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} \, , \tag{8}$$

the latter being the undetermined gain matrix of the plant.

The task of the SOC is to manipulate u_1 and u_2 in such a way that the predicted errors

$$\left.\begin{aligned} e_{p_x} &\equiv x_c - x + \tau(\dot{x}_c - \dot{x}) \\ e_{p_y} &\equiv y_c - y + \tau(\dot{y}_c - \dot{y}) \, , \end{aligned}\right\} \tag{9}$$

where τ is a positive constant, are driven to zero regardless of the value of φ.

Alternative SOC configurations for this application are shown in Figure 2. "Configuration A" uses one Performance Assessment (PA) unit and two Correlation Logic (CL) units. The value signal, v, for this controller is

$$v = - \operatorname{sgn} \frac{d^2}{dt^2} \left(|e_{p_x}| + |e_{p_y}| \right) \, , \tag{10}$$

which can also be written in the form

$$v = - \operatorname{sgn} \left(\ddot{e}_{p_x} \operatorname{sgn} e_{p_x} + \ddot{e}_{p_y} \operatorname{sgn} e_{p_y} \right) \, . \tag{11}$$

Configuration A offers simplicity, but it requires that $\ddot{e}_{p_x}$ and $\ddot{e}_{p_y}$ be computed at frequencies as high as $1/2\,\Delta t$, where Δt is the correlation sampling rate. (Typical Δt values are 10^{-3} times $1/f_c$, in which f_c is the 3 dB frequency of the closed-loop system). To relax the bandwidth requirement on PA circuits, it is helpful to subdivide v into separate value signals for e_x and e_y. Doing this, it is also possible to use feedforward of e_{p_x} and e_{p_y} to Actuation-Correlation Logic (ACL) units, as indicated in Figure 2 under the heading "Configuration B". These units are also described in References 10-12. It has been found that Configuration B provides successful SOC operation even if $\ddot{e}_{p_x}$ and $\ddot{e}_{p_y}$ are computable up to only $3f_c$. Thus, despite its greater component count, Configuration B is attractive when the process has high natural frequencies.

The subdivided value signals for Configuration B are computed as

$$v_x = \text{sgn } e_{p_x} \text{ sgn } \ddot{x}_p \tag{12}$$

and

$$v_y = \text{sgn } e_{p_y} \text{ sgn } \ddot{y}_p \, , \tag{13}$$

where

$$\ddot{x}_p = \ddot{x} + \tau' \dddot{x} \tag{14}$$

$$\ddot{y}_p = \ddot{y} + \tau' \dddot{y} \, , \tag{15}$$

in which τ' is positive.

Figure 3 illustrates signal flow within the system using the Configuration B controller. Consider first the case $\varphi = 0$, when u_1 has exclusive authority over x and u_2 has the same over y. The plant gain matrix has zero off-diagonal terms, because the actuators do not interact ($\sin \varphi = 0$). ACL's 2 and 3 should produce zero outputs, on the average, so as not to disturb ACL 1 in its control of x and ACL 4 in its control of y. The signal path for control of y is indicated by the heaviest lines and black elements of the figure.

Next suppose the controlled object rotates so that $\varphi = 90^\circ$. Now u_1 has exclusive authority over y and u_2 over x. The diagonal terms of the plant gain matrix are zero, and control must be established via the off-diagonal terms, one of which is -1 and the other $+1$. ACL 2 now assumes responsibility for x, the output of ACL 2 reaching the x subplant via the u_2 summing device and the $-\sin \varphi$ element of the plant. Similarly, as indicated by the shaded blocks and medium-weight lines in Figure 3, ACL 3 effects control of y via the u_1 summing device and the $\sin \varphi$ element of the plant. ACL's 1 and 4 should produce zero outputs, on the average, to avoid disturbance to ACL's 3 and 2 at the $\varphi = 90^\circ$ condition.

At $\varphi = 45^\circ$, 135°, etc., all ACL paths are equally important in terms of their potential for producing accelerations of the plant. The polarities of the four ACL outputs must then, on the average, reflect the polarities of their respective gain-matrix elements in the plant. A disturbance anywhere in the system requires alteration of the probability states in the four ACL's.

Figure 4 shows typical responses of the illustrative SOC system for the case $\varphi = 30^\circ$, similar results having been obtained for all other φ values and when φ was time varying. Analog computer elements were used to simulate the plant, and commercial SOC hardware was employed for the controller. Inputs (x_c and y_c) to the system are not shown in the recording. Prior to time "a", both x_c and y_c were zero, and at this time x_c changed abruptly by the amount of a positive unit step. Note that the x response is rapid but smooth and that interaction with y is nearly imperceptible. The $x_c = 1$ command was left on until time "c". At time "b", prior to "c", a positive unit step input was also applied to y; y response was also excellent and the interaction with x was minimal. At "c", both x_c and y_c were changed to -1, and the outputs of the system responded quickly and smoothly to the simultaneous command reversals.

From the above, it is seen that Configuration B requires cross-feed ACL's when the controlled plant has significant off-diagonal gain-matrix terms. The best test for the existence of this condition in a plant is to find if mixing of the $\underline{u}$ components occurs within the plant upstream of its acceleration response, $\underline{\ddot{x}}$. (The SOC monitors changes in plant acceleration, any delays due to actuators and other upstream lags being masked by a finite interval between $\underline{u}$ changes). In the absence of such mixing, the cross-feed ACL's may be eliminated.

3. ELEMENTARY SOC

An elementary self-organizing controller for the pitch axis of the US Air Force F-101B fighter aircraft has been successfully flight tested. This controller was flown with a cockpit electric side stick in a pseudo-fly-by-wire configuration, that is, as a fly-by-wire system with a normally disengaged mechanical backup. Blended pitch-rate and forward normal acceleration feedback (C*) and stabilator position feedback were the primary return signals used by the SOC, which had full authority over the aircraft stabilator within the inherent rate and position limits of its actuator. 32 test flights were conducted with the SOC during July-November 1969, constituting total flying time of approximately 40 hours. These flights encompassed nearly the entire performance envelope of the F-101B and included piloting tasks representative of missions flown with current fighter aircraft. The Air Force pilots rated the SOC between A1 and A2 on the Cooper-Harper Pilot Rating Scale. There were no in-flight malfunctions of the SOC equipment.

References 9 – 11 present the technical background for the elementary SOC flight program. Reference 13 outlines design principles followed in the hardware, while Reference 14 details the entire program, including flight test results.

The elementary SOC was unique in that it contained means for modulated-noise control signal generation. A functional block diagram of the SOC Logic Unit (SLU) which incorporated this feature is shown in Figure 5. The SLU received the error signal, e, from an error function coupler and computed an augmented derivative.

$$e_p = e + 0.15\dot{e} ,$$

which was used to provide lead compensation. The sign of e_p was then determined by a zero-crossing detector (electronic relay). Up to this point, the SLU thus functioned as a relay controller with lead compensation. The novel portion of the unit was that which encoded sgn e_p, the output of this first zero-crossing detector, as sgn u, another binary control signal having the same low-frequency components as sgn e_p but with high-frequency components governed chiefly by the characteristics of the noise carrier used in a modulation process.

sgn u was a binary probability state variable (PSV), that is, the distribution function but not the instantaneous state of sgn u was controlled. A probability control voltage, PCV, established the binary probability distribution for sgn u; specifically, the probability of sgn $u = 1$ increased for increasing PCV, while the probability of sgn $u = -1$ decreased for increasing PCV, and the opposite relationships held for decreasing PCV. For its part, PCV was determined by a nonlinear lag operator acting on sgn e_p.

With the particular parameter values used in the lag operator for PCV, the latter was capable of moving from its midpoint to either of its limits (+3V or -3V) in 0.003 sec, or from one of its limits to the opposite in 0.006 sec. Once at either of its limits, PCV biased the noise source to approximately a 3σ level, i.e., sgn u became equal to sgn e_p with almost complete certainty. The minimum elapsed time interval of 0.006 sec between the PCV limits prevented any coherent oscillation of sgn e_p from appearing in sgn u at frequencies above approximately $1/(0.012\pi) \approx 26.5$ Hz. Although the phase shift of the nonlinear lag between sgn e_p and PCV was 180° at 26.5 Hz (the natural auto-oscillation frequency of the SLU encoder working in an otherwise lead-or-lag free negative feedback loop), the phase shift of this element at basic control-mode frequencies (0.5 Hz) was negligible. Therefore encoding of sgn e_p by the modulated-noise process to obtain sgn u had the effect of rejecting sharply any components of sgn e_p above the design cut-off frequency of approximately 26.5 Hz, while producing no substantial effect on basic control-mode signals.

The filtering properties of the PSV encoder were, in this project, of secondary interest to the ability of this device to mitigate adverse control effects of hysteresis in actuators and in the airframe structure. Because the quantitative values of such hysteresis have not been precisely determined, and probably vary from one aircraft to the next, the following discussion considers only certain qualitative aspects.

Suppose some element within the actuator has a hysteresis loop of the type sketched in Figure 6. Suppose, further, that sgn e_p is oscillating in a limit cycle at a given frequency f_{LC}. Without the PSV encoder, the actuator input might then exhibit the waveform shown in Figure 6 below the actuator input-output hysteresis diagram. This input is not a rectangular wave (even though the sgn e_p waveform is rectangular) because of lags and rate limits between the controller and the actuator. For example, in the F-101B, the stabilator main power actuator is driven by a hydraulic servo (the parallel servo) which has a significant output-rate limit as well as first-order and higher-order dynamic lags. For the assumed coherent input, the output of the hypothetical actuator becomes the waveform shown at the top right of Figure 6. The shaded regions of the sketch represent the lag distortion imposed upon the driving waveform by the actuator hysteresis loop. Note that this distortion represents a substantial delay (essentially a dead time of one-sixth of the limit-cycle period).

Now contrast the above with the behavior of the actuator when sgn e_p has been encoded as a PSV signal. *Provided* the actuator input can assume the form shown in Figure 7, despite the presence of lags and rate limits between sgn u and the actuator, the new output waveform becomes another PSV signal, as shown in the figure, and this new output exhibits significantly less lag distortion than existed previously. In principle, then, and subject to the above proviso, the PSV encoder has reduced the effective dead time of the actuator. Theory says that this reduction in effective dead time should increase the control system limit-cycle frequency, f_{LC}. And with a raise in the frequency, the amplitude of the limit-cycle oscillation should be proportionately reduced (ignoring possible resonance effects). The designer's question, however, is whether the necessary rapid, random excursions about a mean dynamic input to the actuator are achievable in view of the natural up-stream lags and rate limits. Based upon results of this project, the answer appears to be: "For the F-101B tested, yes". The matter seems to hinge on differences between the small-signal and large-signal characteristics of system elements between the controller and the final actuator. Indications are that the small signal responses of these elements are not as severely rate-limited as are the large-signal (mean) responses.

Simulations (with appropriate actuator hysteresis) and ground tests of the F-101B/SOC system measured a closed-loop f_{LC} of 20 Hz with the PSV encoders functioning.* Flight tests showed an intermittent oscillation to be present at approximately 16 Hz† (the lower f_{LC} was possibly due to proximity in frequency of the airframe second bending mode), with no oscillations observed at frequencies higher than 16 Hz. No ground or flight tests were conducted with the PSV encoders disabled. Simulation experiments run without actuator hysteresis produced f_{LC} in the range 33 - 60 Hz (note that the encoder elevated its own nominal 26.5 Hz auto-oscillation frequency by a similar mechanism to that affecting the actuator behavior). A hysteresis magnitude just great enough to lower f_{LC} of the simulation to 20 Hz with the PSV encoders functioning was sufficient to introduce a substantial limit cycle at approximately 2.9 Hz (close to the airframe short-period mode) with these encoders disabled.

---

* Incremental δ_{SPT} amplitude of this oscillation was approximately 0.20 in peak-to-peak.

† Incremental δ_{SPT} amplitude (when oscillation was present) varied, but averaged approximately 0.25 in peak-to-peak.

While flight tests of the elementary SOC have shown the benefits of PSV encoding, simulations and hardware bench tests have established that directed correlation processes (Section 2) are also feasible. These processes offer many potential benefits for flight vehicle control, particularly in three-axis fly-by-wire systems, which rise out of the ability to deal with strongly coupled, nonlinear plants.

At the present time, considerable attention is directed toward turbopropulsion and air-launched missile applications of self-organizing and learning controllers. (Reference 15-18) The next section describes recent computer simulations of a turbine engine SOC system.

4. TURBINE ENGINE SOC

The pilot of an aircraft expects net thrust of the propulsion system to vary in proportion to his setting of the power lever. Full thrust for any prescribed flight condition should be produced when the power lever is at its maximum angle, whereas the system should give idle thrust at the minimum angle. The thrust reference signal, F_{REF} , at time t can be defined as

$$F_{REF}(t) \equiv K\, L(t) , \tag{16}$$

where $L(t)$ is the power lever angle at time t and K is a constant of proportionality which relates the desired percentage of maximum net thrust to power lever angle.

Thrust error at time t is defined as the difference between the reference and computed net thrust levels, viz.

$$e(t) \equiv F_{REF}(t) - F_{NETC}(t) , \tag{17}$$

in which $F_{NETC}(t)$ denotes net thrust at time t as computed by the control system. (The subject of thrust computation is treated in reference 15.)

Suppose the net thrust computation is used as the basis for closed-loop thrust control via a proportional-plus-integral-plus derivative (PID) controller operating on the thrust error signal. At any prescribed thrust level and flight condition, the principal economic figure of merit for a turbopropulsion system is its thrust specific fuel consumption, TSFC, defined as

$$TSFC \equiv w_f/F_{NET} \simeq w_f/F_{NETC} , \tag{18}$$

where w_f is fuel-flow rate in pounds per hour and F_{NET} is propulsion system net thrust in pounds. (A typical value for TSFC is 0.7). In general, TSFC is dependent on settings of geometric parameters such as turbine inlet areas and exhaust nozzle area, as well as on the operating thrust level.

From Equation (18), assuming steady-state operation

$$\Delta(TSFC) = \frac{\partial(TSFC)}{\partial F_{NET}} \Delta F_{NET} + \frac{\partial(TSFC)}{\partial w_f} \Delta w_f \tag{19}$$

or

$$\Delta(TSFC) = -\frac{1}{F_{NET}} (TSFC\, \Delta F_{NET} - \Delta w_f) . \tag{20}$$

Therefore, if one takes

$$\operatorname{sgn} v = -\operatorname{sgn} \Delta(TSFC) \tag{21}$$

he has the performance assessment relationship (Reference 15)

$$\operatorname{sgn} v = \operatorname{sgn} (TSFC\, \Delta F_{NET} - \Delta w_f) , \tag{22}$$

since $F_{NET} > 0$.

Consider the conditions under which Equation (22) suffices for performance assessment in a self-organizing controller used to minimize TSFC at any thrust level and flight condition. Suppose the SOC uses a sampling interval Δt , and at the beginning of each interval certain control increments are computed and transmitted to actuators, while at the ends of the same intervals the corresponding sgn v's are calculated. If the engine reaches equilibrium at the end of each sampling interval, i.e., if the spool acceleration, $\dot{N}$, becomes zero, measurements of ΔF_{NET} and Δw_f at the ends of the intervals will indicate the true polarities of $\Delta(TSFC)$. But if $\dot{N}$ does not return to zero, the underlying assumption of steady-state conditions is violated,

and there is considerable likelihood of sgn v being incorrect. To relax the requirement for reaching zero $\dot{N}$ by the end of each Δt, so as to permit use of smaller Δt's and thereby reduce SOC convergence time during TSFC optimization, the physical approach is to augment ΔF_{NET} and Δw_f by a term involving $\Delta\dot{N}$, viz.

$$v = \text{sgn}\ (\text{TSFC}\ \Delta F_{NET} - \Delta w_f + c\Delta\dot{N}) , \tag{23}$$

where the value of c depends on the expected thrust potential of residual $\Delta\dot{N}$.

Another important requirement on the performance assessment arises when system operation is limited by an engine boundary. The SOC must cause the geometric parameters to be changed in a way that minimizes thrust error without violating the boundary.

The study of a self-organizing controller which employs Equation (23) in the performance assessment unit is reported in Reference 15. This controller used two correlation units, one to adjust turbine inlet area, the other to adjust exhaust nozzle area. *A priori* boundaries imposed by surge, turbine inlet over-temperature, overspeed, and flameout were programmed into the SOC, which was designed to optimize engine geometry for any thrust level, flight condition, or age of the powerplant. Figure 8 illustrates a typical optimization path of the system from an arbitrary initial condition to the immediate vicinity of the minimum TSFC point. Although contours of constant TSFC are plotted in this figure, it is important to realize that the controller had no prior knowledge of the contours, but had to determine its best operating conditions through experimentation.

The data presented in the reference indicate ability of the SOC optimization system with PID thrust controller to achieve TSFC within about 0.2% of the minimum for any given thrust level, with convergence in 10-45 sec depending on distance from the optimum point. Thrust perturbations resulting from variations in exhaust nozzle area and turbine inlet area had a peak magnitude of approximately 0.2% of maximum thrust. Transient thrust responses following disturbances or power lever changes were rapid, with small thrust overshoot and complete thrust recovery in about 0.3 sec. Steady-state thrust errors were negligible.

Reference 16 (Part II) describes application of SOC techniques to control of spike position and cone angle of a supersonic inlet. Reference 17 introduces a method for adaptively learning the location of an unknown surge boundary so that optimization of a variable geometry compressor via on-line adjustment of stator vane angles can be accomplished without inadvertently surging the engine.

5. SUMMARY, FUTURE PROSPECTS, AND PROBLEMS

This paper has briefly traced the design and development of a particular type of learning system and touched on its applications to aircraft and turbopropulsion system control.* These uses of machine learning employ short-term memory and are referred to as self-organizing systems.

The self-organizing controller can be combined with a learning prediction model in the manner shown in Figure 9 when it is necessary to counteract long-term transport delay phenomena in the plant. Although this control system configuration has not yet been investigated very extensively for aeronautical applications, it is currently receiving considerable attention for industrial process control. It is anticipated that development of these "hierarchical" learning control systems (having both long and short memory retention characteristics) will be pushed in the next few years in response to stringent aeronautical requirements, particularly for use in propulsion supervisory control systems needed by high-performance aircraft.

We have, for brevity's sake in this paper, ignored and/or glossed over some of the real difficulties in the learning system approach. We will summarize some of these difficulties at this point without necessarily offering any solutions but possibly pointing out references suggesting fruitful lines of endeavor.

By and large the environmental situations treated in this paper have been, for learning systems, conceptually simple. We mean by this the following: Given a plant which has a well defined set of parameters which measure performance, one can, in principle, design a learning system that will converge within this parameter space to solutions if solutions exist within this space. These parameters we have called goals or performance asessment measures of the system. We feel that these goals, in our present discussion, have been overconstrained with respect to comparable biological systems in the following ways:

(a) The goals are highly problem specific where the biological system is not nearly so problem specific. For example, in much of the treatment given in this paper, we have been constrained to operating on error and error derivatives of the plant response. We would like ideally to tie instead to some kind of system integrity limits, such as in aircraft:

(1) don't exceed allowable skin temperature limits,

and

(2) don't exceed elastic stress limitations, etc.

As already noted in Section 4, work in turbopropulsion learning control systems is presently moving in the direction of more generalized goals.

---

* Recent research has also dealt with control system applications for air-launched missiles (Reference 18).

While flight tests of the elementary SOC have shown the benefits of PSV encoding, simulations and hardware bench tests have established that directed correlation processes (Section 2) are also feasible. These processes offer many potential benefits for flight vehicle control, particularly in three-axis fly-by-wire systems, which rise out of the ability to deal with strongly coupled, nonlinear plants.

At the present time, considerable attention is directed toward turbopropulsion and air-launched missile applications of self-organizing and learning controllers. (Reference 15-18) The next section describes recent computer simulations of a turbine engine SOC system.

4. TURBINE ENGINE SOC

The pilot of an aircraft expects net thrust of the propulsion system to vary in proportion to his setting of the power lever. Full thrust for any prescribed flight condition should be produced when the power lever is at its maximum angle, whereas the system should give idle thrust at the minimum angle. The thrust reference signal, F_{REF} , at time t can be defined as

$$F_{REF}(t) \equiv K\, L(t) , \tag{16}$$

where $L(t)$ is the power lever angle at time t and K is a constant of proportionality which relates the desired percentage of maximum net thrust to power lever angle.

Thrust error at time t is defined as the difference between the reference and computed net thrust levels, viz.

$$e(t) \equiv F_{REF}(t) - F_{NETC}(t) , \tag{17}$$

in which $F_{NETC}(t)$ denotes net thrust at time t as computed by the control system. (The subject of thrust computation is treated in reference 15.)

Suppose the net thrust computation is used as the basis for closed-loop thrust control via a proportional-plus-integral-plus derivative (PID) controller operating on the thrust error signal. At any prescribed thrust level and flight condition, the principal economic figure of merit for a turbopropulsion system is its thrust specific fuel consumption, TSFC, defined as

$$TSFC \equiv w_f/F_{NET} \simeq w_f/F_{NETC} , \tag{18}$$

where w_f is fuel-flow rate in pounds per hour and F_{NET} is propulsion system net thrust in pounds. (A typical value for TSFC is 0.7). In general, TSFC is dependent on settings of geometric parameters such as turbine inlet areas and exhaust nozzle area, as well as on the operating thrust level.

From Equation (18), assuming steady-state operation

$$\Delta(TSFC) = \frac{\partial(TSFC)}{\partial F_{NET}} \Delta F_{NET} + \frac{\partial(TSFC)}{\partial w_f} \Delta w_f \tag{19}$$

or

$$\Delta(TSFC) = -\frac{1}{F_{NET}} (TSFC\, \Delta F_{NET} - \Delta w_f) . \tag{20}$$

Therefore, if one takes

$$\operatorname{sgn} v = -\operatorname{sgn} \Delta(TSFC) \tag{21}$$

he has the performance assessment relationship (Reference 15)

$$\operatorname{sgn} v = \operatorname{sgn} (TSFC\, \Delta F_{NET} - \Delta w_f) , \tag{22}$$

since $F_{NET} > 0$.

Consider the conditions under which Equation (22) suffices for performance assessment in a self-organizing controller used to minimize TSFC at any thrust level and flight condition. Suppose the SOC uses a sampling interval Δt , and at the beginning of each interval certain control increments are computed and transmitted to actuators, while at the ends of the same intervals the corresponding sgn v's are calculated. If the engine reaches equilibrium at the end of each sampling interval, i.e., if the spool acceleration, $\dot{N}$, becomes zero, measurements of ΔF_{NET} and Δw_f at the ends of the intervals will indicate the true polarities of $\Delta(TSFC)$. But if $\dot{N}$ does not return to zero, the underlying assumption of steady-state conditions is violated,

and there is considerable likelihood of sgn v being incorrect. To relax the requirement for reaching zero $\dot{N}$ by the end of each Δt, so as to permit use of smaller Δt's and thereby reduce SOC convergence time during TSFC optimization, the physical approach is to augment ΔF_{NET} and Δw_f by a term involving $\Delta\dot{N}$, viz.

$$v = \text{sgn}\ (\text{TSFC}\ \Delta F_{NET} - \Delta w_f + c\Delta\dot{N}) , \tag{23}$$

where the value of c depends on the expected thrust potential of residual $\Delta\dot{N}$.

Another important requirement on the performance assessment arises when system operation is limited by an engine boundary. The SOC must cause the geometric parameters to be changed in a way that minimizes thrust error without violating the boundary.

The study of a self-organizing controller which employs Equation (23) in the performance assessment unit is reported in Reference 15. This controller used two correlation units, one to adjust turbine inlet area, the other to adjust exhaust nozzle area. *A priori* boundaries imposed by surge, turbine inlet over-temperature, overspeed, and flameout were programmed into the SOC, which was designed to optimize engine geometry for any thrust level, flight condition, or age of the powerplant. Figure 8 illustrates a typical optimization path of the system from an arbitrary initial condition to the immediate vicinity of the minimum TSFC point. Although contours of constant TSFC are plotted in this figure, it is important to realize that the controller had no prior knowledge of the contours, but had to determine its best operating conditions through experimentation.

The data presented in the reference indicate ability of the SOC optimization system with PID thrust controller to achieve TSFC within about 0.2% of the minimum for any given thrust level, with convergence in 10-45 sec depending on distance from the optimum point. Thrust perturbations resulting from variations in exhaust nozzle area and turbine inlet area had a peak magnitude of approximately 0.2% of maximum thrust. Transient thrust responses following disturbances or power lever changes were rapid, with small thrust overshoot and complete thrust recovery in about 0.3 sec. Steady-state thrust errors were negligible.

Reference 16 (Part II) describes application of SOC techniques to control of spike position and cone angle of a supersonic inlet. Reference 17 introduces a method for adaptively learning the location of an unknown surge boundary so that optimization of a variable geometry compressor via on-line adjustment of stator vane angles can be accomplished without inadvertently surging the engine.

5. SUMMARY, FUTURE PROSPECTS, AND PROBLEMS

This paper has briefly traced the design and development of a particular type of learning system and touched on its applications to aircraft and turbopropulsion system control.* These uses of machine learning employ short-term memory and are referred to as self-organizing systems.

The self-organizing controller can be combined with a learning prediction model in the manner shown in Figure 9 when it is necessary to counteract long-term transport delay phenomena in the plant. Although this control system configuration has not yet been investigated very extensively for aeronautical applications, it is currently receiving considerable attention for industrial process control. It is anticipated that development of these "hierarchical" learning control systems (having both long and short memory retention characteristics) will be pushed in the next few years in response to stringent aeronautical requirements, particularly for use in propulsion supervisory control systems needed by high-performance aircraft.

We have, for brevity's sake in this paper, ignored and/or glossed over some of the real difficulties in the learning system approach. We will summarize some of these difficulties at this point without necessarily offering any solutions but possibly pointing out references suggesting fruitful lines of endeavor.

By and large the environmental situations treated in this paper have been, for learning systems, conceptually simple. We mean by this the following: Given a plant which has a well defined set of parameters which measure performance, one can, in principle, design a learning system that will converge within this parameter space to solutions if solutions exist within this space. These parameters we have called goals or performance asessment measures of the system. We feel that these goals, in our present discussion, have been overconstrained with respect to comparable biological systems in the following ways:

(a) The goals are highly problem specific where the biological system is not nearly so problem specific. For example, in much of the treatment given in this paper, we have been constrained to operating on error and error derivatives of the plant response. We would like ideally to tie instead to some kind of system integrity limits, such as in aircraft:

(1) don't exceed allowable skin temperature limits,

and

(2) don't exceed elastic stress limitations, etc.

As already noted in Section 4, work in turbopropulsion learning control systems is presently moving in the direction of more generalized goals.

---

* Recent research has also dealt with control system applications for air-launched missiles (Reference 18).

While flight tests of the elementary SOC have shown the benefits of PSV encoding, simulations and hardware bench tests have established that directed correlation processes (Section 2) are also feasible. These processes offer many potential benefits for flight vehicle control, particularly in three-axis fly-by-wire systems, which rise out of the ability to deal with strongly coupled, nonlinear plants.

At the present time, considerable attention is directed toward turbopropulsion and air-launched missile applications of self-organizing and learning controllers. (Reference 15-18) The next section describes recent computer simulations of a turbine engine SOC system.

4. TURBINE ENGINE SOC

The pilot of an aircraft expects net thrust of the propulsion system to vary in proportion to his setting of the power lever. Full thrust for any prescribed flight condition should be produced when the power lever is at its maximum angle, whereas the system should give idle thrust at the minimum angle. The thrust reference signal, F_{REF} , at time t can be defined as

$$F_{REF}(t) \equiv K\, L(t) , \qquad (16)$$

where $L(t)$ is the power lever angle at time t and K is a constant of proportionality which relates the desired percentage of maximum net thrust to power lever angle.

Thrust error at time t is defined as the difference between the reference and computed net thrust levels, viz.

$$e(t) \equiv F_{REF}(t) - F_{NETC}(t) , \qquad (17)$$

in which $F_{NETC}(t)$ denotes net thrust at time t as computed by the control system. (The subject of thrust computation is treated in reference 15.)

Suppose the net thrust computation is used as the basis for closed-loop thrust control via a proportional-plus-integral-plus derivative (PID) controller operating on the thrust error signal. At any prescribed thrust level and flight condition, the principal economic figure of merit for a turbopropulsion system is its thrust specific fuel consumption, TSFC, defined as

$$TSFC \equiv w_f/F_{NET} \simeq w_f/F_{NETC} , \qquad (18)$$

where w_f is fuel-flow rate in pounds per hour and F_{NET} is propulsion system net thrust in pounds. (A typical value for TSFC is 0.7). In general, TSFC is dependent on settings of geometric parameters such as turbine inlet areas and exhaust nozzle area, as well as on the operating thrust level.

From Equation (18), assuming steady-state operation

$$\Delta(TSFC) = \frac{\partial(TSFC)}{\partial F_{NET}} \Delta F_{NET} + \frac{\partial(TSFC)}{\partial w_f} \Delta w_f \qquad (19)$$

or

$$\Delta(TSFC) = -\frac{1}{F_{NET}} (TSFC\, \Delta F_{NET} - \Delta w_f) . \qquad (20)$$

Therefore, if one takes

$$\text{sgn } v = -\text{sgn } \Delta(TSFC) \qquad (21)$$

he has the performance assessment relationship (Reference 15)

$$\text{sgn } v = \text{sgn } (TSFC\, \Delta F_{NET} - \Delta w_f) , \qquad (22)$$

since $F_{NET} > 0$.

Consider the conditions under which Equation (22) suffices for performance assessment in a self-organizing controller used to minimize TSFC at any thrust level and flight condition. Suppose the SOC uses a sampling interval Δt , and at the beginning of each interval certain control increments are computed and transmitted to actuators, while at the ends of the same intervals the corresponding sgn v's are calculated. If the engine reaches equilibrium at the end of each sampling interval, i.e., if the spool acceleration, $\dot{N}$, becomes zero, measurements of ΔF_{NET} and Δw_f at the ends of the intervals will indicate the true polarities of $\Delta(TSFC)$. But if $\dot{N}$ does not return to zero, the underlying assumption of steady-state conditions is violated,

and there is considerable likelihood of sgn v being incorrect. To relax the requirement for reaching zero $\dot{N}$ by the end of each Δt, so as to permit use of smaller Δt's and thereby reduce SOC convergence time during TSFC optimization, the physical approach is to augment ΔF_{NET} and Δw_f by a term involving $\Delta\dot{N}$, viz.

$$v = \text{sgn}\ (\text{TSFC}\ \Delta F_{NET} - \Delta w_f + c\Delta\dot{N}) , \tag{23}$$

where the value of c depends on the expected thrust potential of residual $\Delta\dot{N}$.

Another important requirement on the performance assessment arises when system operation is limited by an engine boundary. The SOC must cause the geometric parameters to be changed in a way that minimizes thrust error without violating the boundary.

The study of a self-organizing controller which employs Equation (23) in the performance assessment unit is reported in Reference 15. This controller used two correlation units, one to adjust turbine inlet area, the other to adjust exhaust nozzle area. *A priori* boundaries imposed by surge, turbine inlet over-temperature, overspeed, and flameout were programmed into the SOC, which was designed to optimize engine geometry for any thrust level, flight condition, or age of the powerplant. Figure 8 illustrates a typical optimization path of the system from an arbitrary initial condition to the immediate vicinity of the minimum TSFC point. Although contours of constant TSFC are plotted in this figure, it is important to realize that the controller had no prior knowledge of the contours, but had to determine its best operating conditions through experimentation.

The data presented in the reference indicate ability of the SOC optimization system with PID thrust controller to achieve TSFC within about 0.2% of the minimum for any given thrust level, with convergence in 10-45 sec depending on distance from the optimum point. Thrust perturbations resulting from variations in exhaust nozzle area and turbine inlet area had a peak magnitude of approximately 0.2% of maximum thrust. Transient thrust responses following disturbances or power lever changes were rapid, with small thrust overshoot and complete thrust recovery in about 0.3 sec. Steady-state thrust errors were negligible.

Reference 16 (Part II) describes application of SOC techniques to control of spike position and cone angle of a supersonic inlet. Reference 17 introduces a method for adaptively learning the location of an unknown surge boundary so that optimization of a variable geometry compressor via on-line adjustment of stator vane angles can be accomplished without inadvertently surging the engine.

5. SUMMARY, FUTURE PROSPECTS, AND PROBLEMS

This paper has briefly traced the design and development of a particular type of learning system and touched on its applications to aircraft and turbopropulsion system control.* These uses of machine learning employ short-term memory and are referred to as self-organizing systems.

The self-organizing controller can be combined with a learning prediction model in the manner shown in Figure 9 when it is necessary to counteract long-term transport delay phenomena in the plant. Although this control system configuration has not yet been investigated very extensively for aeronautical applications, it is currently receiving considerable attention for industrial process control. It is anticipated that development of these "hierarchical" learning control systems (having both long and short memory retention characteristics) will be pushed in the next few years in response to stringent aeronautical requirements, particularly for use in propulsion supervisory control systems needed by high-performance aircraft.

We have, for brevity's sake in this paper, ignored and/or glossed over some of the real difficulties in the learning system approach. We will summarize some of these difficulties at this point without necessarily offering any solutions but possibly pointing out references suggesting fruitful lines of endeavor.

By and large the environmental situations treated in this paper have been, for learning systems, conceptually simple. We mean by this the following: Given a plant which has a well defined set of parameters which measure performance, one can, in principle, design a learning system that will converge within this parameter space to solutions if solutions exist within this space. These parameters we have called goals or performance asessment measures of the system. We feel that these goals, in our present discussion, have been overconstrained with respect to comparable biological systems in the following ways:

(a) The goals are highly problem specific where the biological system is not nearly so problem specific. For example, in much of the treatment given in this paper, we have been constrained to operating on error and error derivatives of the plant response. We would like ideally to tie instead to some kind of system integrity limits, such as in aircraft:

(1) don't exceed allowable skin temperature limits,

and

(2) don't exceed elastic stress limitations, etc.

As already noted in Section 4, work in turbopropulsion learning control systems is presently moving in the direction of more generalized goals.

---

* Recent research has also dealt with control system applications for air-launched missiles (Reference 18).

While flight tests of the elementary SOC have shown the benefits of PSV encoding, simulations and hardware bench tests have established that directed correlation processes (Section 2) are also feasible. These processes offer many potential benefits for flight vehicle control, particularly in three-axis fly-by-wire systems, which rise out of the ability to deal with strongly coupled, nonlinear plants.

At the present time, considerable attention is directed toward turbopropulsion and air-launched missile applications of self-organizing and learning controllers. (Reference 15-18) The next section describes recent computer simulations of a turbine engine SOC system.

4. TURBINE ENGINE SOC

The pilot of an aircraft expects net thrust of the propulsion system to vary in proportion to his setting of the power lever. Full thrust for any prescribed flight condition should be produced when the power lever is at its maximum angle, whereas the system should give idle thrust at the minimum angle. The thrust reference signal, F_{REF} , at time t can be defined as

$$F_{REF}(t) \equiv K\, L(t) , \tag{16}$$

where $L(t)$ is the power lever angle at time t and K is a constant of proportionality which relates the desired percentage of maximum net thrust to power lever angle.

Thrust error at time t is defined as the difference between the reference and computed net thrust levels, viz.

$$e(t) \equiv F_{REF}(t) - F_{NETC}(t) , \tag{17}$$

in which $F_{NETC}(t)$ denotes net thrust at time t as computed by the control system. (The subject of thrust computation is treated in reference 15.)

Suppose the net thrust computation is used as the basis for closed-loop thrust control via a proportional-plus-integral-plus derivative (PID) controller operating on the thrust error signal. At any prescribed thrust level and flight condition, the principal economic figure of merit for a turbopropulsion system is its thrust specific fuel consumption, TSFC, defined as

$$TSFC \equiv w_f/F_{NET} \simeq w_f/F_{NETC} , \tag{18}$$

where w_f is fuel-flow rate in pounds per hour and F_{NET} is propulsion system net thrust in pounds. (A typical value for TSFC is 0.7). In general, TSFC is dependent on settings of geometric parameters such as turbine inlet areas and exhaust nozzle area, as well as on the operating thrust level.

From Equation (18), assuming steady-state operation

$$\Delta(TSFC) = \frac{\partial(TSFC)}{\partial F_{NET}} \Delta F_{NET} + \frac{\partial(TSFC)}{\partial w_f} \Delta w_f \tag{19}$$

or

$$\Delta(TSFC) = -\frac{1}{F_{NET}} (TSFC\, \Delta F_{NET} - \Delta w_f) . \tag{20}$$

Therefore, if one takes

$$\operatorname{sgn} v = -\operatorname{sgn} \Delta(TSFC) \tag{21}$$

he has the performance assessment relationship (Reference 15)

$$\operatorname{sgn} v = \operatorname{sgn} (TSFC\, \Delta F_{NET} - \Delta w_f) , \tag{22}$$

since $F_{NET} > 0$.

Consider the conditions under which Equation (22) suffices for performance assessment in a self-organizing controller used to minimize TSFC at any thrust level and flight condition. Suppose the SOC uses a sampling interval Δt , and at the beginning of each interval certain control increments are computed and transmitted to actuators, while at the ends of the same intervals the corresponding sgn v's are calculated. If the engine reaches equilibrium at the end of each sampling interval, i.e., if the spool acceleration, $\dot{N}$, becomes zero, measurements of ΔF_{NET} and Δw_f at the ends of the intervals will indicate the true polarities of $\Delta(TSFC)$. But if $\dot{N}$ does not return to zero, the underlying assumption of steady-state conditions is violated,

and there is considerable likelihood of sgn v being incorrect. To relax the requirement for reaching zero $\dot{N}$ by the end of each Δt, so as to permit use of smaller Δt's and thereby reduce SOC convergence time during TSFC optimization, the physical approach is to augment ΔF_{NET} and Δw_f by a term involving $\Delta\dot{N}$, viz.

$$v = \text{sgn}\ (\text{TSFC}\ \Delta F_{NET} - \Delta w_f + c\Delta\dot{N}) , \tag{23}$$

where the value of c depends on the expected thrust potential of residual $\Delta\dot{N}$.

Another important requirement on the performance assessment arises when system operation is limited by an engine boundary. The SOC must cause the geometric parameters to be changed in a way that minimizes thrust error without violating the boundary.

The study of a self-organizing controller which employs Equation (23) in the performance assessment unit is reported in Reference 15. This controller used two correlation units, one to adjust turbine inlet area, the other to adjust exhaust nozzle area. *A priori* boundaries imposed by surge, turbine inlet over-temperature, overspeed, and flameout were programmed into the SOC, which was designed to optimize engine geometry for any thrust level, flight condition, or age of the powerplant. Figure 8 illustrates a typical optimization path of the system from an arbitrary initial condition to the immediate vicinity of the minimum TSFC point. Although contours of constant TSFC are plotted in this figure, it is important to realize that the controller had no prior knowledge of the contours, but had to determine its best operating conditions through experimentation.

The data presented in the reference indicate ability of the SOC optimization system with PID thrust controller to achieve TSFC within about 0.2% of the minimum for any given thrust level, with convergence in 10-45 sec depending on distance from the optimum point. Thrust perturbations resulting from variations in exhaust nozzle area and turbine inlet area had a peak magnitude of approximately 0.2% of maximum thrust. Transient thrust responses following disturbances or power lever changes were rapid, with small thrust overshoot and complete thrust recovery in about 0.3 sec. Steady-state thrust errors were negligible.

Reference 16 (Part II) describes application of SOC techniques to control of spike position and cone angle of a supersonic inlet. Reference 17 introduces a method for adaptively learning the location of an unknown surge boundary so that optimization of a variable geometry compressor via on-line adjustment of stator vane angles can be accomplished without inadvertently surging the engine.

5. SUMMARY, FUTURE PROSPECTS, AND PROBLEMS

This paper has briefly traced the design and development of a particular type of learning system and touched on its applications to aircraft and turbopropulsion system control.* These uses of machine learning employ short-term memory and are referred to as self-organizing systems.

The self-organizing controller can be combined with a learning prediction model in the manner shown in Figure 9 when it is necessary to counteract long-term transport delay phenomena in the plant. Although this control system configuration has not yet been investigated very extensively for aeronautical applications, it is currently receiving considerable attention for industrial process control. It is anticipated that development of these "hierarchical" learning control systems (having both long and short memory retention characteristics) will be pushed in the next few years in response to stringent aeronautical requirements, particularly for use in propulsion supervisory control systems needed by high-performance aircraft.

We have, for brevity's sake in this paper, ignored and/or glossed over some of the real difficulties in the learning system approach. We will summarize some of these difficulties at this point without necessarily offering any solutions but possibly pointing out references suggesting fruitful lines of endeavor.

By and large the environmental situations treated in this paper have been, for learning systems, conceptually simple. We mean by this the following: Given a plant which has a well defined set of parameters which measure performance, one can, in principle, design a learning system that will converge within this parameter space to solutions if solutions exist within this space. These parameters we have called goals or performance asessment measures of the system. We feel that these goals, in our present discussion, have been overconstrained with respect to comparable biological systems in the following ways:

(a) The goals are highly problem specific where the biological system is not nearly so problem specific. For example, in much of the treatment given in this paper, we have been constrained to operating on error and error derivatives of the plant response. We would like ideally to tie instead to some kind of system integrity limits, such as in aircraft:

(1) don't exceed allowable skin temperature limits,

and

(2) don't exceed elastic stress limitations, etc.

As already noted in Section 4, work in turbopropulsion learning control systems is presently moving in the direction of more generalized goals.

---

* Recent research has also dealt with control system applications for air-launched missiles (Reference 18).

(b) We have only briefly discussed situations where multiple goals are the essence. Possibility of competition among the goals may arise. Solutions for one goal may not be solutions when several goals have to be simultaneously satisfied. This will almost certainly be the case in systems which contain high degrees of interaction among their subsystems and/or degrees of freedom. References 10 and 12 deal with conflicting goals involving system error level versus control resource expenditures.

(c) We have attempted no discussion of unreliable, noisy, or incompletely specified goals. In any complex situation with many degrees of freedom this is probably more often the case than not. This is related to the further question as to whether it is possible to have non-specific goals that have the property of evolving more specific sub-goals. This certainly seems to be the case in living systems.

We have neither the time nor the space to go into these problems and their possible solutions. Probably the most complete discussion to date may be found in the whole of Reference 8.

In a more extended discussion, we would have been much more careful in discussing the relation between various logical processes and learning. There are at least three logical processes to be distinguished. These are known as deduction, induction, and abduction.

Deduction is, of course, reasoning from some general premise to particular cases. It is applicable to any situation where we have complete knowledge and/or not too many degrees of freedom. Mathematical reasoning is a case in point, as is also design of optimum controllers or filters for linear systems.

Induction is reasoning from particular cases to general conclusions. It is basically a trial and error experimental procedure. It is apparently applicable to any situation which is not too large and/or for which we do not have complete knowledge.

If the environmental situation about which we attempt to reason becomes too large, then inductive methods become difficult. This is easily understood. There are too many possible inductions or experiments that can be made and one needs some kind of procedure to suggest which inductions could most profitably be tried first. This procedure has been called abduction or reasoning by analogy, by hunch, or possibly more suggestive, reasoning from the seemingly unrelated. This method is, by and large, that used by living systems in all their creative endeavors. It is one of the qualities we identify most closely with intelligence. It allows, for example, living systems to reason about dangerous situations without themselves having to conduct dangerous experiments in the situation. It allows gaining of knowledge about unknown situations by analogy to known situations.

None of the machines we have discussed do this kind of logical processing. We feel, however, that the inductive process for these machines has been demonstrated in this paper.

Abduction seems possible in machines, but that awaits the future.

6. ACKNOWLEDGMENT

The authors gratefully acknowledge the support and sponsorship of the Air Force Systems Command, US Air Force.

REFERENCES

1. Gwinn, C.W. — *The Scope and Methods of Engineering Bionics.* Scientia, Vol.100, No.246, 1965.

2. Wiener, N. et al. — *Behaviour, Purpose, and Teleology in Modern Systems Research for Behavioral Scientists.* W.Buckley, Ed., Aldine Publ. Co., 1968, pp.221-226.

3. Wiener, N. — *Cybernetics.* MIT Press, 1948.

4. Ashby, W.R. — *Design for a Brain.* Wiley, 1952.

5. Ashby, W.R. — *Introduction to Cybernetics.* Wiley, 1956.

6. Eddington, A.S. — *Fundamental Theory.* Cambridge Univ. Press, 1948, p.30.

7. Rosen, R. — *Optimality Principles in Biology.* Butterworths, 1967, Chap.11.

8. Beer, S. — *Decision and Control.* Wiley, 1966.

9. Barron, R.L. — *Self-Organizing and Learning Control Systems in Cybernetic Problems in Bionics.* (H.L.Oestreicher and D.R.Moore, Eds.), Gordon and Breach, 1968, pp.147-203.

10. Barron, R.L. — *Adaptive Flight Control Systems.* Paper presented at NATO AGARD Bionics Symposium, Brussels, September 1968.

11. Barron, R.L. — *Self-Organizing Control: The Next Generation of Controllers. Part I - The Elementary SOC. Part II - The General Purpose SOC.* Control Engineering, Vol.15, Nos.2 and 3, February 1968, pp.70-74, March 1968, pp.69-74.

12. Barron, R.L. — *Analysis and Synthesis of Advanced Self-Organizing Control Systems - II.* Adaptronics, Inc. final technical report under Contract F33615-67-C-1681 with US Air Force Avionics Laboratory, AFAL TR-68-236, September 1968.

13. McKechnie, R.M. III; Barron, R.L. — *Design Principles for Self-Organizing Control System Flight Hardware.* Proc. 19th Ann. NAECON, 1967, pp.465-473.

14. — *Design, Fabrication, and Flight Testing of Self-Organizing Flight Control System.* Adaptronics, Inc. final technical report under Contract AF 33(615)-5141 with US Air Force Flight Dynamics and Avionics Laboratories, May 1970.

15. Barron, R.L.; Cleveland, D. et al. — *Self-Organizing Control of Advanced Turbine Engines.* Adaptronics Inc. final technical report under Contract F33615-68-C-1606 with US Air Force Aero Propulsion Laboratory, August 1969.

16. Billig, L.O. — *Adaptive Controls. Part I: A Survey. Part II: Self-Organizing Systems.* Instruments and Control Systems, Vol.42, September 1969, pp.147-152, October 1969, pp.126-131.

17. Mucciardi, A.N.; Barron, R.L. et al. — *Adaptive Optimization System for Maximizing Performance of Variable-Geometry Turbojet Compressor with Unknown Surge Boundaries.* Adaptronics, Inc. proprietary technical report #2 under Pratt & Whitney Aircraft Division, United Aircraft Corp., Purchase Order F191275, April 1970.

18. Cleveland, D.; Barron, R.L. et al. — *Research and Development on Self-Organizing Control Systems for Air-Launched Missiles.* Adaptronics, Inc. final technical report under Contract N00019-69-C-0106 with US Naval Air Systems Command, November 1969.

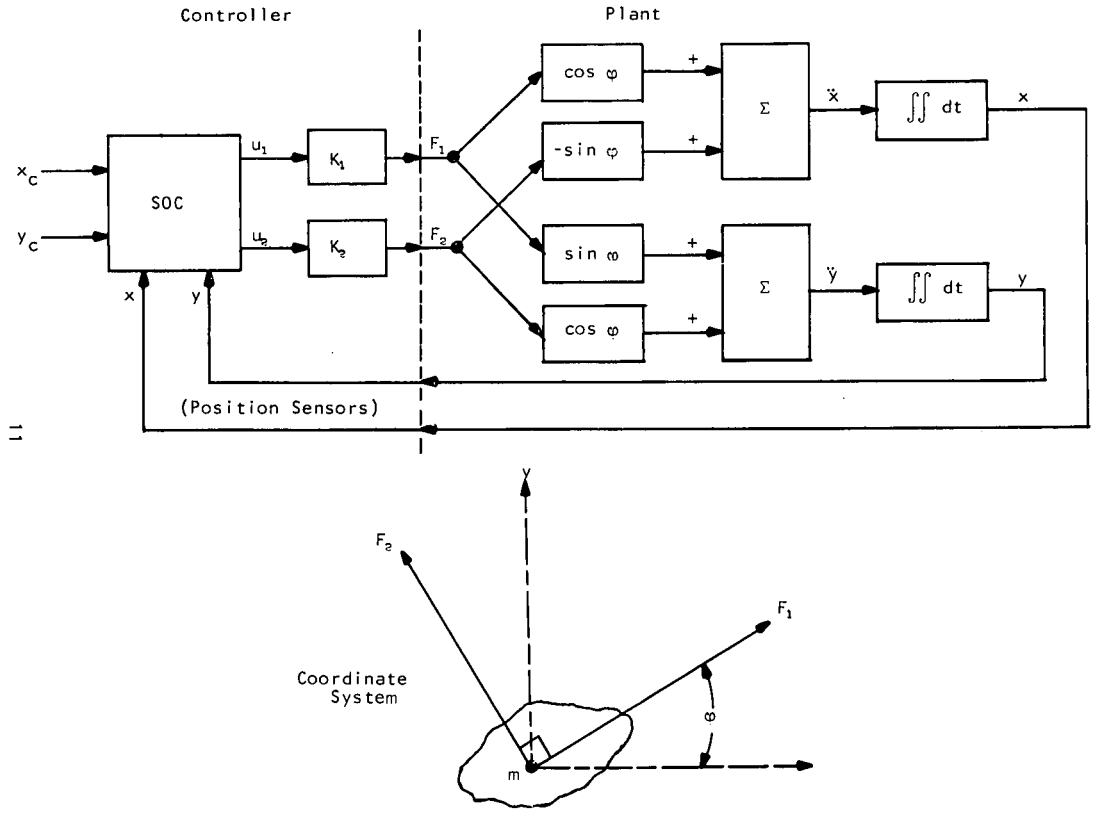
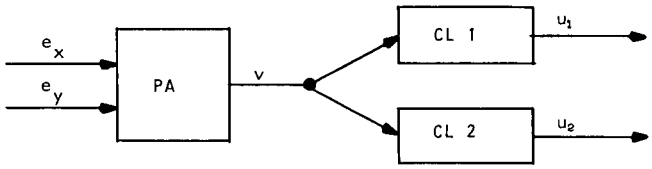


Fig. 1 Illustrative SOC application

Configuration A:



Configuration B:

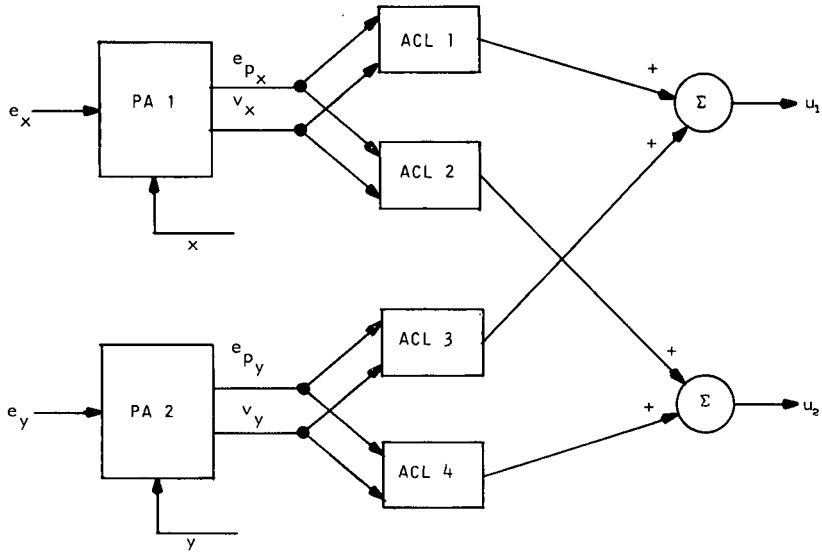


Fig. 2 Alternative SOC configurations for illustrative application

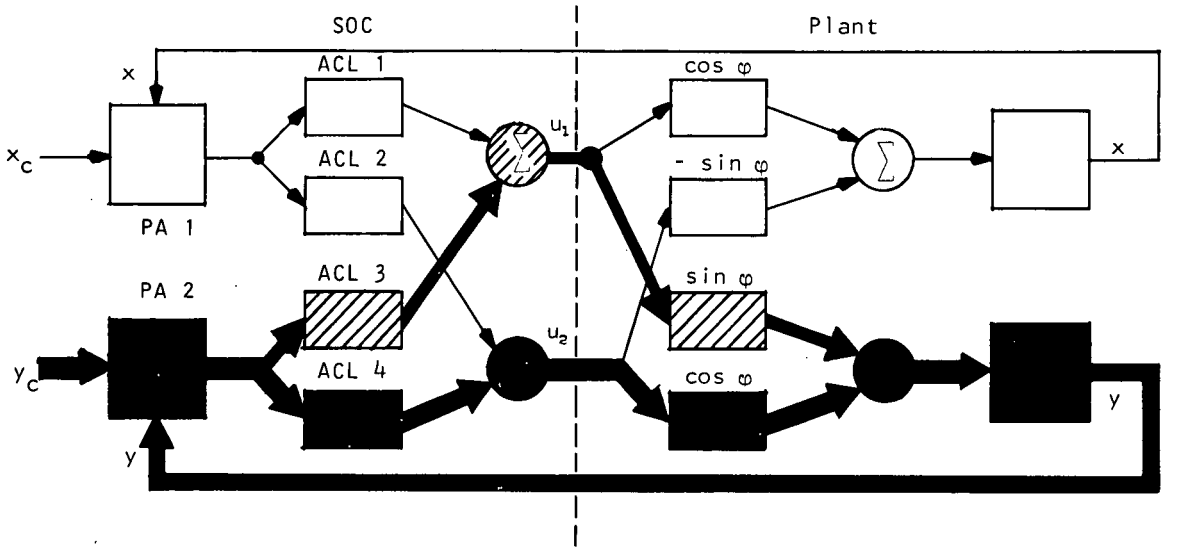


Fig.3 Signal flow in illustrative SOC application

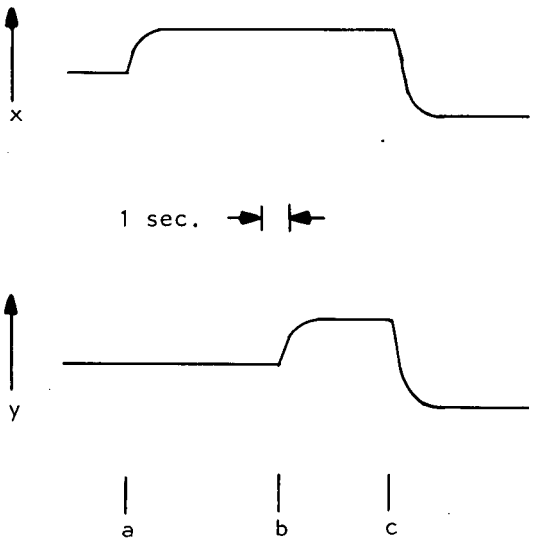


Fig.4 Response of illustrative SOC system to x_c and y_c steps. $\varphi = 30^o$

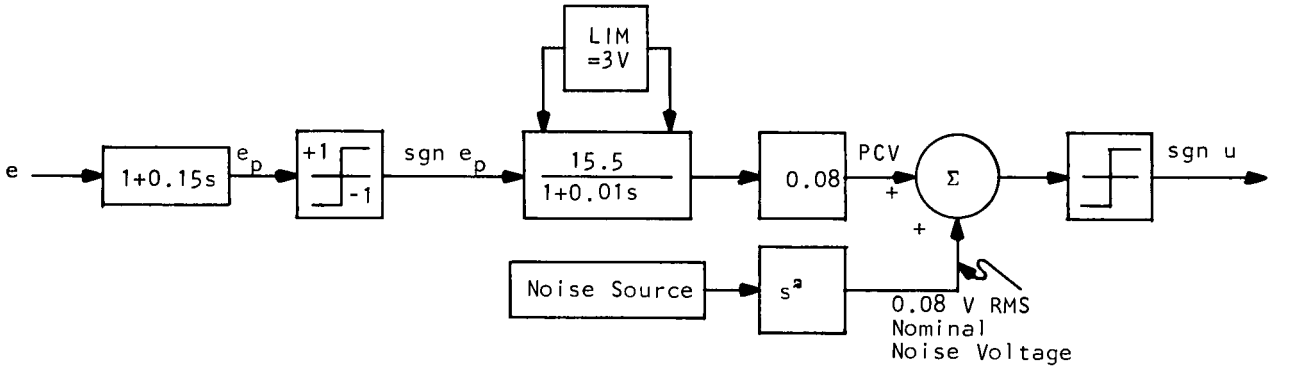


Fig.5 Functional block diagram of SOC logic unit (SLU)

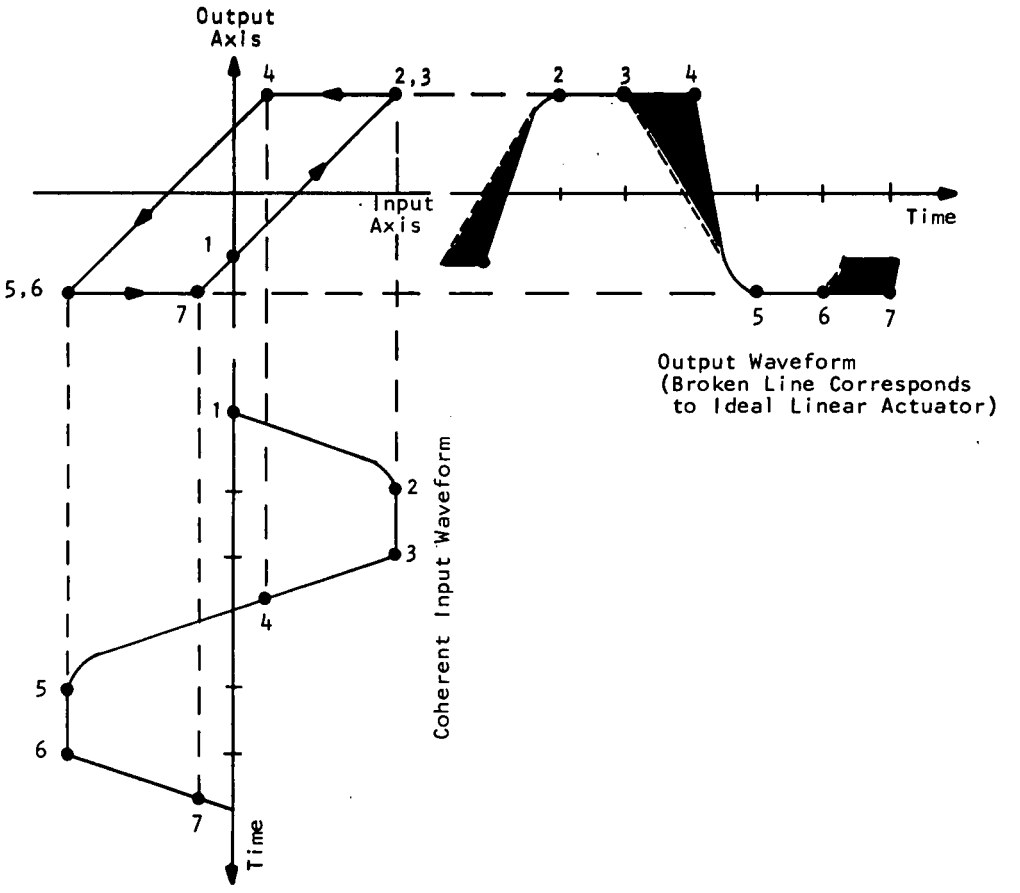


Fig.6 Sketch illustrating effect of actuator hysteresis on waveform of deterministic control signal

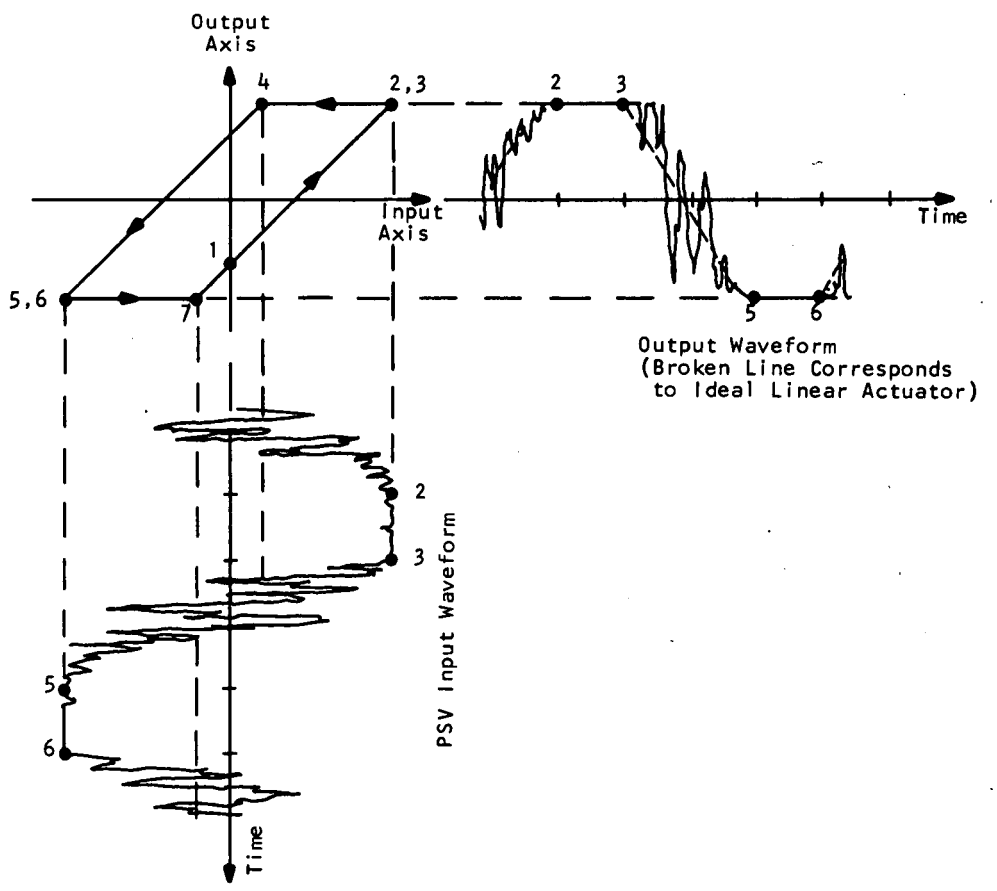


Fig.7 Sketch illustrating typical effect of actuator hysteresis on waveform of PSV control signal

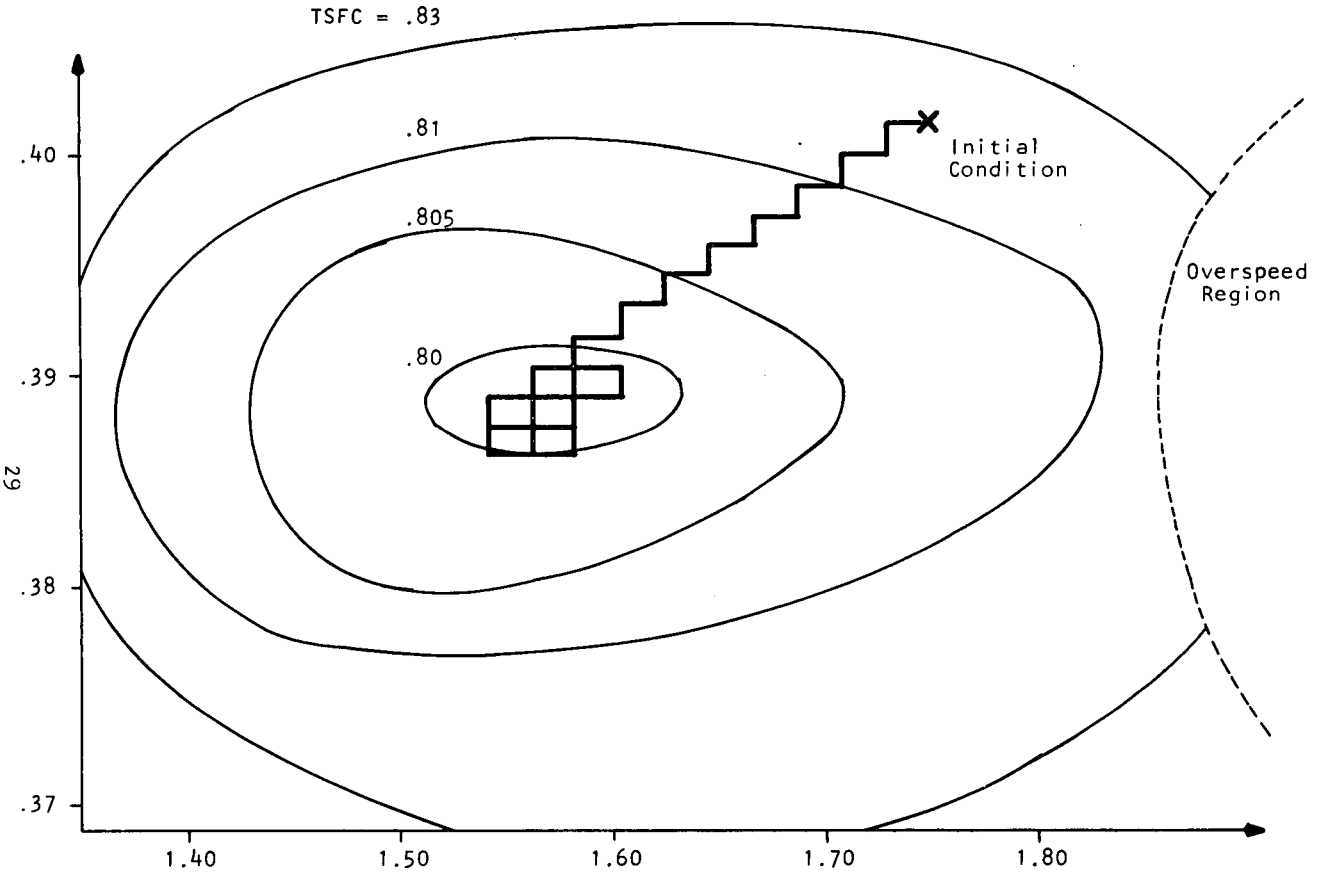


Fig.8 TSFC optimization for a high power condition

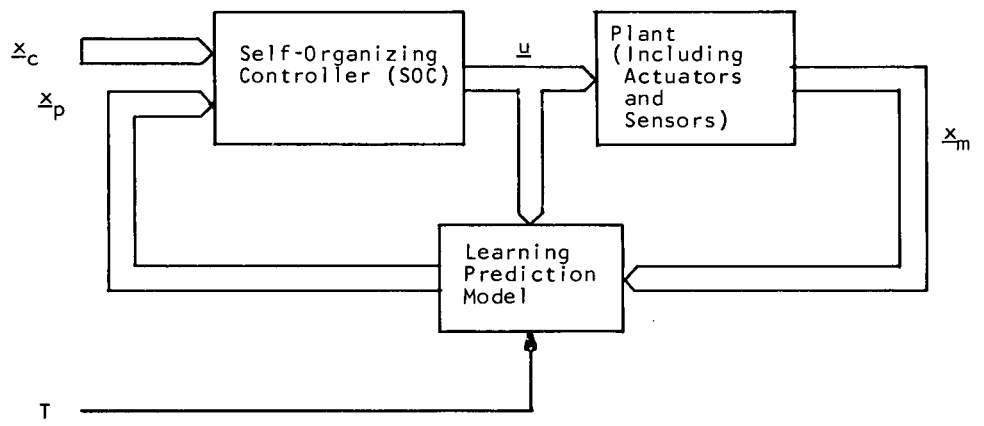


Fig.9 Multivariable control system incorporating SOC and learning prediction model

HIGH INTEGRITY DIGITAL FLIGHT CONTROL

by

J. F. Meredith and K. A. Helps

Smiths Industries Limited, Aviation Division,
Cheltenham, England.

HIGH INTEGRITY DIGITAL FLIGHT CONTROL

J. F. Meredith and K. A. Helps

1. INTRODUCTION

The trend from analogue toward digital implementation of flight control systems is being caused by several factors, some concerned with the rapidly developing digital technology and others which relate to certain inherent difficulties in providing high integrity operation with analogue systems. One of the more important of this latter class is the consequence of the variation in performance of nominally identical pieces of equipment. Thus in a multiplex system each analogue sub-channel has a certain design transfer function which is approximately achieved in practice by the use of one or more amplifiers of finite gain together with various smoothing and stabilising filters. Discrepancies between sub-channels will arise because of the legitimate variation of component values within their tolerance band. These discrepancies in transfer function may affect the signals which are being processed, in a way which is indistinguishable from certain fault conditions. In practice this means that the level of the threshold of signal discrepancy above which the system detects a fault must be made higher than the level which may legitimately be caused by component value variations. The system integrity is thus compromised because it allows the existence of a set of fault conditions which cannot be detected.

Furthermore the effective use of automatic testing procedures, which with the increasing complexity of aircraft control systems, are of considerable benefit in terms of reduced turn-round time and increased availability, is made more difficult by the presence of the analogue system tolerances.

It is upon such problems as these that a digital high integrity system may make a significant impact.

2. INTEGRITY

If the pilot is regarded as the manager of the aircraft, then serving him he has various special units which are controlling particular aspects of the aeroplane performance. He is providing the command to engage this particular control mode or to perform that particular manoeuvre and is then delegating the responsibility to the particular unit to correctly carry out the operation.

Now some control modes are more critical to the immediate safety of the aircraft than are others. Particularly in the landing manoeuvre the time available for the crew to take any effective remedial action after a system failure has been detected, is in some circumstances zero and in all cases less than a few seconds. Thus before the pilot can commit the aeroplane to such a critical manoeuvre he must make the decision that the probability of system failure is acceptable low.

The evidence which allows him to make this decision arises in two stages. Firstly, the design of the equipment and the quality of the components used have assured him, or his representative, that provided the equipment starts from a defined state, then the probability of failure within some specified time interval is less than some acceptable figure. This evidence is clearly not dependent upon any particular flight and so secondly he must be informed of the current state of the equipment, that is whether it is in the design state for which the a priori probability figure may be applied, or whether certain faults or defects have been detected which will lead to a higher probability of failure in subsequent use.

The attribute of the system which is responsible for that degree of self analysis and information transfer which allows the system to warn the pilot of any malfunction, either total or incipient, is the system integrity. The total system integrity is achieved in part by pre-flight testing and in part by the system self examination and warning carried out during the flight.

A figure of merit for a system claiming a level of integirty can be provided by the following ratio:-

$$I = \frac{\text{Conditional Probability of System failure given that no warning has occurred}}{\text{Probability of System failure}}$$

This is unity for a system with no integrity pretensions and will tend to zero as the integrity of the system increases. Thus in order to achieve a high integrity the system must be designed to have a sufficient degree of internal self-checking to provide a warning of any system defect which could result in system failure.

3. DIGITAL SYSTEMS

In the achievement of this aim digital systems provide both advantages and problems. To make best use of the advantages, which derive principally from the discrete nature of information within a digital machine and the consequently finite probability of equality between two numbers; while at the same time minimising the problems, which involve the large numbers of definable states within a computer and the consequent impossibility of complete testing in any reasonable time period, it is necessary to provide a system structure which allows effective checking of the programme being performed.

Consider a multiplexed system in which the individual computers are performing the sub-channel functions. The outputs of the sub-channels will differ both in magnitude and time of occurance; in time because the independence of the computers will demand separate clocks and in magnitude because the sensor information will in general be sampled at slightly different times. In consequence the task of checking between these independent sub-channels is made more difficult and less effective than is possible by constraining them to operate in ways described below.

Firstly a set of independent sub-channel computers can be held in a sufficient state of synchronisation to allow the free exchange of data between them, while still retaining independent clocks. The requirement of sub-channel independence prohibits the use of interrupt as a means of communication. Fig. 1 shows the preferred arrangement. Data to be transferred is placed by the sending computer in a frontier store, which is particular to information transfer in this direction between these two sub-channels, from this store it may be picked up by the intended recipient at a time determined by it's own programme and clock. To enable this form of transfer the degree of synchronisation required is only that necessary to ensure that the correct data is taken up without too great a penalty in time; that is individual sub-channel computers need only be synchronised to the extent of being within one or two word cycles of each other.

This data exchange process which is enabled by the synchronisation between the computers can itself be used in order to effect the synchronisation process. Each computer can compare the time of receipt of a timing signal from each of the other sub-channels with the time at which it issues its timing signal to the others. Given this information each sub-channel can independently make an assessment of its timing relative to that of the others and take an appropriate length path through a delay routine. This synchronisation process can be an entirely software routine and carried out once per cycle of the main computer programme.

The data exchange paths provide a physical link between the individual sub-channels; clearly their unidirectional nature is essential if the possibility of common faults is to be avoided. The use of fibre optic techniques may provide the ultimate degree of isolation between the sub-channels but their use is by no means essential.

Using the facility for data exchange between the individual sub-channels the system can be operated in such a way that each sub-channel is provided with identical rather than similar sensor data. If each sub-channel inputs sensor data through the A/D interfaces (Fig. 2) then the resulting digital numbers can be exchanged across the frontiers so that each sub-channel is in possession of an identical set of sensor data. If then each produces an amalgamated result by the same software process then the calculations can begin in each sub-channel with identical numbers and consequently the numbers at any subsequent stage of the calculations will be identical. This means that subsequent cross checking between the sub-channels, which is again enabled by the data exchange links, can take place at any desired stage and the non identity of the signals from these separate sub-channels is sufficient evidence for the existence of a fault. Similarly if a computer fault leads to a route change the subsequent loss of synchronism gives indication of the occurrence.

We have therefore the possibility of a set of independent sub-channels each of which is continuously forming an assessment of the system state by sensor analysis, by comparison

between the computers for both magnitude and time discrepancies. Any sub-channel is capable of warning the pilot if it sees any discrepancy and from the warnings the pilot is able to decide whether the safety of the aircraft can be entrusted to the equipment.

In order to make such a scheme successful, the programmes of the individual sub-channels must be made to include a substantial body of checking routines. This checking must test not only the correct functioning of the program which is being performed in the current circumstance but also that which will be performed in the event of a fault occurring elsewhere in the system. In this manner latent faults in the system are exposed.

4. LATENT COMPUTER FAULTS

The concept of a latent fault is essential to an appreciation of the practical difficulties of achieving high integrity. A latent fault in a system is a fault that has no effect on the outputs of the system until another fault occurs. When another fault occurs it may or may not reveal itself, but if it does then the system will behave as though it were undergoing a double fault. The need to cope with simultaneous faults is something that should be minimized; it cannot be avoided altogether by a non-distributed computer because a single hardware fault will often manifest itself as a multiple operational fault.

The task of minimizing latent faults may be seen as the task of ensuring that contingency plans incorporated in the program can be achieved by every computer which remains operational. Such contingency plans are:

(1) Isolating the system from a computer that is reckoned to be faulty by its colleagues.

(2) Ignoring a sensor that is agreed by the computers to be faulty.

(3) Giving pilot warnings (e.g. of system state).

(4) Obeying a program associated with a subsequent and critical flight task (e.g. autolanding).

The importance of detecting latent faults is particularly acute in duplex systems which have a reversion to a simple non-digital control in the event of unexpected computer disagreement, cut out being effected whenever either computer detects a discrepancy. A single latent fault in such a system could undermine its integrity. But latent faults can still be damaging to both triplex and monitored duplex systems. However acute or mild the loss of integrity caused by latent faults, it will be aggravated by long periods of switch off.

5. CONVENTIONAL COMPUTER CHECKS

There are well established techniques for carrying out checksum, arithmetic and logic tests, and for relating these tests to the computer's structure to avoid excessive duplication of checking or untested areas of hardware. Because of the large proportion of hardware faults which are sufficiently context dependent to avoid being detected in this way, one is forced to consider other methods for reducing the number of possible latent faults. (It should be noted in passing that any number sensitive or context sensitive fault i.e. any fault which might slip through the mesh of a well devised conventional check out scheme, may be thought of as a latent fault).

An important tactic for achieving the strategy of minimizing possible latent faults, is to program the computers to run frequently and regularly through important latent program zones.

6. SYSTEMATIC AD HOC CHECKING

It is important to check that each critical program branch is in fact capable of performing its function correctly. Fig. 3 displays a simple technique for doing this and the programming pattern shown is one which occurs repeatedly in a high integrity digital flight control system that does not ignore the possibility of latent faults. As will be seen from the figure and the accompanying fault analysis, at the superficial level where number sensitive faults are ignored, a possible latent fault can be removed at the expense of extra program and time. (This is an area where hardware realization of the same ad hoc checking principle may prove an effective method). The contingency exit referred to in fig. 3 would normally lead to a total or partial channel cut out in the case of a duplex reversionary or monitored duplex system. In the case of a triplex system the programming tends to be more interwoven because of the three way comparisons and majority decisions. But the same principle applies in general, that at each key program branch to emergency action, if the emergency is not present the program simulates an emergency and sets a test indicator, and the branch is passed again, leading this time to the beginning of the emergency action, which in turn will be aborted because the test indicator has been set.

7. OVERLAPPING PROGRAMMING

The technique of systematic ad hoc checking of key program branching has as its natural complement the technique of overlapping programming. This may be illustrated at its simplest by the cut out procedure security arrangements for a duplex reversionary digital controller, in which a variety of possible discrepancies detected at various parts of the program of either computer can lead to a cut out of both computers on the grounds that some simple reversionary control is preferable to suspect sophisticated digital control.

In an overlapping programming scheme (see Fig. 4) the cut outs consist of two parallel pairs of switches in series, one pair for each computer. Each critical program branch has its own ad hoc test to check that the route to the beginning of cut out is open in emergency. During every control cycle a test route is followed which, after overlapping the ends of all the ad hoc tests, opens, checks and recloses one of the cut out switches before opening, checking and reclosing the other switch. The whole checking scheme is so devised and its components overlap in such a way that when a proper cut out is required, the whole route to cut out and the whole double cut out operation has been very recently proved with all branches having been tested in the appropriate sense.

This technique is, of course, not exclusive to duplex reversionary systems with disagreement triggered cut out, but has wider applications in realistic digital controllers with higher levels of multiplexing.

8. TESTING IN CONTEXT

It was stated above that a substantial proportion of computer faults may well be of the number sensitive or context sensitive variety. (Such faults in general purpose computation normally pass unnoticed and are hardly ever pinpointed.) New computer designs and computers mounted in severe environmental conditions are particularly likely to exhibit such faults. More importantly, whether or not such faults are exhibited, the demonstration of their absence is a difficult matter, and has a bearing on the way that one attempts to achieve integrity.

An advantage of the ad hoc checking of latent route entry branches and of the overlapping programming technique is that the digital operations are checked as nearly as possible in the context in which they might susbequently be required, so that the number sensitive faults which are detected are those most likely to be of danger when the real contingency subsequently arises.

9. TRANSIENT FAULTS AND COMPUTER LOADING

There is a problem in distinguishing between context sensitive faults and transient faults provoked by external noise. Any scheme for detecting one kind is likely to detect the other. Their chief distinguishing feature is that context sensitive faults are more likely to be repeated each computing cycle. It seems likely that the best way of dealing with transient faults, if for reliability reasons one is driven from treating them in the same category as permanent faults, is to take provisional action on the basis of a suspected computer defect, to give a warning and to allow for subsequent attempted reinstatement only as a result of human decision. Such a reinstatement would be quite similar to initial engagement and need involve no great extra program loading. To attempt automatic survival by a faulty computer of a fault condition pending a decision on the permanence of the fault is a sure way of substantially increasing the computer loading and jeopardizing the integrity.

Typically, that part of the control cycle and program length required for the kind of checking outlined above is 60% of the total. The time loading is fairly flexible since a substantial part of the overlapping checks need not be done as a whole during every control cycle.

The internal integrity techniques described are not essential to a successful high integrity digital scheme, but may be regarded as good standard practice for approaches to the problem which make use of conventional computing techniques.

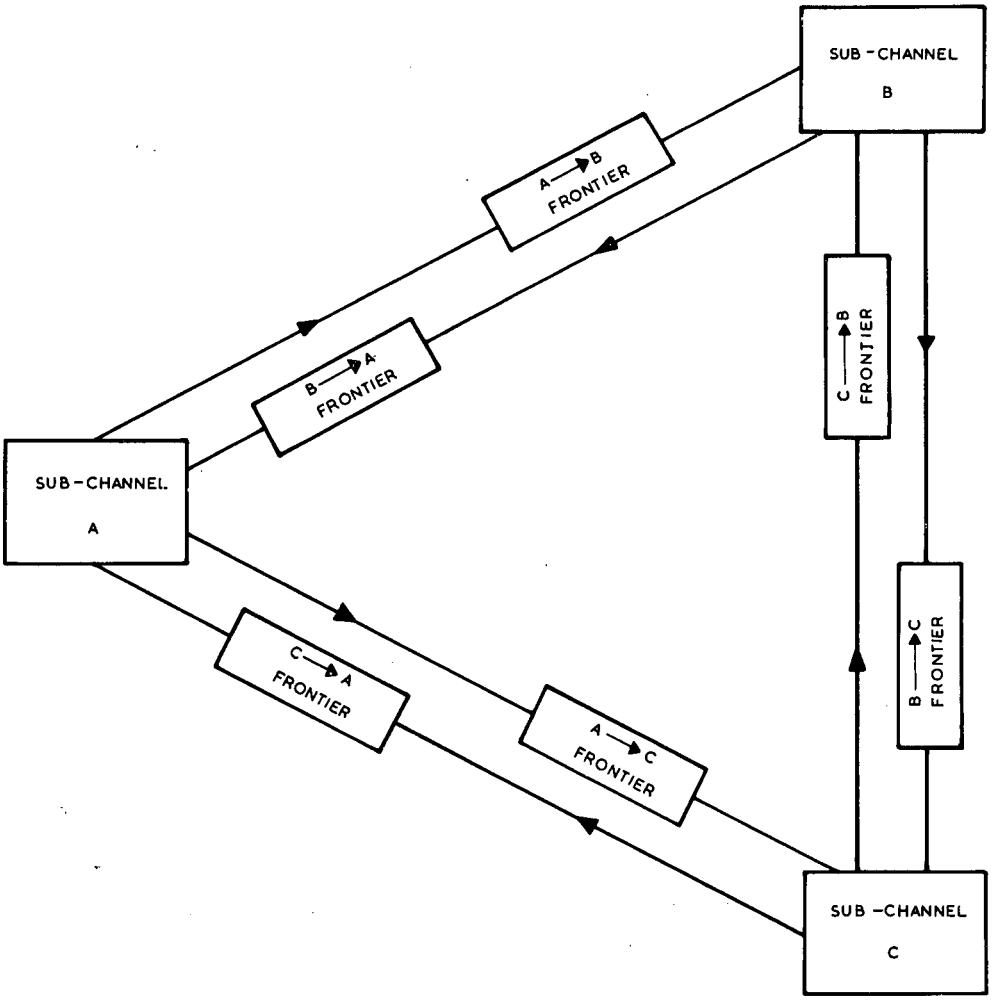


Fig. 1 Sub-channel data exchange

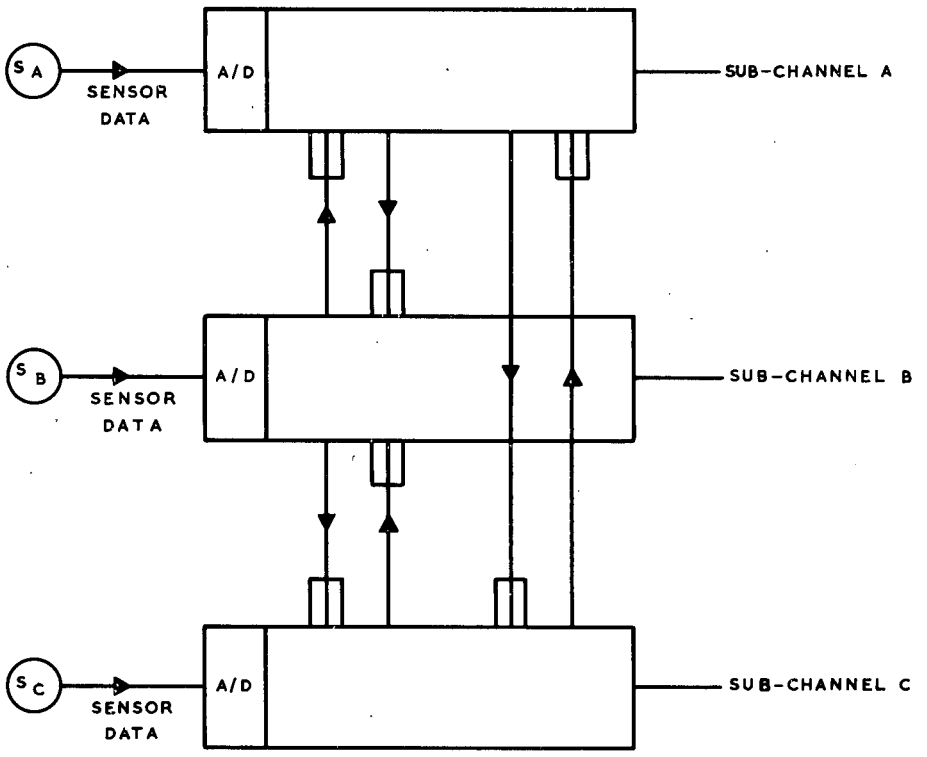
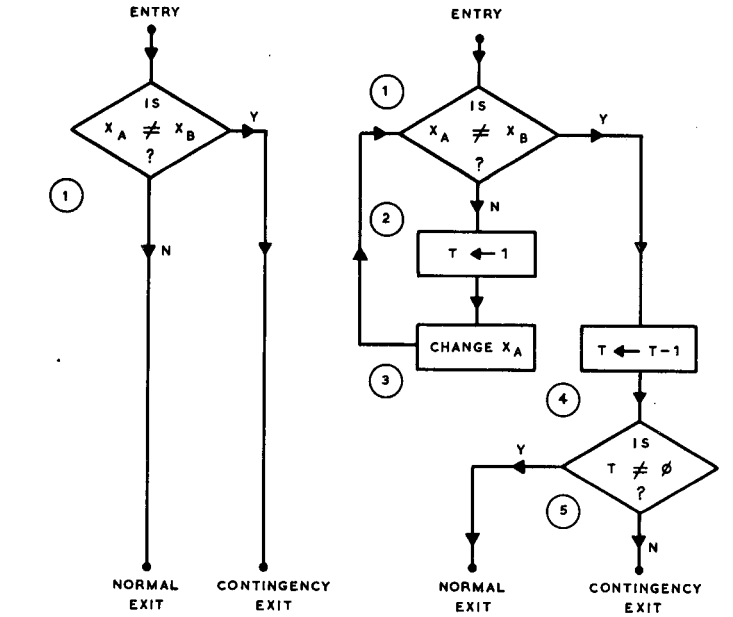


Fig. 2 Sensor data exchange via the sub-channel data highways



SUPERFICIAL FAULT ANALYSIS:

(a) ITEM FAULT	COMMENT
(a) TEST ANSWER YES	CONTINGENCY ACTION
TEST ANSWER NO	LATENT FAULT

(b) ITEM FAULT	COMMENT
(1) TEST ANSWER YES	CONTINGENCY ACTION
TEST ANSWER NO	SYNCHRONISATION ERROR DETECTED BY ANOTHER COMPUTER.
(2) T←NOT 1	CONTINGENCY ACTION
(3) X_A NOT CHANGED	SYNCHRONISATION ERROR
(4) T←NOT (T−1)	CONTINGENCY ACTION
(5) TEST ANSWER YES	CHECKED SUBSEQUENTLY
TEST ANSWER NO	CONTINGENCY ACTION

Fig. 3 Ad Hoc protection against latent faults

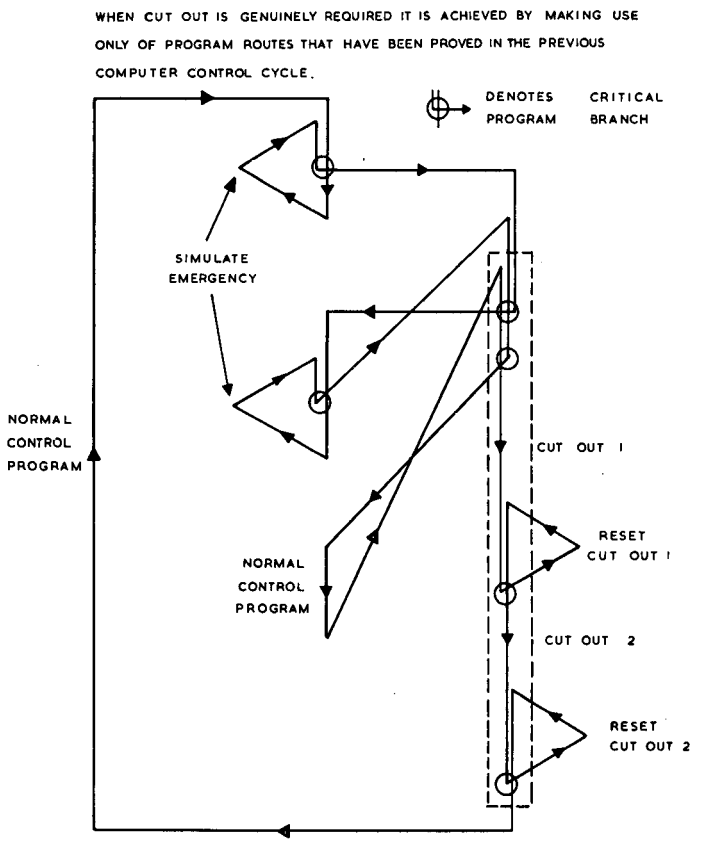


Fig. 4 Schematic example of overlapping programming

REALISATION OF NONLINEAR CONTROL METHODS WITH DIGITAL CONTROL UNITS

by

W.Sobotta and H.J.Berger

Summary

Firstly, some characteristic disadvantages of linear control methods used in conventional control units for attitude stabilization of VTOL-Aircraft will be discussed.

In order to avoid these disadvantages, different non-linear control methods have been developed.

A special non-linear SAS was successfully tested on the "flying bedstead" SG 1262 of VFW, Bremen, in order to obtain information on the improvement of handling qualities in VTOL-Aircraft. According hereto, it is advantageous when the response time is about proportional to the pilot commands. This does not only signify a more repid response of the aircraft at small commands, however, a better reducting the influences of disturbances, too.

A digital control unit was developed which at given sampling time and given maximum control power yields a response time proportional to the command about greater commands. About smaller commands response time is constant.

By choosing this proper control characteristic the fatigue of the engine delivering control power can be made as small as possible.

The control algorithms have been combined in such manner that both the commands and the disturbances are optimized. The control units are distinguished by the relative simple setup with few digital computer elements.

Realisation of Nonlinear Control Methods with Digital Control Units

Dr. W. Sobotta and Dipl.Ing. H.J. Berger

The paper gives a short review of software and hardware experience gained in the field of digital control systems at VFW-Fokker in Bremen during the last three years.

Analogue controllers generally consist of passive and active networks. The essential elements of the networks are resistors, capacitors and operational amplifiers. Naturally, most analogue controllers are therefore linear controllers whose behaviour can be represented by a transfer function. The analytical treatment of such control systems is made particularly easy by the Laplace transformation. This contributed to the fact that the first digital controllers became copies of analogue controllers by trying to realize the behaviour of lead/lag networks with digital techniques.

On the one hand, this method of digital control was very complicated and showed the necessity to develop new control methods which lead to algorithms that can easily be treated digitally. On the other hand, the experience had been made that the linear technique would by no means always afford optimum motion response.

Particularly in the VTOL sector in attitude stabilization systems the linear control techniques had disadvantageous effects. The problem arising in the stabilization of a vertical takeoff and landing aircraft is the generation of control momentums, for which two techniques are used today. Either bleed air is tapped from the engine compressors and blown out through control nozzles at the fuselage or wing tips or use is made of a thrust modulation of correspondingly arranged lift engines, i.e. if, for example, one engine each is located in the fuselage forward and rear sections, a control momentum about the pitch axis can be generated by upward control of one engine and downward control of the other. This means that the project engineer for a VTOL-Aircraft must provide for suitable engine reserves of which he cannot dispose in the thrust balance for takeoff thrust. This is why the demand for extremely low control momentums is understandable.

For this reason a non-linear control technique has been developed which allows a considerable reduction of the control momentums while further improving the controllability and stability characteristics of the aircraft.

First of all, let us briefly review the adverse features of linear control techniques. Figure 1 shows in the upper left corner a PD controller realized by means of an operational amplifier.

The transfer function of this controller is

$$\frac{\varepsilon}{\vartheta} = V_R (1 + T_V s)$$

with regard to the controlled variable and

$$\frac{\varepsilon}{\vartheta_c} = V_R$$

with regard to the control input.

The lower left-hand corner of Figure 1 shows the complete closed control loop. In this case the transfer function for the final control element has been simplified in order

to facilitate the analysis of the problem.

On the right the transient response of the aircraft attitude on step stick commands from the pilot are plotted versus time. This figure illustrates the very characteristic feature of all linear control techniques, namely that the transition time t_{95} is independent from the amount of the commanded attitude changes. The first time we experienced the disadvantageous effect of this phenomenon on flight control characteristics was during flight tests when we wanted to determine in flight the most favourable response times.

Whenever the pilot considered the response time for small attitude changes to be correct, they were too rapid for large attitude changes; whenever they were all right for large attitude changes, the pilot considered them to be too slow in the case of small changes. So what the pilot wanted was a progressive alteration of the response time proportional to the commanded attitude changes.

The low stability reserve represented the second disadvantage. If, for instance, the control momentums are limited to 0.2 rad/sec^2 corresponding to a maximum angular acceleration in the standardized form used, then the controller operates for major commanded attitude changes like an on-off controller, i.e. it is at its stops for most of the time (Figure 2). If the controller is optimized for the linear range, increasingly larger command quantities result in heavier overshoot conditions. This process can be shown to the control engineer very clearly in the phase plane. In the phase plane the change of the controlled variable in this case, the angular velocity of the aircraft is plotted versus the controlled variable, i.e. the aircraft attitude. If we deflect the system to 1 rad attitude displacement and leave it to the controller, the trajectory describing the state of the system follows a parabola which is given by the maximum control momentum in the relation to the momentum of inertia. During this time the controller is at the stop and accelerates the system. The linear range of the controller is described by the plotted straight line. This range is passed very rapidly and then the controller decelerates the system, a process which results in considerable overshooting. From this point the process repeats itself until the system settles to a steady state. The upper left-hand corner of Figure 2 shows the percentage overshoot against the command quantity at different momentum limitations. At present we operate in the range of $\ddot{\vartheta}_{max}$ = 0.6 rad/sec^2. However, it will be shown in the following how to get into smaller ranges without having to put up with an overshoot of 50 per cent.

The third disadvantage of the linear technique was that by fixing the response times, the closed loop gain was firmly fixed, too, and with it the residual error of the system in the case of constant disturbance momentums. In this case the residual error is equal to the disturbance divided by the closed-loop gain.

In order to find out how these disadvantages can be removed by means of a non-linear control technique, first a parametric study was carried out. The result of this study is shown in Fig. 3. In this case, the control parameters T_V and V_K of the PD control loop shown in Fig. 1 were varied over wide ranges and their effect on the form of the transient response function was observed. The transient response is described for a specific V_K and T_V each. We realize that the form of the transient function does not change along certain hyperbolas whereas the transition times do. In other words, relative damping of the system remains constant along this line, while the transition time increases with a decreasing closed-loop gain V_K and a growing lead time constant T_V. Higher response times require lower control momentums. So this result proves how the controller parameters must be changed as a function of the error in order to obtain

progressive response times with acceptable transient behaviour.

With reference to the analogue controller this means that the resistor or capacitors forming the external networks of the operational amplifier must be changed in relation to the error. In the present case we have changed the input resistors of the operational amplifier linearly with the error, so that actually a hyperbolic dependence T_V and V_K resulted. This correlation led to the non-linear differential equation for the closed loop behaviour.

$$\ddot{\vartheta} + R_2 C_1 \dot{\vartheta} \frac{R_2}{+ R_1 + R' |x|} \vartheta = 0$$

The variation of the resistors as a function of the error was realized by means of special multipliers, which, of course, made the controller considerably more complicated.

Fig. 5 shows the effect achieved. In the top right-hand corner it can be seen how the area in which the final control element operates in the linear range is bent over towards the x-axis. As the center line of this area represents the 0^o isoclinic line it is seen how, by means of the parameter variations, the maximum rate of attitude change cannot be increased, even in the case of larger attitude changes, as opposed to linear control techniques. The difference is also illustrated by the trajectories in the two left-hand diagrams. Whereas the maximum velocities in the linear system (bottom) constantly increase, they remain constant, even with increasing attitude changes, for the non-linear procedure. The trajectories fall within the linear operating range of the final control element as defined by the momentum limitations and therefore overshoot conditions are not possible any more. In the bottom right-hand corner of Fig. 5 the transient functions of the attitude angle are shown for different step commands and from this it can clearly be seen how the transition time changes between 0.5 to 1.5 seconds. Flight testing of the controller was carried out on the basis of these settings. Pilot evaluation was excellent.

Encouraged by the successful tests, we then attempted to actually realize the non-linear behaviour in a digital controller in order to arrive performance and complexity comparison. In doing this we have tried not to copy the analogue controller but to find out the most suitable method for the digital system.

The solution is shown in Fig. 6 in the form of a block diagram. The command signal ϑ_c is supplied to a digital command computer. The latter computes a control sequence in such a way that the process falls in time - suboptimally with the command signal value, in a similar way to the analogue controller, without overshoot or steady-state error. The computer must be informed of the state of the system to be controlled at the time of command signal lock-on. The effect of disturbances acting upon the controlled-system is determined by comparing the state variables of the controlled system with those of a model system not affected by the disturbance. The resulting difference signals are passed to a special digital disturbance computer applying to the controlled process additional singals ε_n which compensate the influence of the disturbances.

The algorithm of the command signal computer has to take into account the characteristic of the controlled system if the command behaviour is to correspond to the requirements indicated. In developing the calculation rule, the difference equation of the system to be controlled forms a good basis. Provided the controlled-system input is constant within specified intervals, the sampling intervals, then

$$\underline{x}[(k+1)T] = \underline{\Phi}\underline{x}[kT] + \underline{h}\varepsilon[kT] \qquad (1)$$

where $\underline{x}\,[kT]$ is the n-dimensional state vector of the controlled system at the time kT, $\underline{\Phi}$ a n x n matrix, $\underline{h}$ a n x 1 matrix. $\underline{\Phi}$ and $\underline{h}$ depend on the system parameters and the sampling time selected.

When the system is in the initial state $\underline{x}_o$ at t = 0, the solution is found successively from equation (1).

$$x[kT] = \underline{\Phi}^k \underline{x}_o + \sum_{i=0}^{k-1} \underline{\Phi}^{k-i-1}\ \underline{h}\,\varepsilon[iT] \qquad (2)$$

Provided the controller output is not restricted, a system of n-th order can be brought in n-steps into any desired stable end state predetermined by the command signal. From equation (2) with transponding the sum we obtain

$$\underline{x}[nT] = \underline{\Phi}^n \underline{x}_o + [\underline{\Phi}^{n-1}\underline{h} \quad \underline{\Phi}^{n-2}\underline{h} \ldots \quad \underline{\Phi}^o \underline{h}]\,\underline{\varepsilon} \qquad (3)$$

$$\underline{x}[nT] = \underline{\Phi}^n \underline{x}_o + \underline{\Psi}\,\underline{\varepsilon} \qquad (4)$$

This equation resolved in accordance with $\underline{\varepsilon}$ results in n definition equations for n control quantities.

$$\underline{\varepsilon} = \underline{\Psi}^{-1}(\underline{x}_n - \underline{\Phi}^n \underline{x}_o) \qquad (5)$$

If the system cannot be brought to a settled steady state condition in n-steps because of a given control quantity limitation, control can be achieved over several periods with constant ε . The elements of the matrix $\underline{\Psi}$ will then be composed of separate sums.

$$\underline{x}[NT] = \underline{\Phi}^N \underline{x}_o + \left[\sum_{i=a}^{N-1} \underline{\Phi}^i \underline{h} \middle| \sum_{i=b}^{a-1} \underline{\Phi}^i \underline{h} \middle| \ldots \middle| \sum_{i=0}^{z-1} \underline{\Phi}^i \underline{h}\right] \underline{\varepsilon} \qquad (6)$$

Calculation of the disturbance controller is based on the assumption that all state variables of the controlled system are known and that the disturbances can be compensated in n-steps. After every sampling step the disturbance controller is to lock on a signal ε_n to the controlled system, in addition to the command signal, which is composed of the linear combination of the state variable and dependent on the disturbance acting upon it.

Assuming the disturbance to be constant and acting from the controlled system input, the following equation system can be set up:

$$\begin{aligned}
\varepsilon_n[T] &= \alpha_1 \vartheta_n^{(n)} = F_1 x_{1n}[T] + \ldots + F_n x_{nn}[T] \\
&\vdots \quad\quad \vdots \quad\quad \vdots \quad\quad \vdots \\
\varepsilon_n[(n-1)T] &= \alpha_{n-1} \vartheta_n^{(n)} = F_1 x_{1n}(n-1)T + \ldots + F_n x_{nn}[(n-1)T] \\
\varepsilon_n[nT] &= \alpha_n \vartheta_n^{(n)} = F_1 x_{1n}[nT] + \ldots + F_n x_{nn}[nT]
\end{aligned}$$

With the last but one equation the system must pass over into a steady state. If all the following intervals are controlled with the last equation, the system remains in the steady state. The equation system written in matrix notation results in

$$\underline{\alpha}\ \ \vartheta_n^{(n)} = \underline{x}'_n \cdot \underline{f} \qquad (7)$$

The matrix $\underline{X}'_n$ can be set up with the aid of equation (2). It is dependent on $\underline{\Phi}$ and $\underline{h}$ and each element contains the value $\vartheta_n^{(n)}$. Therefore, from equation (7) we obtain

$$\underline{\alpha} = \underline{x}_n \cdot \underline{f} \qquad (8)$$

The unknown matrix $\underline{\alpha}$ can be computed from equation (2) on the basis of the steady final state. Thus, the disturbance controller constants $\underline{f}$ can be determined from equation (8).

For reasons of simplification the above described method will now be applied to the controlled system $F(s) = \frac{1}{s^2}$ as given, for example, by VTOL-Aircraft. The resulting control system is shown in fig. 7. The disturbance controller constants are represented by the quantities $F_1 = \frac{1}{T^2}$ and $F_2 = \frac{3}{2T}$. Transient response functions for different command signals and an associated control momentum such as the control law are shown in Fig. 8. On the basis of the equations 5 and 6 the command algorithm was modified in such a way that a selected minimum response time (T_E = 1 sec) must be reached in every instance. If for a larger commands the control momentum required for the specified response time becomes too high, the response time is correspondingly increased.

Fig. 9 shows transition events in the case of disturbances being encountered. A steady-state error proportional to T^2 results.

As from its concept the control system is also adaptive, disturbances occuring within the system to be controlled or system changes that cannot be covered are corrected, too, without the command or disturbance controller algorithms having to be changed. Fig. 10 shows a comparison of a transient response function, with a command algorithm exactly adapted to the controlled system, and a transition trend in the case of controlled-system gain by 40 per cent.

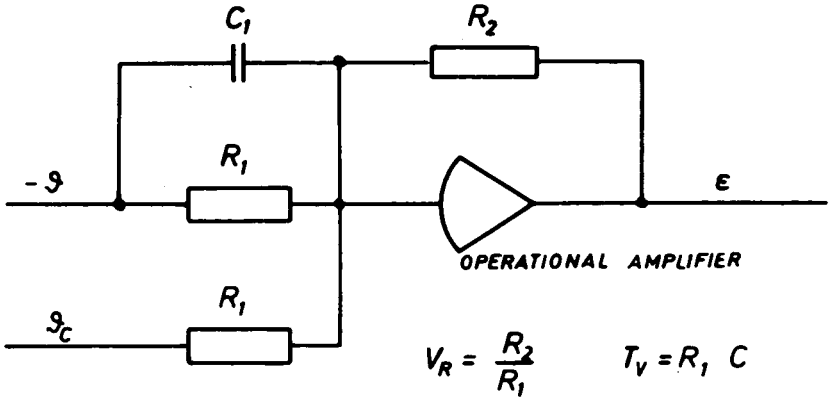
From the explanation you can see, that the digital controller has the same performance as the analogous non-linear controller. By varying of the minimal response time and of the limited control momentum you can also have any degressive controller-characteristics. As the system is adaptiv too, the plant must not be exactly known.

The constructional costs - in view of the standard of digital techniques - are not higher than those of the analogous non-linear controller.

It is also important, that the here developed theory is very simple and therefore easy to use for general systems of n-th order.

Notation

$\underline{f}$	Vector of noised control unit
$\underline{h}$	Vector
i	Integer
k	Integer
s	Laplace-operator
T	Sampling period
T_E	Response time
T_V	Lead time
V	Gain
V_R	Gain of non-linear controller
$\underline{X}$	State variable
$\underline{X}_n$	Noised state variable
$\underline{\alpha}$	Vector
ε	Output of the controller for commanded input
ε_n	Output of the controller for noised input
ϑ	Output of system
ϑ_c	Input of system
$\underline{\phi}$	Matrix
$\underline{\psi}$	Matrix



$$V_R = \frac{R_2}{R_1} \qquad T_V = R_1\ C$$

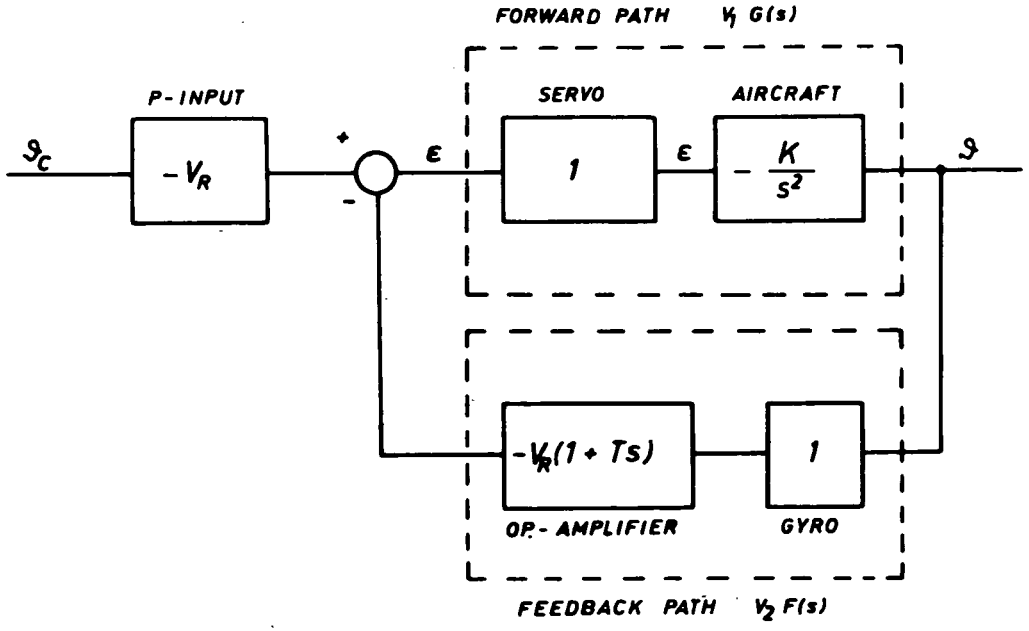
TRANSFER FUNCTION

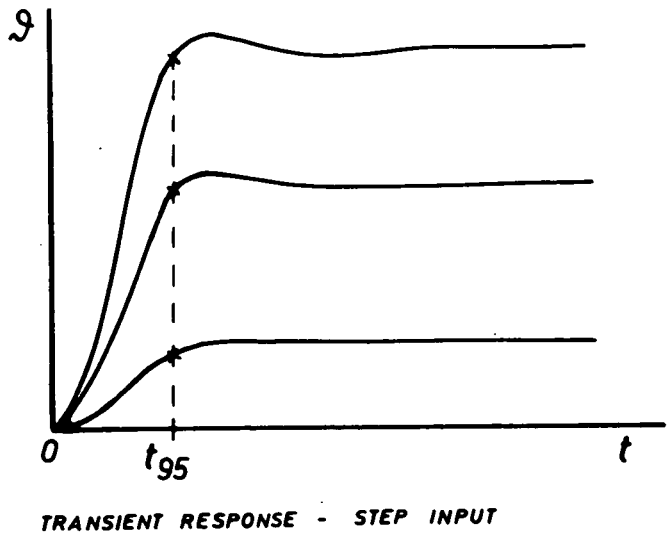
CONTROLLER: $\frac{\varepsilon}{\vartheta} = V_R\ (1 + T_V s)$

$$\frac{\varepsilon}{\vartheta_C} = -V_R$$

CLOSED LOOP:

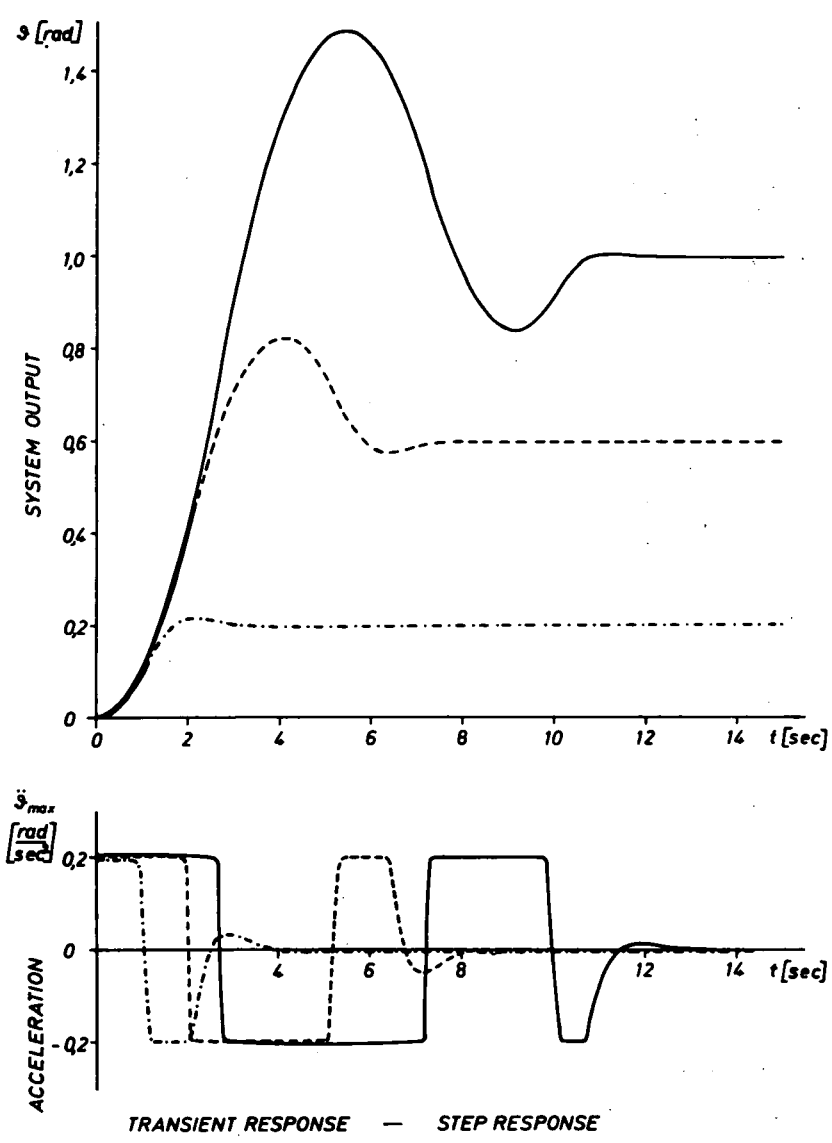
$$\frac{\vartheta}{\vartheta_C} = \frac{V_K}{s^2 + T_V V_K s + V_K}$$

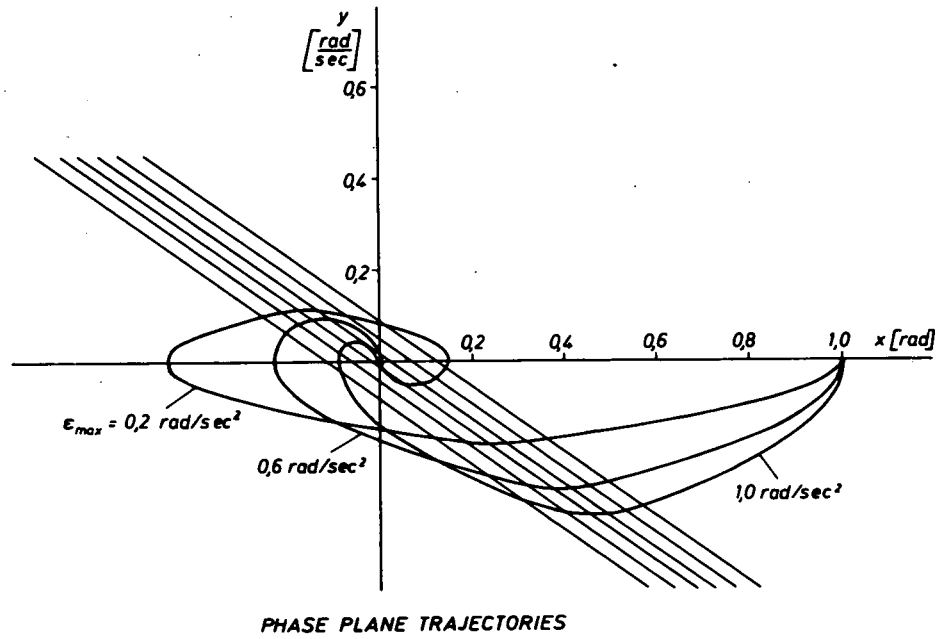




TRANSIENT RESPONSE - STEP INPUT

FIG. 1 LINEAR CONTROL SYSTEM - PD-CONTROLLER





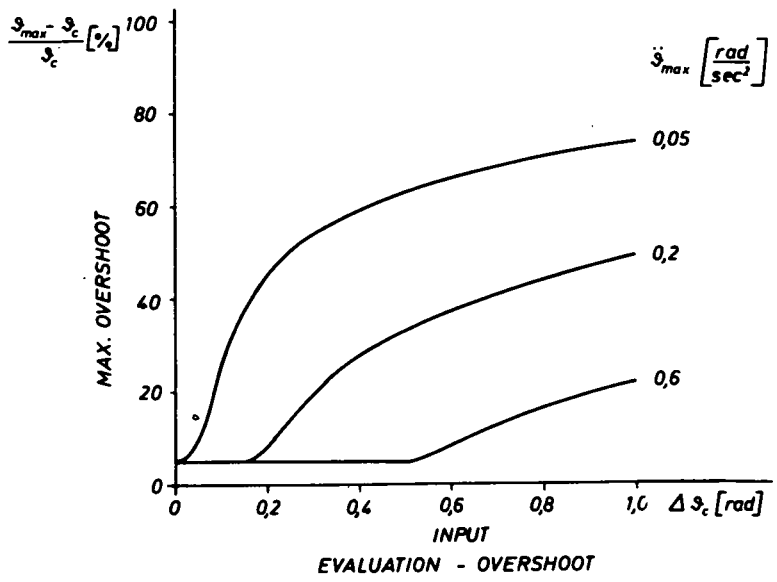


FIG. 2 LINEAR CONTROL SYSTEM - TRANSIENT BEHAVIOR WITH LIMITED CONTROL MOMENTUM

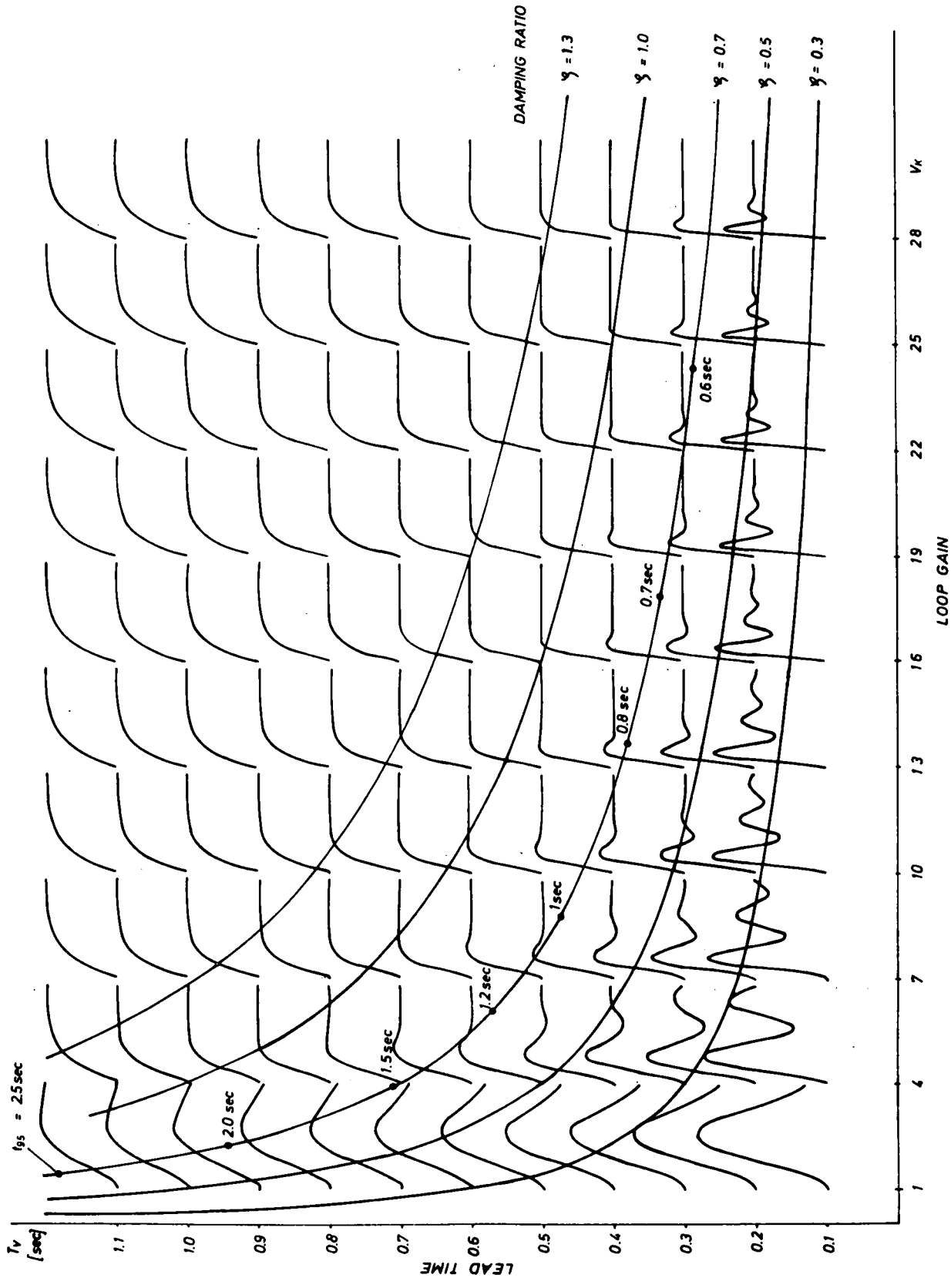
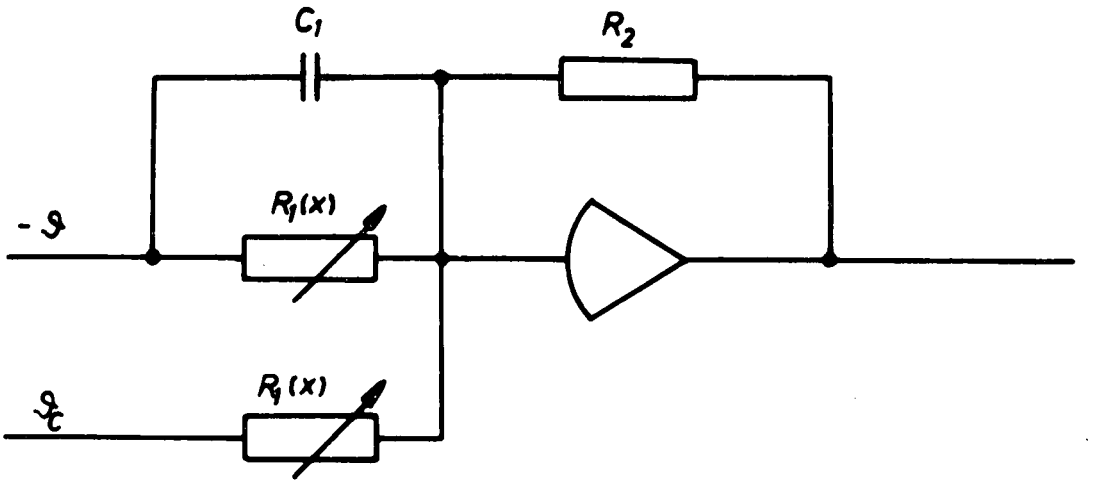


Fig. 3 TRANSIENT RESPONSES AT VARYING PARAMETERS



$$V_R = \frac{R_2}{R_1(x)}$$

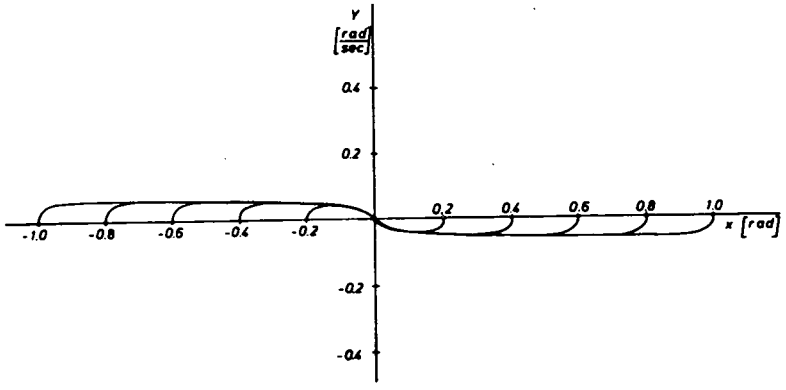
$$T_V = R_1(x) \cdot C_1$$

WITH $R_1(x) = R_1 + R'|x|$

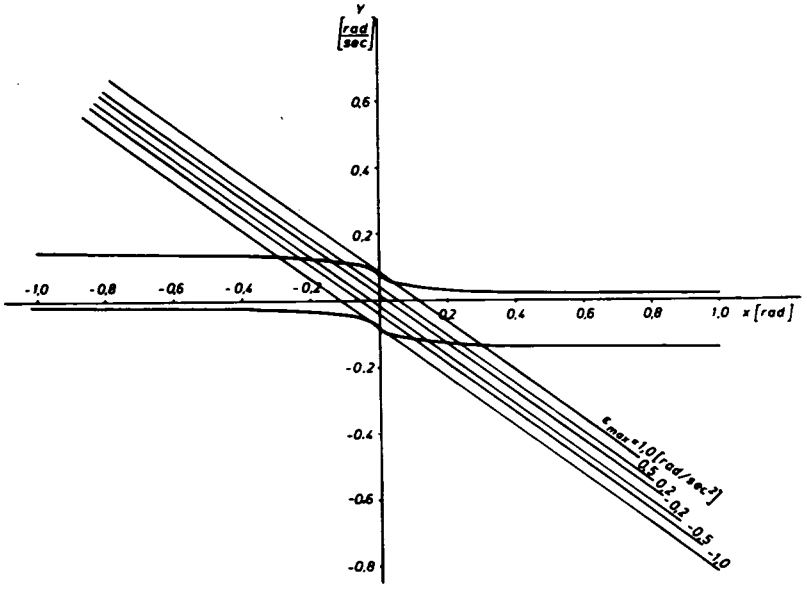
DIFFERENTIAL EQUATION OF THE LOOP :

$$\ddot{\vartheta} + R_2 C_1 \dot{\vartheta} + \frac{R_2}{R_1 + R'|x|}\vartheta = 0$$

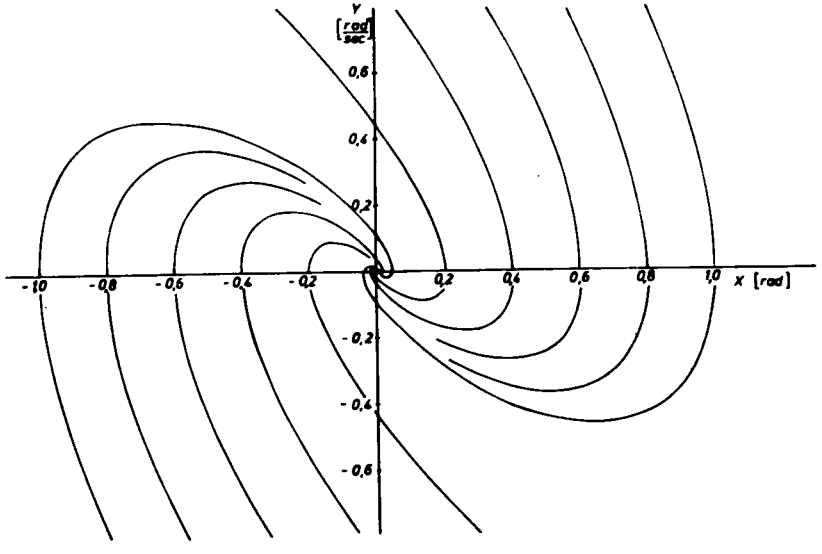
FIG. 4 NON-LINEAR CONTROL SYSTEM



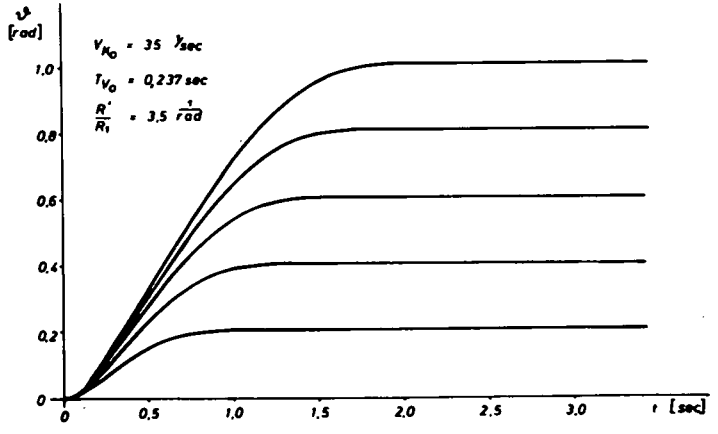
TRAJECTORY OF THE NON - LINEAR SYSTEM



AREA OF THE CONTROLLER IN THE PHASE PLANE - COMPARISON LINEAR – NON-LINEAR



TRAJECTORY OF THE LINEAR SYSTEM



TRANSIENT RESPONSE – STEP RESPONSE

FIG. 5 NON-LINEAR CONTROL - TRANSIENT BEHAVIOR

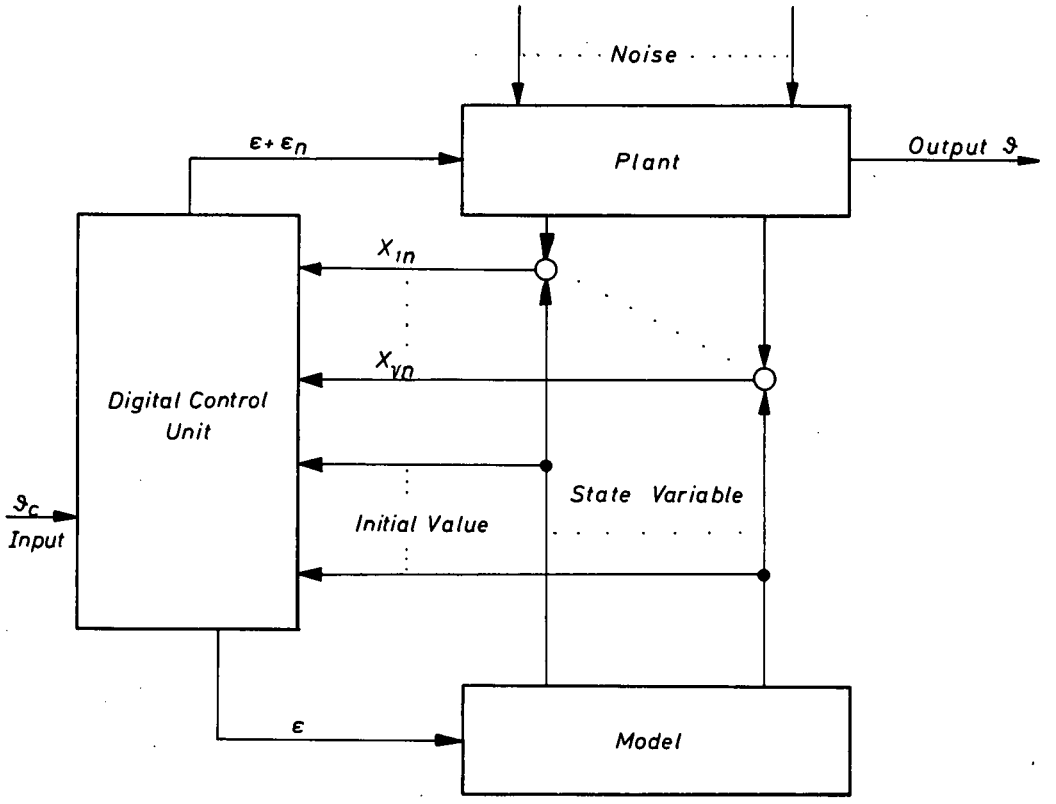


Fig. 6 Basic block diagram of digital control system

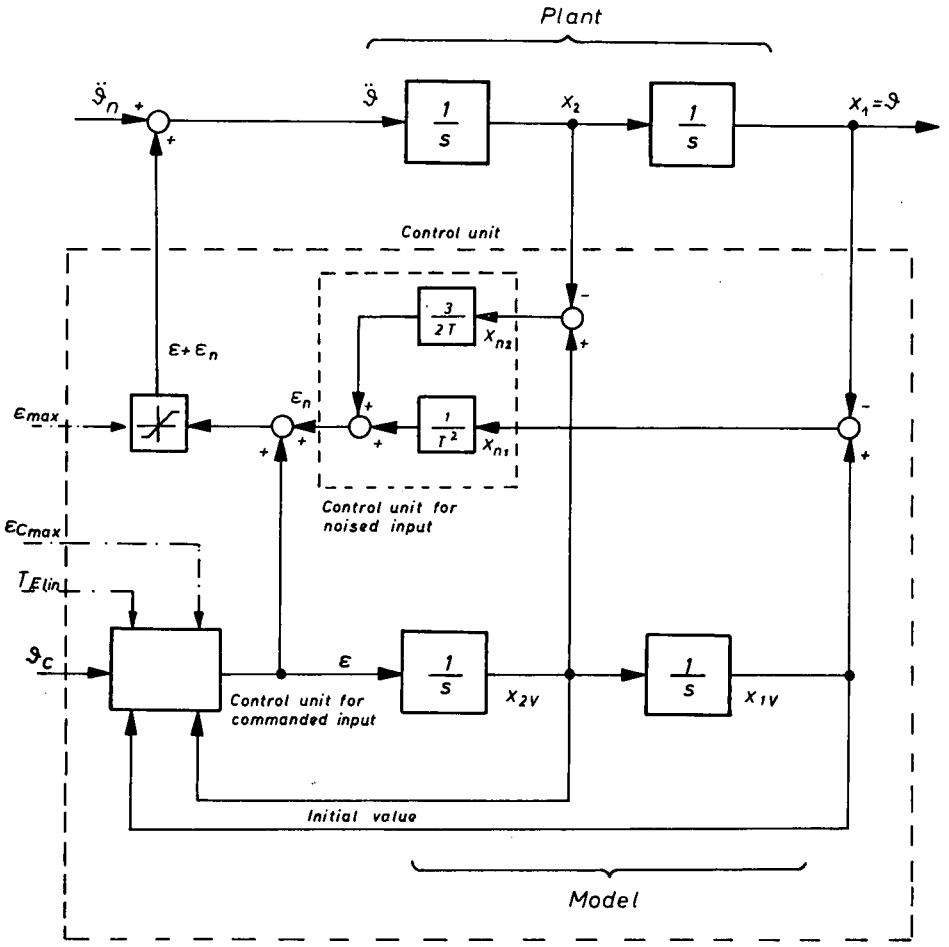


Fig. 7 Special digital control system

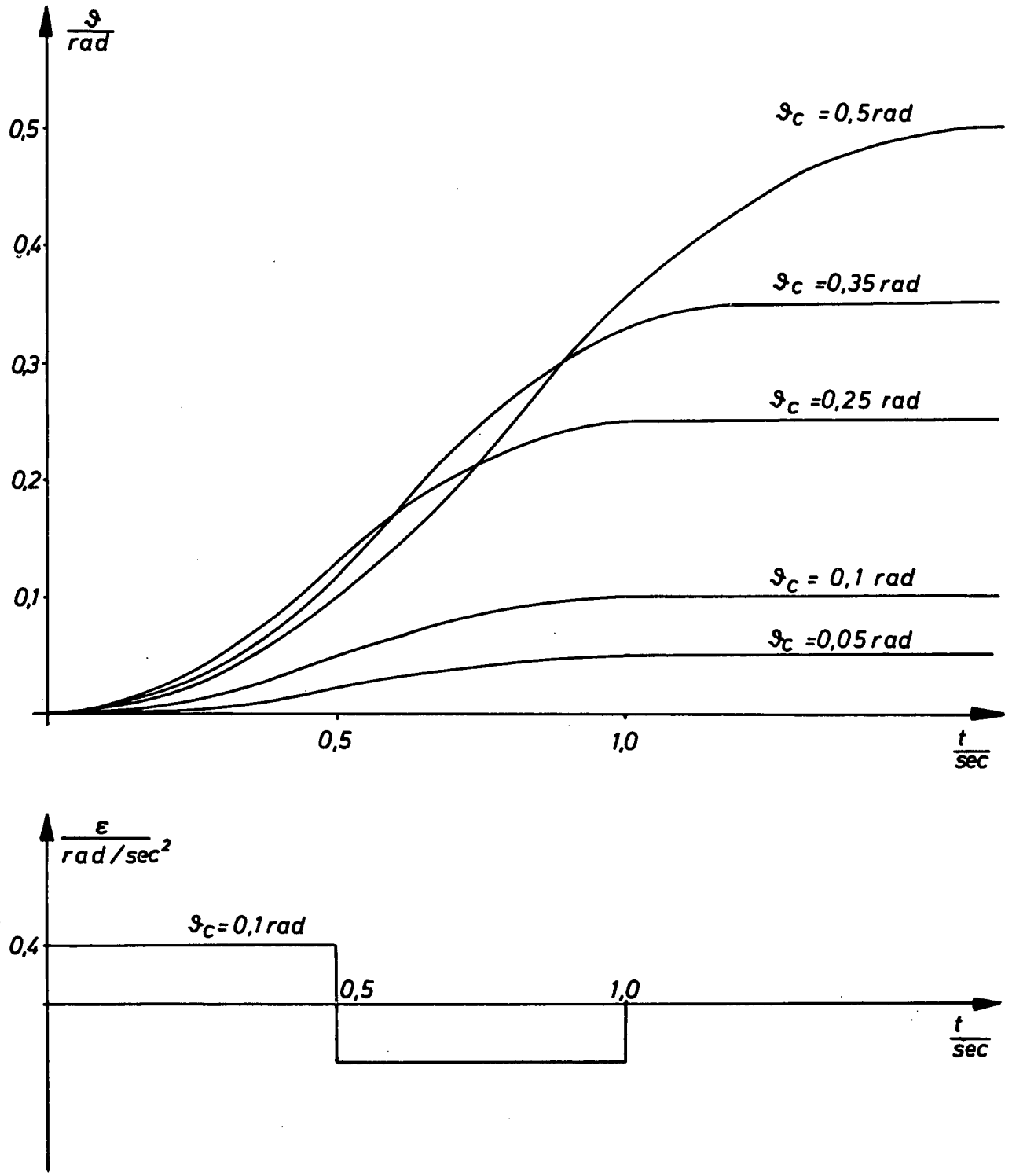


Fig. 8 Step response and control power of the system

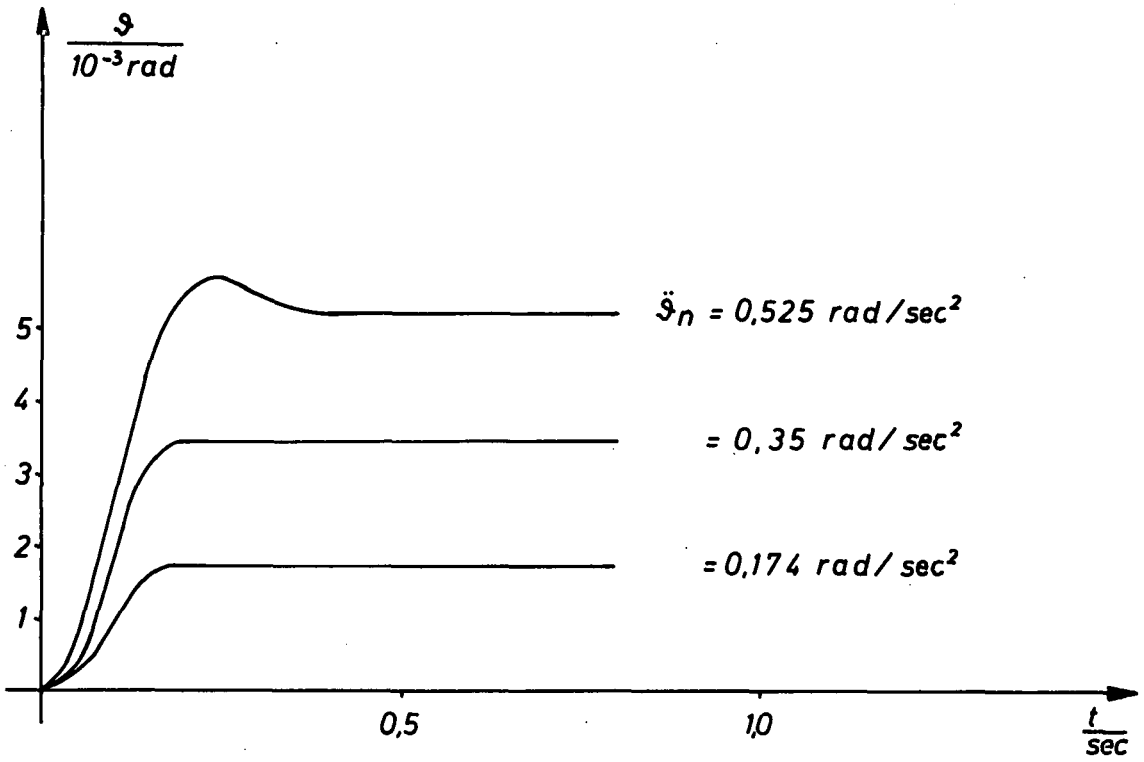


Fig. 9 Step response at different disturbances

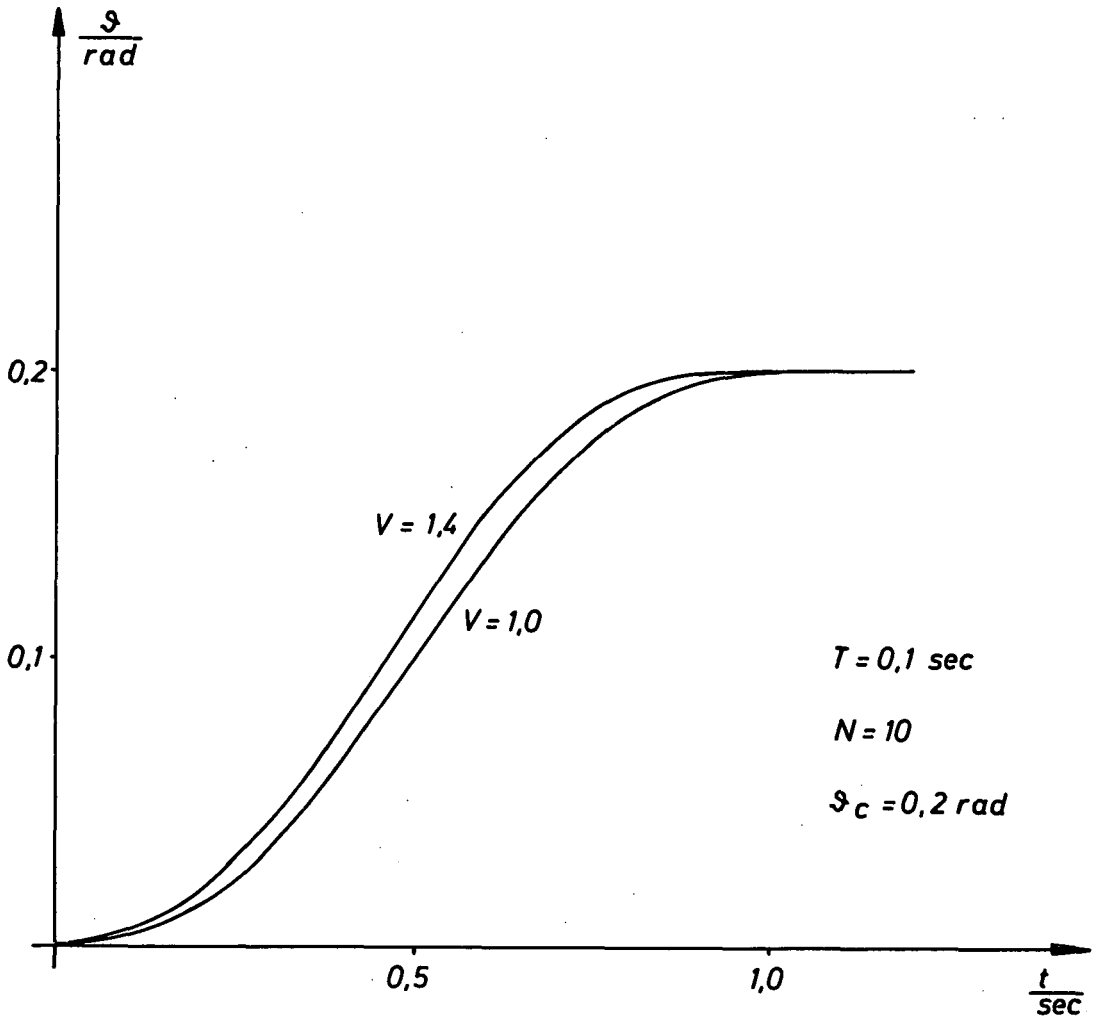


Fig. 10 Step response at varying parameters of system

DIGITAL DATA PROCESSING IN AUTOMATIC FLIGHT CONTROL SYSTEMS

R. W. Howard.
Elliott Flight Automation Ltd.,
Airport Works, Rochester, Kent.

SUMMARY

Digital data processing is examined as another technique for the designer of automatic flight control systems. It is discussed in relation to three general aspects of all AFCS, those of inner loop, outer loop and executive functions. Various types of modern aircraft system are compared in assessing the impact of digital techniques and it is concluded that although it is not cost effective for all computing requirements at present, it will progressively replace analogue flight control computing in the future. A warning is given against adopting central computing techniques as a means of speeding this process.

DIGITAL DATA PROCESSING IN AUTOMATIC FLIGHT CONTROL.

1. INTRODUCTION

Until recently all computing and associated aspects of automatic flight control have been analogue in nature, and systems have varied considerably in design and complexity depending principally on the application, the number and types of modes of control provided, and the degree of redundancy required. Indeed, requirements are as diverse as aircraft types and there will certainly be little change in this situation in the future.

It might therefore be expected that the impact of digital data processing, as a new technique in the armoury of a system designer, will also vary considerably for different AFCS applications.

Hence, if an attempt is to be made to assess the effect of digital techniques over a wide range of applications, it will be necessary to select some common functions from the various types of AFCS and to consider these in turn for each application.

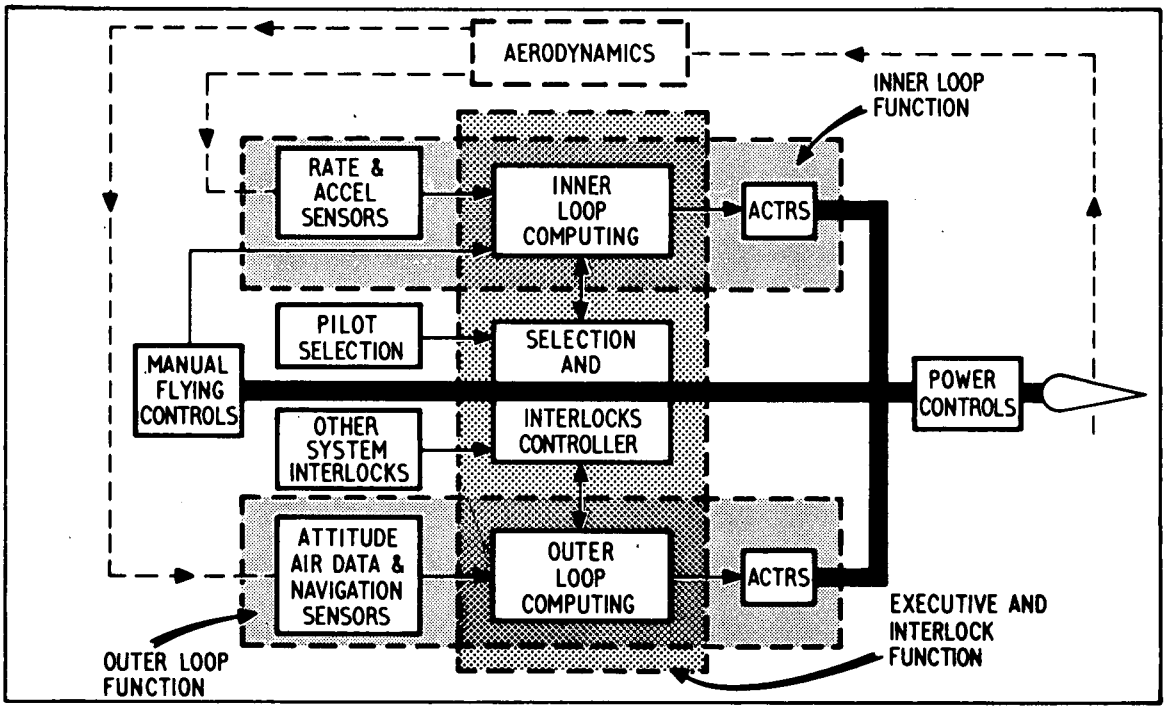
Three such areas of technical commonality suggest themselves:-

(i) The inner loop system involved with aircraft stability and command control which probably has a relatively fast response, and is driven from angular rate and acceleration sensors and possibly also sensors in the pilots' controls.

(ii) The outer loop system which implements a number of alternative control configurations such as height and speed lock, attitude hold, auto-attack, terrain following, automatic navigation, automatic landing etc., effectively giving three dimensional control of the aircraft and obtaining its inputs from air data, attitude sensors, radio, radar, nav/attack systems etc.

(iii) The executive sub-system which responds to pilot selections, controls the interconnection of the outer loop components, schedules inner and outer loop gains and time constants, and provides necessary interlocks to reduce pilot workload and to ensure the engagement of safe combinations of control functions. Logically this area also embraces any integrated self-test function.

The functional relationship of these three areas is illustrated in Figure 1 and their general properties are compared in the table of Figure 2.



FUNCTIONAL RELATIONSHIP OF THREE AREAS OF AN AFCS.
Figure 1

It is the main object of this paper to comment upon the impact of digital techniques on various AFCS applications, taking as examples combat/strike stability augmentation and terrain following, VTOL jet lift stabilisation, SST stability augmentation and civil transport automatic landing, and considering the above three areas of commonality in each case.

Before doing this, however, it is worthwhile to highlight a number of the relevant characteristics of digital computing, as compared with analogue computing. Most of these are well known, but they are worth reiterating for completeness. In addition, a short review of the present problems and trends in airborne

digital data processing is desirable in order to isolate from the specific assessment of automatic flight control systems, characteristics relevant to all digital avionics. These aspects are covered in the next two sections.

PARAMETER	INNER LOOP	OUTER LOOP	EXECUTIVE COMMAND
Operating Speed	Fast	Slow	Logic only
Input Data	Mainly analogue at present	Analogue and Digital	Pilot selection
Accuracy	Varied requirements	High	Not applicable
Output format	Mainly analogue at present	Mainly analogue and digital	Logic only
Need for ease of modification	After flight testing and later configuration changes.	Occasional; change of role or updating system capability; customer selection.	Desirable, for improvement in human engineering, customer option; system updating.
Degree of integration with other aircraft systems, including digital developments.	Minimal	Increasing	Desirable

GENERAL PROPERTIES OF THE THREE AREAS OF AN AFCS

Figure 2

2. RELEVANT CHARACTERISTICS OF DIGITAL v. ANALOGUE COMPUTING.

It is not the intention here to discuss the basic technology of digital versus analogue computing, other than where there may be variations of special use in the AFCS field. However, the reiteration of certain well known characteristics is worthwhile as background for the AFCS assessment.

The relevant characteristics may be summarised as:-

(i) Computation.

There is a lower level of hardware, of significance economically, below which a single digital calculation cannot be performed. This is especially true of the general purpose computer, as distinct from the D. D. A. The hardware costs to perform the simplest single analogue computation are negligible in comparison. On the other hand the time shared nature of a digital processor enables more complex calculations to be carried out, if necessary, without the corresponding increase in the complexity of the hardware required by analogue computing.

(ii) Accuracy.

The accuracy and resolutions of a digital calculation can be readily controlled, since they are a function of word length and programme organisation, both of which are controllable during design and development, rather than being inherent characteristics of the basic components used as in analogue mechanisations. On the other hand the decisions made in determining the design of a digital processor may impose an ultimate limit on the extent to which a new requirement can be met, especially in matters of speed of operation. This may be so even if spare capacity is included in the first instance. This is not so much the case with analogue computing.

(iii) Drift.

Digital computation presents fewer problems of "drift" and "bias", because of the numerical nature of the data being processed, than simple analogue computation.

(iv) Logic.

The logical characteristics of a digital processor enables switching and interlock operations to be programmed without the necessity for design of additional special complex logic arrays. Once a given degree of complexity is required, the problems of special logic hardware designs and the associated time consuming "failure analysis" of specialised arrangements which, for example,

are inherent in analogue flight control systems can be reduced. This assumes that any interconnections and logic design external to the computer are compatible.

(v) Storage

In terms of cost effective hardware, digital computer storage devices are much more efficient than analogue storage devices. On the other hand the digital processor, by its nature, generates its own storage requirements, and expands the size of this element in a control system. Nevertheless, there is undoubtedly an advantage for digital computers in this respect.

(vi) Future Development.

Digital computer elements, in various standard microcircuit forms, are being widely applied and used in very large numbers. This makes for dramatic reductions in cost and size of future computers, and hopefully improvements in reliability and functional capability. The comparison with analogue equivalents is a continuously changing activity, but the greater pressure on digital components is undoubtedly causing a rapid movement to the more extensive use of this technology.

These are then the basic characteristics which must be assessed when considering the impact of digital technology on automatic flight control systems. Like all new technological developments, digital computing will not be the panacea which will eliminate all of our problems, and careful balancing of all available technologies, including this newer one, is necessary to ensure the best, most cost effective solution for any requirement at any time.

3. CURRENT PROBLEMS AND TRENDS IN AIRBORNE DIGITAL DATA PROCESSING.

Digital techniques are now being used widely in certain airborne applications and the current experience is of interest in assessing the probable uses in automatic flight control systems.

The problem probably revolves around four main aspects. These are:-

- The cost and integrity of memories.
- Data transmission.
- Checking and monitoring.
- Interfacing with analogue devices.

Storage devices have up till now been the area of the computer most resistant to price reduction and have presented a large number of problems. Considering first of all a conventional core stack, a typical price is of the order of 10 U.S. cents/bit. In addition, memory corruption can take place due to power supply transients, lightning strikes, etc. For high integrity a fixed program memory is thus required and until recently the rope core memory has been the best method available and the cost/bit of this method is about 3 U.S. cents/bit. M.O.S. read-only memories promise to take over this role but to achieve the full potential of these devices a compatible random access memory is required. This enables programming to be carried out readily in development and be fixed in production. The silicon gate M.O.S. random access memory cell gives us the required compatibility and these combinations will enable more economical storage to be achieved.

It is possible conveniently physically to separate processor storage into fixed program memory, "alterable constant" memory and working space for intermediate products, and to take special steps to protect the volatile workspace memory from transient corruption. In fact, the future holds out great promise for a number of compatible storage techniques with a wide variety of convenient properties. In particular, several methods for easy re-programming of non-volatile media are rapidly being developed.

New methods of using current semi-conductor materials, more effective methods of design and mass production applied to read-only memories, and development in amorphous semiconductors all offer promise in cheaply-altered non-volatile storage.

Digital data transmission presents problems in a high integrity environment again largely due to noise corruption of the data. Parity checks enable some of these errors to be detected, but essentially multiple redundant digital data transmission is the only practical solution. However the imposition of such a solution on all data handled by a system may escalate the cost and weight of a complete installation to an unreasonable degree. Sealed optical data links may offer considerable advantages in integrity, due to their lack of susceptibility and generation of electromagnetic interference, but the use of such a technique is in its infancy and again a multiple system would probably be required.

Speaking generally the checking and monitoring of continuous on-line operations using digital computing appears to be no simpler than with analogue computers, despite early hopes to the contrary. The concept of a computer which "checks itself during a spare moment" turned out to be a computer with a greatly increased memory and a software problem more extensive than the hardware on which self-monitoring is required to overcome.

More specifically, the problem of self-monitoring a digital computer to a level required for automatic landing requires a self checking capability of at least 99.99%, and to achieve this requires

extremely extensive hardware and software so that the cost of the computer and its program in the worst case can almost double. The price which has to be paid is in additional store capacity (for monitor programs and extra organizational software) and program cycle time.

Finally there is the problem of interfacing with analogue input and output devices. This can have the most crippling effect on the cost effectiveness of digital computing in any system. In automatic flight control systems in the recent past the preponderance of analogue peripheral devices has had such an influence that for certain cases it has rarely been sensible even to examine the introduction of digital computing. Fortunately, a great deal of pressure is now being applied to the development of digital-output sensors which do not rely upon expensive A to D converters and this should considerably improve the viability of future applications.

4. INNER LOOP CONTROL FUNCTION

The inner loop control functions of an AFCS are characterised generally by:

- Close involvement with aircraft flying characteristics.
- Relatively high bandwidth.
- Predominantly analogue outputs from aircraft motion sensors - at the present time anyway.
- A demand for high integrity.
- Minimal requirements for changes of mode or response characteristics.

All of these aspects apply, for example, to the high speed combat/strike aircraft which may employ multiple axis stability augmentation. If this is required over a wide range of its flight envelope it may also be multiple redundant. The aircraft short period frequences may vary from 0.5 to 3 Hz and elaborate structural filtering may be required to avoid the excitation of dangerous structural oscillatory modes. The simplest current analogue systems may achieve the requirement with computing hardware weighing around $1\frac{1}{2}$ lbs. per axis of control. i.e. a four axis simple system might weigh around 6 lbs. and a four axis duplex or self-monitored system might weigh around 12 lbs. (c.f. a two axis Harrier autostabiliser, including self test and rate gyros, weighs $5\frac{1}{2}$ lbs.)

We now wish to see how the combat/strike stability augmentation task could be performed digitally. If we consider a computer with an average instruction time of 4 uS a practical four axis program (pitch rate, roll rate, yaw rate, height rate) will take 4.5 m S per iteration. Stability analysis indicates that from consideration of phase lag and amplitude shape of notch filters, an iteration period of 10 m S must not be exceeded. Hence, the task is feasible digitally from timing considerations. 370 words of storage are required and such a "four axis" computer might weigh 15 lbs. If the computer is required to be self-monitored, using existing techniques, the time per iteration might easily increase to 15 m S, and a store 3 times larger might be required. It is therefore clearly a better approach at present to use separate redundant computers if fail-safety is required. However we already have then 30 lbs. of digital computing and this hardly compares with the 12 lbs. in the analogue form - and interface aspects have yet to be included.

There is also the problem that to make the assessment as reasonable as possible, one computer is assumed to handle concurrently 4 separate axes of control. Detailed analysis of the actual implementation will, however, probably show this to be quite unacceptable, as a single processor failure will put out of action 4 control axes at a time. Even if adequate duplicate redundancy were involved, such a failure would be startling, if not dangerous.

Hence integrity, weight and cost remain the overriding factors, not computing performance capability. The situation is likely to change, of course, as digital components become cheaper, but despite the considerable decrease in component costs over the past year the "cross-over" point when autostabilisation becomes cheaper using digital techniques is probably still some way off. When this does happen, even further advances in autostabiliser capability will be realised as digital versions will have an accuracy and sense of freedom from computer generated drift which will greatly enhance aircraft performance capability and allow high safety as a result of low interchannel disparities in redundant combinations.

Similar conclusions were drawn from a recent investigation of VTOL jet lift stabilisation, with the difference that the problem is even more difficult to solve from the integrity viewpoint. Specifically, using analogue electronic hardware, reliabilities were near to acceptable for military VTOL types, but probably still unacceptable for civil transport types.

The SST autostabilisation problem is similar to the combat/strike one, except that in general much more redundancy is required. Again, the analogue solution must be the better at the present time for the actual control loop. However, experience is showing that variations is gain scheduling, especially late in a development program might warrant a more sophisticated treatment than at present provided by specialised internal discrete switching arrays. More is said about this in Section 5.

The inner loop natural frequencies in a civil transport automatic landing system are an order lower than for a combat/strike autostabiliser system and it might, therefore, be expected that such control presents no basic timing problem to a digital computer. In fact investigations of such systems have been

conducted around an existing computer having a4000word store and using iteration rates of 40 per second. The program takes about 3000 words and with an add time of 3 microseconds the computer is about 70% utilised. This is without monitoring, and as was discussed in Section 3, monitoring increases the program length and time requirements considerably, especially considering the confidence levels required for all-weather landing. It is therefore concluded that at this time digital inner loop control during automatic landing is not cost effective as the computer required would involve at least double the hardware of existing analogue devices.

It appears, therefore, that the use of non-time-shared digital processors are generally very unattractive at present for inner-loop-aircraft control functions, and even with time sharing, for more complex systems, practical considerations of integrity will also still override their use. Also, at present, the predominance of analogue sensors and actuators maximises the interface problem. It seems unlikely therefore that digital computing will provide the best solution for inner loops in the near future.

5. OUTER LOOP CONTROL FUNCTION

The outer loop control functions of an AFCS are characterised in general by

- Relatively low bandwidth
- Inputs from sensors many of which are currently digital (air data, automatic navigation) and many of which may be digital at negligible cost differential (VOR, radio altimeter)
- Varied demands for high integrity
- Requirements to operate in a large number of different modes in different phases of flight, with control of response characteristics according to aerodynamic parameters.

The combat/strike aircraft, having an autopilot system of moderate complexity (i. e. heading lock with turn coordination, height lock autothrottle and Nav /attack coupling) using, as in the previous example of section 4, a computer with an average instruction time of 4 u S will take, according to our analysis, 2 mS per iteration, or 18 mS if completely self-monitored. Stability analysis indicates that 200 mS would be tolerated, so it is clearly possible to have a much slower computer from timing considerations alone. In fact a change from the normal parallel to a serial arithmetic unit and various store modifications (solid state instead of core) can be contemplated. These actions might lengthen the average instruction time to say 30 uS but this might still be tolerable. Such considerations are worthwhile in assessing first whether digital computing is economically sensible for a given application, and if so, whether separate computing is feasible, or whether integration with other computing functions is more desirable. Normally autocontrol integrity requirements will rule against integration, but this can be tempting if the task is simple.

Investigations show that using a slow speed special to purpose computer, digital and analogue designs for combat/strike requirements are roughly comparable in reliability, cost and weight.

When the requirements are extended to include a broader control envelope such as is demanded by the hover mode additions of VTOL attitude demand control or similar command functions, the slow speed special-to-purpose digital computer will undoubtedly have some advantages over analogue designs.

In fact once computers in the two technologies are comparable in weight, cost and reliability and from the timing viewpoint, the basic accuracy, low drift and other desirable feathres of digital computation can consolidate overall advantages.

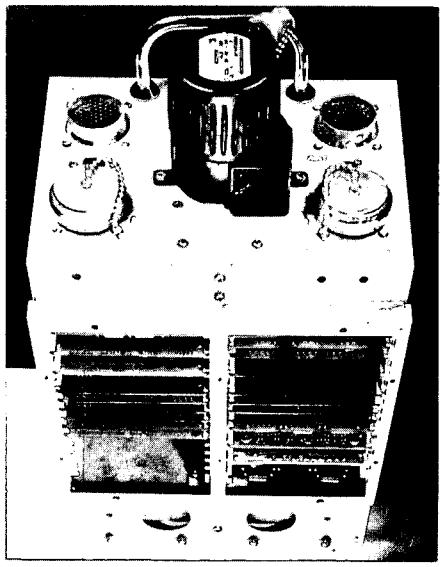
The problems in development of SST flight control systems, and other civil transports employing very complex outer loop control systems such as automatic landing should be greatly reduced by the use of digital techniques.

Those who are involved in the AFCS business will know without conducting any investigation or analysis of the past that the problems which dwarf all others in cost and complexity are the modification and retrofit programs just prior to or after an aircraft goes into service. This alone can consume 25 to 40% of the total cost of the development program for a complex equipment like failure-survival civil automatic landing. The same applies to combat/strike types and the potential problems on aircraft of the VTOL jet lift transport type, if present day systems and aircraft control design methods were adopted, may be insurmountable.

The fundamental difficulty arises from the close dependence of an AFCS on aircraft performance and flying controls characteristics which may not be adequately known until late in a flight test or certification proving program. Indeed until aircraft designs are configured around "control" rather than "handling" criteria the situation will not change. Last minute discoveries in flight testing can reflect themselves extensively through the detailed design of a system and the problem of acquiring the necessary materials for modifications can alone cause considerable cost increases or substantial delays.

There is considerable evidence that this situation can be improved by using digital techniques which can be made to lean more heavily on software rather than hardware when the modification crisis arrives. In the same way the inevitable modifications which AFCS undergo during their service life, for a

multitude of reasons, can more easily be handled if a good relationship is established in the basic design between hardware and software aspects. Figure 3 shows a digital computer, currently in large scale production for Head Up Displays which has just this sort of capability. A plug in 2000 word read-only memory incorporates the complete definition of display symbology. Alternatives can be inserted in a few seconds to change completely the display characteristics. This is possible also with an analogue computer, but is very much more complex to achieve.



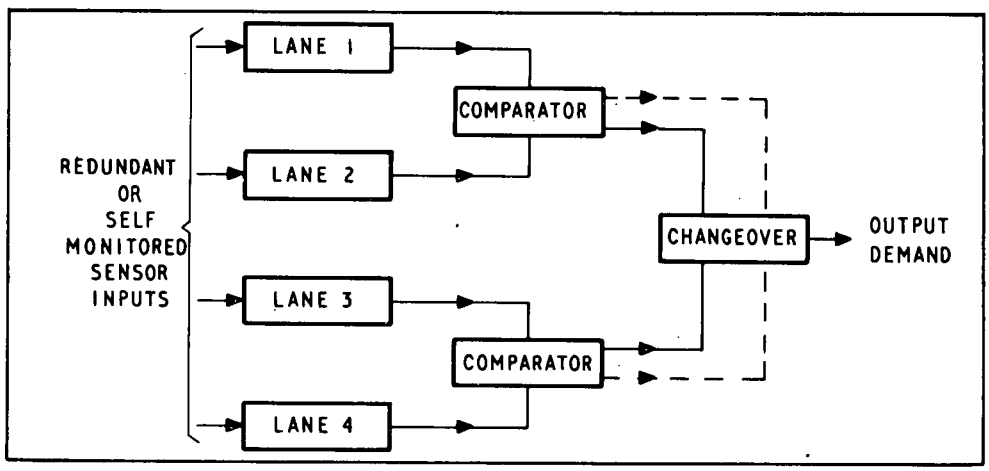
Electronic Unit for Head Up Display System 2048 word, 18 bit store 1 u sec cycle time. Now in very large quantity production.

Figure 3

This potential for simplifying major modification programs must undoubtedly be rated as the greatest potential advantage which digital techniques can offer at the present time. The advantage arises mainly with very complex systems and should be a major consideration at the design stage along with normal performance, environmental and other assessments.

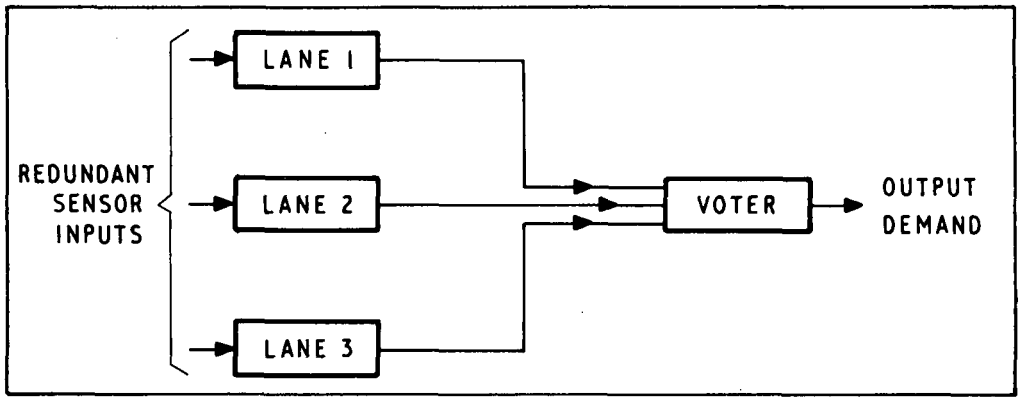
Although for outer loop controls in complex systems such as automatic landing, the digital processor looks feasible from weight, cost and timing considerations, there is still the problem of integrity. A great deal of work has been devoted over the last decade to the development of safe redundant analogue systems and it is not clear that the results of this will be directly transferable.

However, we may usefully compare the techniques used in modern analogue systems with their related concepts in digital systems. For a single-failure fail-operative requirement there are three main analogue solutions in current use, which are a dual-dual configuration with changeover (Figure 4A), a triplex voting arrangement(Figure 4B)and a duplicate self-monitored arrangement (Figure 4C).



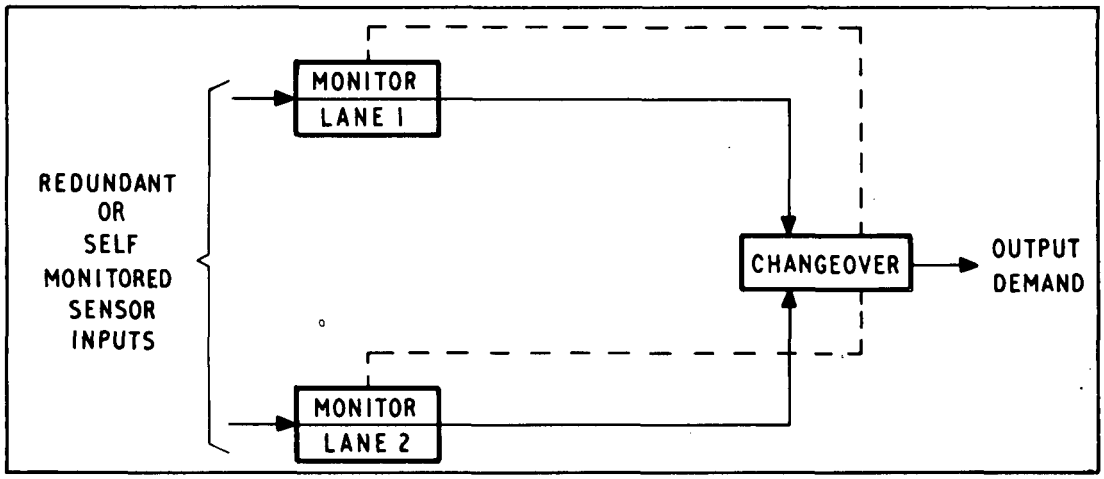
DUAL-DUAL WITH CHANGEOVER

Figure 4A



TRIPLEX VOTER

Figure 4B



DUPLICATE SELF-MONITORED

Figure 4C

Both the dual-dual and triplex arrangements should offer performance advantages in digital form over their analogue equivalents as the increased accuracy and low drift inherent in correctly applied digital computing should give very low "missed fault" and "nuisance disconnect" probability.

In the search for the simplext failure survival hardware configurations using digital on-line controllers there is also the third option of a duplicate-monitored system using only two self-monitored processors. Techniques of combined software and hardware self-monitoring are generally able to achieve a high degree of confidence in detecting processor faults, however to date there is little evidence as indicated in section 3, that this can be achieved economically if confidence levels much above 99% are required, and this is certainly necessary for applications like civil automatic landing.

Recent assessment of typical outer loop control tasks also indicates that with present component techniques a monitored processor is best implemented as a parallel machine whereas a non self-monitored processor is more cheaply made using serial techniques, provided there is adequate speed margin. It appears that a non-time-sharing incremental machine is uneconomic for such a task and of course may also lose in its inability to easily accommodate changes in control parameters.

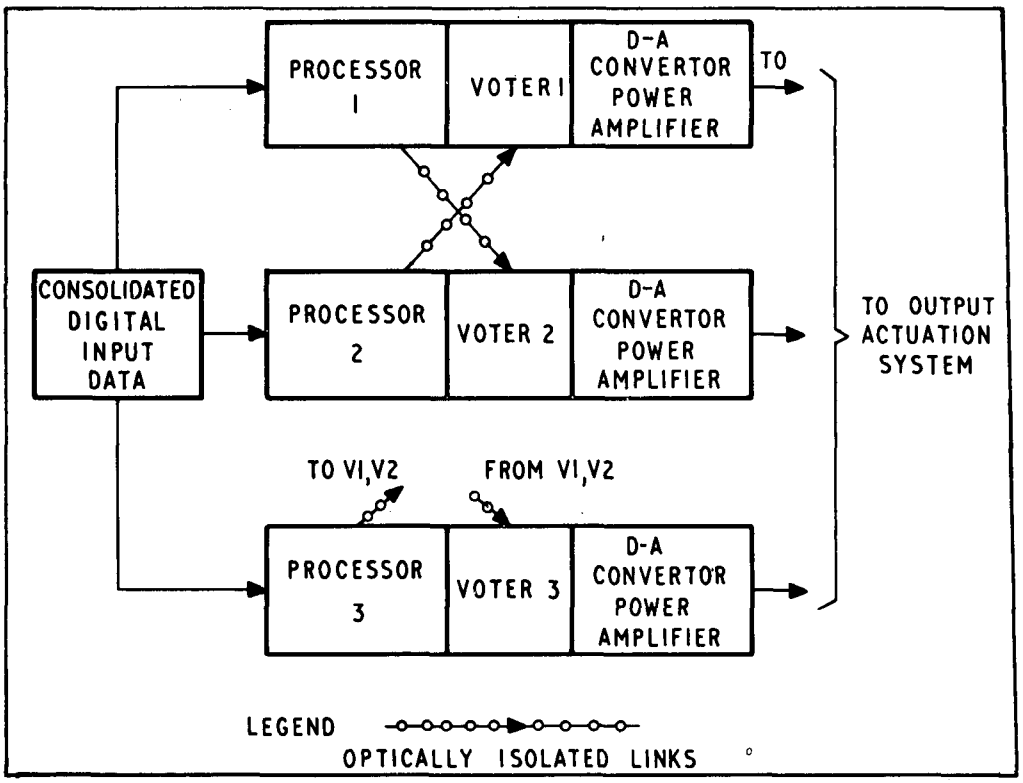
From these considerations it therefore appears that simple repetitive redundancy may offer the most attractive solution for the first generation of all-digital autopilots and the simplest of these, for a single failure survival capability is the combination of three unmonitored serial processors in a voting configuration. Despite the improved accuracy and drift some cross-voting of data will be required, but this potentially dangerous process can now be conducted with very high integrity by optical means.

A block diagram of such a system is shown in Figure 5.

To achieve valid channel comparison at the voter it is necessary to ensure that identical information is fed to each processor (the data must not be unequally stale) and that the processors operate in time synchronisation. This will involve optical crossfeed also of timing signals probably at program interrupt level (not illustrated) and allows voting by either software techniques of comparison, or hardware cross-examination of the serial outputs of the processors. It also allows the exchange between computers of scratch-pad data, if this becomes corrupted in any one processor the correct time history is lost.

The box labelled "consolidated digital input data" is deliberately not detailed to avoid undue

complexity. Its method of operation will depend upon the number and type of sensor inputs to be consolidated.



TRIPLEX VOTING CONFIGURATION
Figure 5

In considering generally the problem of "failure-survival" it is also worth point out that a consequence of employing a time shared digital processor in place of a multi-loop analogue control system is that a significant proportion of component failures cause a complete loss of function rather than a degradation in performance or loss of only one mode. This "graceful-degradation" characteristic of many analogue autopilots must not be lightly discarded, although it obviously decreases in importance as reliability increases.

To conclude then the consideration of the application of digital computing to outer loop aspects of automatic flight control systems, it appears that this is now bordering upon economic feasibility for both simple and complex systems. It is likely to offer significant advantages in ease of modification both following flight testing and in service life, and it is expected that increased accuracy and performance will be achieved by redundant systems. A limiting factor for a few years may be overall-reliability as the time-shared processor lacks the "graceful-degradation" characteristics of its analogue counterpart.

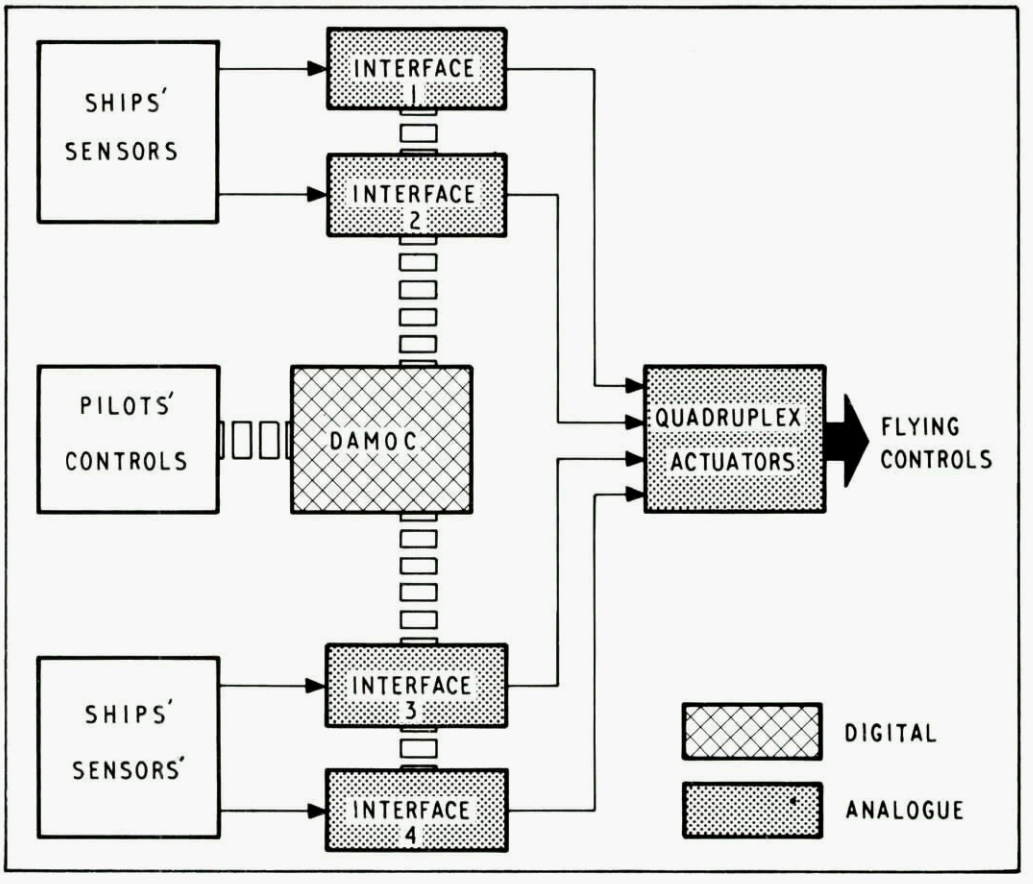
6. EXECUTIVE AND INTERLOCK FUNCTION

The concept of separating executive mode-switching and interlock functions from the hardware of control functions arose from an assessment of the solid state switching involved in the design of a very complex multi-mode civil AFCS, when it was found that the analogue computer boxes contained enough logic elements to make a small digital computer. It was therefore apparent that a much more flexible set of hardware could be obtained by actually using a small digital computer for this function.

Specifically the command and interlock functions using logic which are carried out in a flight control system fall broadly into the following categories:

- Mode compatibility logic (interlocks & acceptance)
- Preengagement tests (eg prevention of engagement if any internal failure or aircraft unacceptable error condition exists)
- Self-test selection and sequencing (BITE)
- Datum adjustment checks.
- Inter-mode sequencing.
- Warning indications.
- Gain switching over the flight envelope.
- Inhibit actions following malfunctions.
- Disconnects, changeovers and voter actions.
- Limit stop and limit rate assessments.
- Other limits on pilots controls and instruments.
- Instinctive cut out and power supply logic.

As indicated above, the complexity of these functions, especially in a complex system with considerable redundancy, such as civil automatic landing and VTOL transport controls, is sufficient to warrent the special use of a small digital processor and there arose the beginning of the hybrid AFCS concept which has now manifested itself in a practical new civil autopilot system. The system, known as E80 is shown in block diagram form in Figure 6. It achieves the desirable characteristics of modification flexibility most sought after and outlined in section 5. Briefly the system makes use of a small digital computer to carry out executive logic functions and transmit commands to the analogue control computers, along a serial digital transmission link. Further details are shown in Figures 7 and 8.



E80 AFCS - PRINCIPLE OF OPERATION

Figure 6

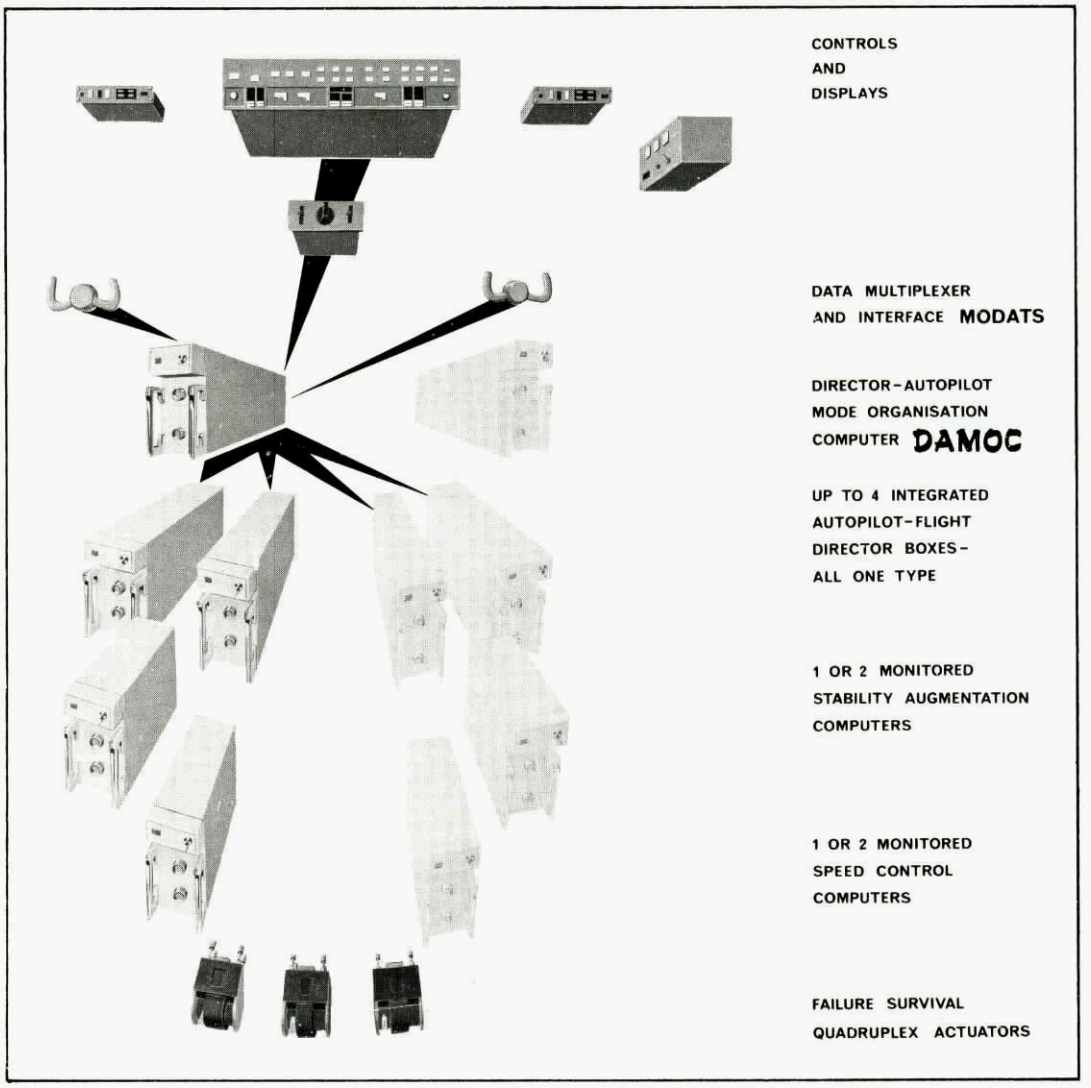
The ability to alter parameters easily and reversibly by changing a computer program using a variable store offers particular benefit while a system is undergoing flight testing. When flight testing is completed the variable store in the E80 system is replaced by a permanent read-only memory module incorporating the desired characteristics determined during the flight test program. (Figure 9)

Experience with military systems, even the more complex combat/strike terrain following type shows that such an approach may not be cost-effective. The simple systems can in fact contain all of the logic required in a simple controller and a few card modules. Nevertheless the combination of system logic requirements and the outer loop functions may jointly justify an overall digital solution as indicated in Section 5, but much will depend upon the complexity of interfaces required with the aircraft motion sensors, guidance devices and actuators.

To conclude then, the executive selection and interlock function is, by its logical nature, suitable for digital data processing techniques, but only in the complex civil systems will it occupy any substantial computer capacity. In this application the use of standard rather than special purpose logic designs gives more assurance that interlocks and mode selection functions will not be prey to the "sneak-path" difficulties of special purpose analogue equipment. The biggest advantage achieved is the much sought-after modification flexibility.

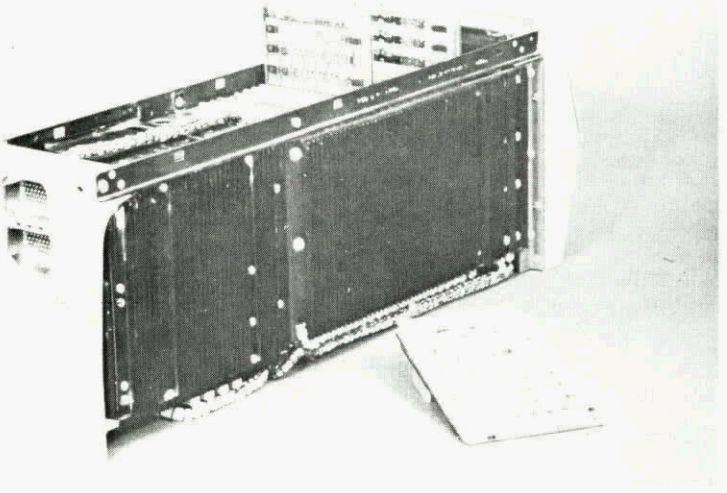
7. SOFTWARE DESIGN

At the present juncture software design must proceed under the constraint of the cost of memory capacity; the cost of present day minimal processors tends to be dominated by store costs. This provides the system designer with a number of problems. The main one is that there are all too few on-line programmers who can produce a really efficient software package and even fewer who can generate



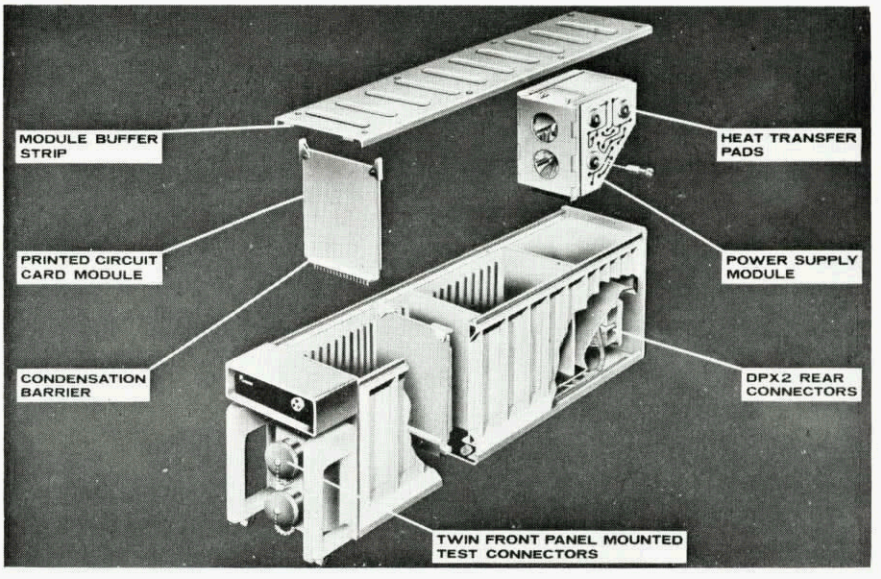
THE UNITS OF THE E80 SYSTEM

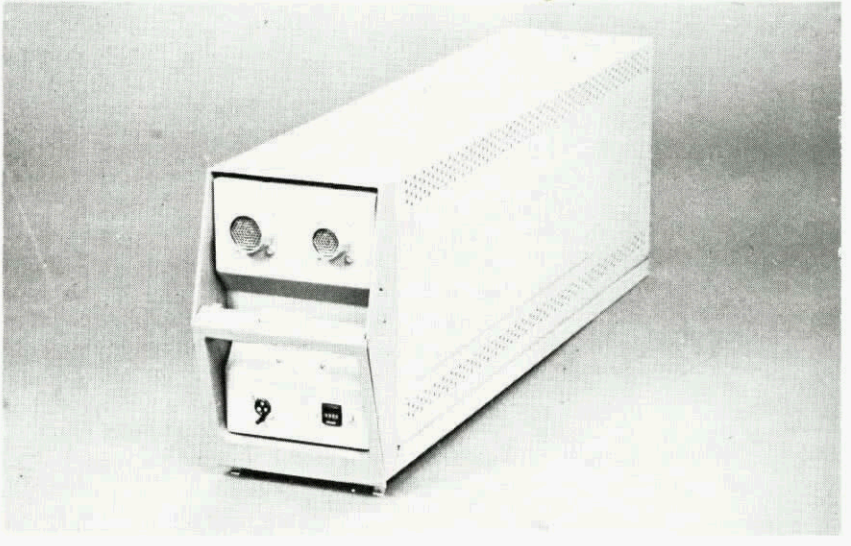
Figure 7



Director/
Autopilot
Mode
Organization
Computer
DAMOC
(Covers removed)

Analogue
Interface Unit
(illustrating thermal
design)





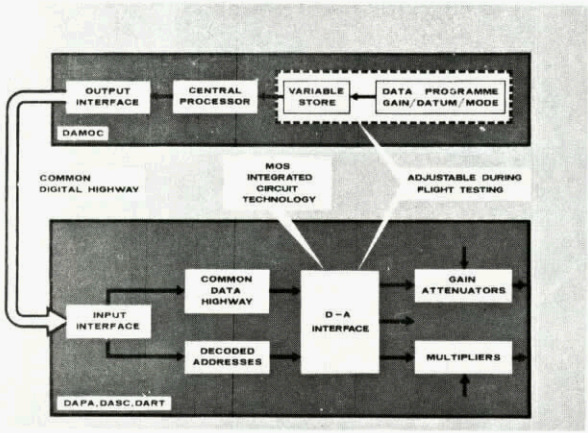
DAMOC

covers replaced

E 80 PACKAGES

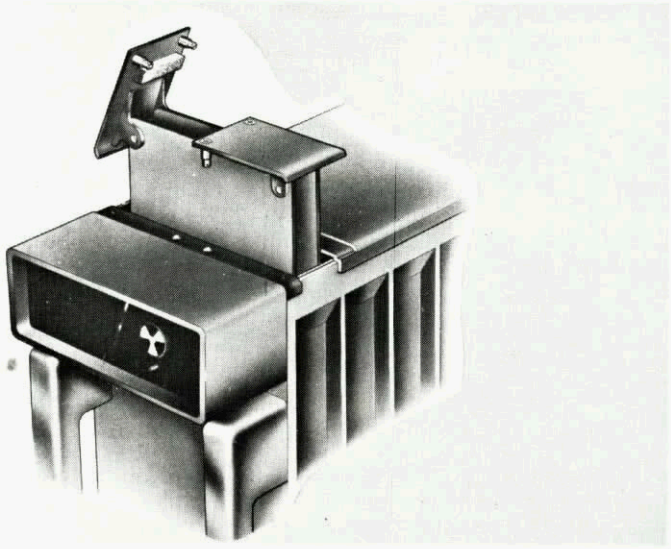
Figure 8

one which is both efficient in cycle time and memory utilisation and which can be readily understood and modified safely by less skilled people. The last requirement is most important if we are to obtain the benefits of flexible modifications, easy fault location, and fast certification which we so badly need.



E80 AFCS flexibility

E80 AFCS DAMOC removable program module - the E80 system adapter.



E80 FLIGHT TEST & MODIFICATION CONCEPT
Figure 9

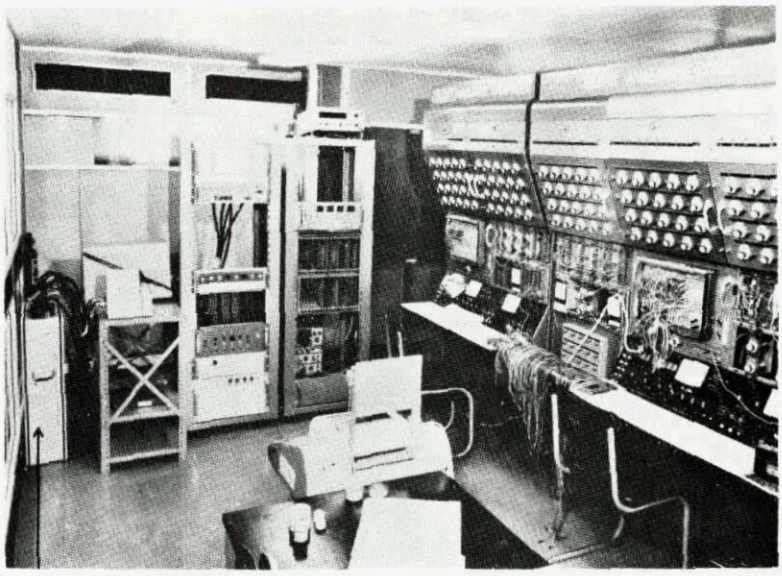
However by making use of established program compilation procedures, by taking advantage of the falling costs of semiconductor storage, and by using the technique of allocating different physical areas of store to different functions, this problem may readily be overcome. It should also be remembered that software must be designed for maintainability, just as much as hardware. Airline schedules or military mission success rates will suffer equally from shortcomings in either aspect.

As with analogue systems a great deal can be learned from comprehensive simulation, and in fact the speed of development of new digital circuit elements demands extensive and continuous systems investigations to keep the reasonable technological targets for automatic flight control systems within sight. The Elliott laboratory devoted to this is shown in Figure 10.

8. CONCLUSIONS

Digital computing is now a usable, cost effective technique for certain aspects of automatic flight control systems, particularly those which require complex but relatively low bandwidth computation. In particular the near future may see complex redundant systems such as civil automatic landing advantageously employ digital techniques. However reliability and integrity aspects need further developments, especially those related to the data transmission.

The higher bandwidth "inner-loop" auto-stabilisation type controls as yet cannot employ digital computing to advantage, but the speed of development is such that this may come about within the next 5 years. Certainly such a possibility will be enhanced by the advent of digital output sensors and actuators which could respond to digitally coded commands.



THE ELLIOTT DIGITAL AUTOPILOT DEVELOPMENT LABORATORY
Figure 10

Selective Digital Data Processing can certainly be employed now in automatic flight control systems and one example is the Elliott E80 series which incorporates a digital computer to organise autopilot modes and schedule gains.

The pressure towards system integration to produce more cost-effective avionic system complexes should be treated with great caution. This brings dangers in that the attraction of sharing tasks within a single system complex may encourage designers to overlook the widely different requirements for system integrity. Few ideas have led to more wasted time and in some cases failed programmes, that the plausible one of "let us take advantage of the GP computer's power and spare capacity to do all these things as well". The "systems separation" philosophy in airborne installations is as old as flying machines themselves. Digital computing is another technique now available to use or not to use, according to system, not computer, requirements.

With regard to automatic flight control systems we can sum up the probable trend in the next generations as progressively greater use of digital data processing.

Acknowledgement.

Acknowledgement is made to the many members of the engineering staff of Elliott Flight Automation Ltd., on whose very detailed work the general comments in this paper are based.

THE USE OF A DIGITAL COMPUTER FOR THE ELDO INERTIAL GUIDANCE SYSTEM

by

M. J. W. Gage

Space Department, Royal Aircraft Establishment,
Farnborough, Hants., England

SUMMARY

The projected ELDO Inertial Guidance System employs a small digital computer.

Those systems of a typical satellite launch vehicle that are suitable for mechanisation on such an on-board digital computer are listed, and details given for the case of the ELDO vehicle. The reasons for the choice of the computer and its role in the system are explained together with some details of the formulation and programming of the guidance law subsystem.

1 INTRODUCTION

The task of placing a satellite in an Earth orbit is at present accomplished by the use of a rocket launch vehicle (a launcher). In order to achieve the right orbit with as large a payload as possible, the launcher must follow the trajectory that minimises the fuel consumption. If the launcher was not subject to any unpredictable forces and if its performance was known exactly, it could be guided along this optimum trajectory by the use of a stored precomputed steering program. In practice such perturbations do occur, and so in order to still achieve an accurate injection into the required orbit, an active guidance system has to be employed. A requirement for this guidance system to be self-contained on-board the launcher, then leads to the need for an on-board computer. Once such a computer is available it can also be employed on other tasks such as flight sequencing, vehicle control, and data handling.

This paper is divided into three sections. Firstly a brief description is given of the systems involved in the guidance of a launch vehicle, and those systems suitable for mechanisation on an on-board digital computer are listed. Details are given for the case of the ELDO vehicle and its inertial guidance system (IGS). Secondly the performance of the computer chosen for the ELDO vehicle (Elliott MCS920M) is discussed, with an explanation of the reasons for its choice and of its IGS role. Lastly an outline is given of the derivation of an explicit guidance law, suitable for evaluation on the 920M, and its application to the ELDO vehicle. The paper explains the steps taken to provide a versatile system within the confines of a small on-board computer.

2 THE GUIDANCE OF A SATELLITE LAUNCHER

A launcher is basically a tool for accelerating a satellite from an initial position and velocity on the Earth's surface to a position and velocity corresponding to some required Earth orbit. The necessary thrust is generated as momentum reaction from a controlled chemical explosion, a rocket motor. In order to waste as little energy as possible accelerating unwanted structure, most vehicles are made up of a number of stages (3 in the case of the ELDO vehicle). Each stage is a separate rocket (fuel tanks, engines, etc.) that is discarded when burnt out.

The task of guiding such a satellite launch vehicle into a specified Earth orbit can be separated into three main problem areas, state determination, guidance philosophy, and steering execution. State determination is the navigation problem of computing the current state variables (position and velocity) of the launcher. Guidance philosophy is the problem of how to utilise this navigation information, in conjunction with the state variables corresponding to the required orbit and the expected vehicle performance (in terms of thrust levels, etc.), to generate a program of steering demands that, if followed, would guide the vehicle into this orbit. Once the necessary guidance demands have been derived they are applied to the engines of the launcher through a control system; the steering execution problem.

State determination

Two main alternative methods are at present employed for determining the information needed in order to calculate the current state variables of the vehicle. There is either ground-based radar/optical tracking or on-board inertial sensing. A combination of these techniques, plus others, could be used; however such schemes are usually more applicable to long term space missions rather than the brief launch phase. The former method plus a radio link with the vehicle has been used for all ELDO firings to date (1970). Future firings (from Kourou, French Guiana, rather than Woomera, Australia) will adopt the latter on-board inertial technique.

For a typical IGS, such as that for the ELDO vehicle (1), the state determination subsystems may be listed as: an inertial platform (space fixed) to provide an attitude reference for the vehicle, inertial accelerometers to measure the impressed acceleration (that due to forces other than gravity), and navigation equations to transform this information into some appropriate set of state variables. The evaluation of these navigation equations forms one of the main tasks of the on-board computer.

Guidance philosophy

The guidance philosophy subsystems may be listed as: a guidance law, the implementation of any constraints, the sequencing of various events, and the output of required event signals. There are two main classes of guidance laws; firstly implicit guidance, in which the steering commands are generated by reference to the departures of the current state variables of the vehicle from some pre-computed nominal trajectory; secondly explicit guidance, in which the steering commands are directly derived from the relation of the current state variables to those required for the orbital injection.

The guidance law (2) that forms part of the ELDO IGS is an explicit scheme based on the principle of in-flight optimisation. It was formulated to meet the requirements of high injection accuracy, fuel optimality, minimum pre-computed data, and high mission flexibility. At the same time the law also had to be suitable for real time evaluation on the 920M computer.

Steering execution

For a launch vehicle, steering is usually effected in terms of demanded attitude. The major part of the thrust is directed along the axis of the vehicle through its centre of mass, and is hence aligned with the so called attitude of the vehicle. A small amount of thrust is available for exercising a torque to change this attitude (usually either by deflecting the main thrust or by the employment of small extra thrusters). The vehicle is then steered by controlling the engines to null any existing errors between the current vehicle attitude and the demanded attitude. The subsystems involved are control equations, autopilots, motor actuators, and rocket motors.

The guidance configuration for the ELDO vehicle computer

For the ELDO vehicle under inertial guidance the systems incorporated into the 920M computer are, the navigation subsystem, the guidance law subsystem, some event sequencing, the output of motor cut-off signals, the implementation of attitude rate constraints at staging, and data telemetry to the ground for post flight analysis. This choice was of course influenced by the decision to retain all the systems not directly affected by the change from radio to inertial guidance, for example the autopilots.

Flight sequence

The basic flight sequence of the ELDO vehicle is as follows. Stage 1 (carrying stage 2 and stage 3) is controlled to follow a pre-computed attitude program, designed mainly to allow it to pass safely through the atmosphere. After stage 1 has burnt out and been discarded, stage 2 (carrying stage 3) is controlled to follow the inertial guidance demands until it in turn burns out and is discarded. Finally stage 3 is inertially guided up to orbital injection into a low parking orbit. This is in fact accomplished in two phases. Firstly, for the 3A phase, stage 3 is guided under main thrust (about 23000 N) until a certain set of state variables is achieved, when the guidance law issues a command to cut the main motor. Secondly, for the 3B phase, stage 3 is then guided under vernier thrust (about 800 N) up to the injection state variable, when the guidance law issues a further command to cut the vernier motors. The final task of the computer is to navigate and provide attitude control during a period of free fall motion round the parking orbit, which is terminated by the separation of the PAS spacecraft from stage 3. The PAS spacecraft (perigee apogee system) then transfers the satellite to a synchronous orbit using a two impulse Hohmann transfer from the initial low orbit.

3 THE COMPUTER SYSTEM

The choice of an on-board computer for the ELDO IGS was dictated by a number of requirements. First and most obviously it had to be of a suitable mass and volume. Secondly it had to be capable of operation in the launch environment and compatible (in terms of power levels, etc.) with launch vehicle facilities. Thirdly it had to be capable of being programmed to provide navigation data and guidance demands for accurate guidance of the vehicle into a near-Earth orbit. This last requirement was established by the U.K. firm Elliott Brothers (London) Ltd. (now Elliott Space and Guided Weapon Division) for their 920M computer, by means of a study (3) done under a contract from the then U.K. Ministry of Aviation. Elliott was appointed prime contractor for the development of the IGS on 1 January 1967. The Royal Aircraft Establishment, Farnborough, was given the task of making a study of alternative guidance laws and then supplying their chosen law to Elliott for programming on the 920M.

The Elliott MCS920M is a microminiature digital computer (4) that is capable of operation in the launch vehicle environment. The volume of the central processor is 0.012 m^3 with a mass of 15.2 kg and an average power consumption of 45 W. The 920M is constructed in three sections, a control unit, an arithmetic unit, and a ferrite core store. In each section the logic modules are clamped together to form a thermally conductive block. This enables the heat generated during operation to be conducted to the side panels of the aluminium casting, which act as heat exchangers. The computer is designed to operate under an ambient temperature range of $-10^{\circ}C$ to $+55^{\circ}C$, all store circuits being 'worst case designed' to work over the range $-20^{\circ}C$ to $+100^{\circ}C$. Other operational limits are, vibration ± 75 m/s^2 for 0-2000 Hz, shock 250 m/s^2 for 10 ms, and linear acceleration 150 m/s^2.

The 920M is available with either a 5 μs or a 2 μs store cycle time, the 5 μs version giving gross operation times of 19 μs for addition, 21 μs for subtraction, 38 μs for multiplication, and 39 μs for division. Numerical data is represented as 18 bit fixed point binary fractions, employing a 2's complement representation of negative numbers. The core store has a capacity of 8192 of the 18 bit words. The 920M has an explicit order code of 16 instructions which covers all basic arithmetic, logic, and transfer operations. Double length working with a resolution of 35 bits is available for certain operations. Provision is made for 4 program priority levels which are arranged to operate on an interrupt basis. Upon receipt of an interrupt signal for a higher priority level program, the program currently being obeyed is temporarily suspended and the higher level program entered. Upon completion of this higher level program, the computer then continues the lower level program from the point of interruption.

Application to the ELDO IGS

A schematic diagram of the IGS is shown in Fig.1. A Ferranti FE 610 inertial platform provides the attitude reference for the vehicle. Synchros on the gimbals of the platform measure the attitude angles of the vehicle and this information is fed into the computer via a synchro to digital converter, in the form of 12-bit digital quantities. Three single axis force feedback accelerometers, orthogonally mounted on the platform cluster, measure the impressed acceleration of the vehicle. This information is integrated by analogue means in an accelerometer encoder (three channels) and fed into the 920M in the form of pulses, each pulse representing 1 m/s.

The computer accumulates this integrated impressed acceleration (which is in effect the true impressed velocity rounded down to the nearest m/s) and implements the navigation program to determine current state variables, and the guidance program to generate a demanded attitude. The difference between this demanded attitude, expressed in gimbal angles, and the current vehicle attitude is output, via a decoder, as an analogue error signal to the autopilots. Vehicle rates information (from rate gyros) is also input to the autopilots for control purposes. The computer outputs signals to cut-off the main and vernier engines of the third stage and to initiate the PAS separation sequencer. The various tasks of the computer are executed on a priority basis by assignment to one of the four priority levels. The use of the levels is synchronised by triggering the interrupt facilities using timing pulses derived from the telemetry clock.

Use of priority levels

The fourth and lowest level is allocated to a self check program. This operates asynchronously whenever none of the higher levels is operating and checks the correct functional performance of the computer and the integrity of the overall program.

The third priority level is entered every 200 ms. The accumulated impressed velocity is extrapolated (using the three times from the receipt of the last increment in each channel, the current nominal acceleration magnitude and the current vehicle attitude) to obtain the current impressed velocity more accurately than to the nearest m/s. This is then resolved from the platform axes into the set of inertial (space fixed) axes used for navigation. The navigation program then includes the effect of gravity (from a model of the gravitational field of the Earth) to form the true current velocity of the vehicle. The equation used is

$$\dot{\underline{X}} = \underline{V} + \int \underline{g}(\underline{X})\, dt \qquad (1)$$

where $\underline{X}$ is the position vector, $\underline{V}$ the impressed velocity vector, and $\underline{g}(\underline{X})$ a function representing the acceleration of the vehicle due to gravity. This function is derived from a gravitational potential β chosen to provide results well within the ELDO specification on navigation accuracy.

$$\beta = \frac{\mu}{r} - J_2 \frac{\mu}{r^3} (3 \sin^2 \lambda - 1) \frac{D^2}{2} \qquad (2)$$

where μ is the gravitational constant of the Earth, J_2 the largest zonal harmonic coefficient, D the equatorial radius of the Earth, r the distance of the launcher from the Earth's centre, and λ the geocentric latitude of the launcher.

For the guidance of stage 2 and stage 3 the current vehicle state variables are resolved from the navigation axes into the guidance axes and fed into the guidance sub-program. Here these state variables, in conjunction with those corresponding to the required orbit, are used to generate an optimum demanded attitude. This is then resolved back into the navigation axes and in turn expressed as demanded gimbal angles. The variable matrix for the transformation from platform to vehicle body axes is also computed.

The current impressed acceleration level is monitored in order to detect the stage 1 and stage 2 burn-outs and initiate the program sequencing to ensure that, corresponding to the current stage, the correct part of the program is being exercised. The times for the stage 3A and 3B cut-offs are computed as part of the guidance law. Level three is also used to assemble the telemetry as 48 words, each of 8 bits. The choice of a 5 Hz navigation and guidance iteration cycle frequency (cycle time 200 ms) was made as the result of a detailed accuracy analysis that concluded that the cycle time (and therefore the navigation step length) had negligible affect on the computation accuracy provided it was not greater than 400 ms.

The second priority level is entered every 12.5 ms. The integrated acceleration from the encoders is read and accumulated. The three times of receipt of the last increment in each channel are stored for use in the level 3 extrapolation. The program also reads the current vehicle attitude, as measured by the synchros, and forms the difference between this and the current demanded attitude, obtained from the guidance in level 3 during the second and third stage flight or from a pre-computed program in level 2 during the first stage flight. This difference is resolved into pitch, yaw and roll errors in the vehicle body axes and output to the autopilots via a digital to analogue decoder.

The high iteration rate (80 Hz) for this level was chosen to avoid any interaction with any of the autopilot loops; for example the critical frequencies of the first stage autopilot range up to 30 Hz. The effects of the introduction of the digital element into the existing analogue autopilot loops (5) were evaluated by detailed simulation. The important factors were the quantisation of the input attitude and output error signals and the 5 Hz and 80 Hz iteration rates. A problem was encountered with the attitude bit size leading to limit cycling which, for the first stage, could have interacted with rigid body and fuel sloshing frequencies. This problem was successfully overcome by the insertion of 112.5 Hz sinusoidal dither (as an analogue signal) before the encoders and the inclusion of output modulation (in the computer) before the decoder.

The first and highest level is used to output the 48 words, assembled in level 3, to the telemetry, and is entered on demand when the telemetry requires data. By this means all the inputs and outputs to the computer, plus other basic parameters, are made available for post flight analysis. The details of the times spent in each program level during one 200 ms cycle are shown in Fig.2.

Programming details

The 920M is basically a fixed point machine and although higher language compilers are available, for its real time on-line task in the IGS the flight program (6) was written directly in machine code. This enabled each sequence of operations to be programmed in an optimal manner. The storage capacity was found to be adequate for the IGS task (the flight program other than the self check routines occupying 50% of the available capacity), hence the main requirement was for the reduction of the operating time. In fact (as shown in Fig.2) the flight program other than the self check routines occupies 51% of the 200 ms cycle.

For the IGS application the range of parameter variation, although large, is well determined and so no major difficulties were encountered due to the use of the fixed point mode. A detailed error analysis showed that the accuracy of the computation was fundamentally related to the resolution of the input data, and that to minimise errors full use must be made of this data. As the resolution (i.e. bit size) available in floating point arithmetic is dependent on the magnitude of each parameter it becomes difficult to control the overall accuracy. The preferred solution of fixed point scaling enables total control over numerical representation and the full utilisation of input data.

The selected scaling gives basic bit sizes of 64 m for position, 5/64 m/s for velocity, and 6.25 ms for time. Certain specific areas of the program do however require greater accuracy than can be provided by one 18-bit word. For example the integrations of the navigation equations are performed using a trapezoidal routine that employs double length working.

4 THE GUIDANCE LAW

In principle the best and most flexible solution to the guidance problem is to carry out periodically in flight a complete optimisation of the future flight using an accurate mathematical model of the vehicle's motion. This then forms an explicit guidance scheme, the required steering commands being generated purely from a knowledge of the vehicle's current and required state variables plus a knowledge of the vehicle's likely performance and the Earth's gravitational field. However this optimisation does involve a considerable amount of computation as a result of the non-linear nature of the equations involved. Hence for a real-time application on a small computer it is necessary to simplify this basic concept in such a way that the computational effort necessary is compatible with the computing performance available.

In order to overcome the computational complexities connected with optimisation in the gravitational field of the Earth, use has been made of the fact that the optimum steering angles in a constant unidirectional force field can be expressed as a comparatively simple analytical formula. A first order of simplification solution (7) is to substitute this formula back into the accurate mathematical model of the vehicle's translational motion and to solve for the required current guidance demand. This solution still involves extensive computation, in particular numerical integration, and so in order to derive a closed form analytical solution further simplifications are made. This second order of simplification solution (8) is to substitute the optimum steering formula into a simplified mathematical model of the vehicle's translational motion, chosen so as to lead eventually to such a closed form solution.

As a typical constant thrust launcher vehicle requires only two control parameters (corresponding to pitch and yaw steering), the trajectory generated using the simple model (referred to as a flat Earth; hence the name FE guidance) can be made similar in its essential features to the accurate optimum trajectory by the introduction of two correction quantities, $\bar{g}$ and $\bar{e}$, into the flat Earth equations of motion. $\bar{g}$ and $\bar{e}$ are chosen in such a way that they not only correct for the transformation from the accurate model to the flat Earth model but also allow for further simplifications to the flat Earth equations.

The guidance equations are most simply formulated in spherical polar coordinates, as the motion takes place in what is primarily a central force field. The cut-off boundary conditions assume their simplest form relative to these axes, and thus any guidance law involving these conditions will have its simplest formulation when employing these coordinates. The axes are shown in Fig.3. Pu is parallel to the plane of the required orbit π, Pw is Earth centre pointing, and Pv completes the right-handed orthogonal triad through the vehicle. OXYZ are the navigation axes, the Z coordinate measuring the distance of the vehicle from the plane π. χ and ψ are the steering angles, χ being the angle between the demanded attitude and the Puv plane (local horizontal) and ψ the angle between the projection of this attitude on the Puv plane and the Pu axis.

For a launch vehicle with constant fuel mass flow rate the cost function to be minimised can be taken as the flight time. If $\underline{S}(t)$ is the vehicle's state vector at time t, the problem is to find the $\chi(t)$ and $\psi(t)$ that minimise $(T - t_c)$ subject to the constraints

$$\underline{S}(t_c) = \text{vehicle's state vector at current time } t_c \tag{3}$$

$$\underline{S}(T) = \text{state vector required at cut-off time } T \tag{4}$$

$$\dot{\underline{S}} = \underline{F}(\underline{S}, t, I, \tau, \mu, J_2, \chi, \psi) \tag{5}$$

where this last equation forms an accurate description of the translational motion of the vehicle. I and τ are parameters describing the thrust performance of the vehicle, the impressed acceleration a being given by

$$a = I/(\tau - t) \tag{6}$$

where I is the exhaust speed and $\tau = (m_o/\dot{m}) + t_o$, m_o the total mass of the vehicle at time t_o and $\dot{m}$ the fuel mass flow rate.

The flat Earth solution is obtained by replacing Eq. (5) by

$$\dot{\underline{S}} = \underline{f}(\underline{S}, t, I, \tau, g, \chi, \psi) \tag{7}$$

where $\underline{f}$ is a simplified form of $\underline{F}$ and g a constant accounting for gravity. The optimum $\chi(t)$ and $\psi(t)$ for this problem can be shown to have the form

$$\sec\psi \tan\chi = At + B , \quad \tan\psi = Ct + D \tag{8}$$

where A, B, C, D (all constants) and T are determined by the 5 constraints represented by Eq. (4). These are 5 in number as they correspond to the case of injection into a circular orbit (u, v, w velocity, r, Z position, all constrained).

Two constants $\bar{g}$ and $\bar{e}$ are now introduced into Eq. (7) so that in Eq. (8) $\chi(t)$ will become dependent on $\bar{g}$ ($\bar{g}$ in fact merely replaces g) and $\psi(t)$ on $\bar{e}$. These quantities $\bar{g}$ and $\bar{e}$ can then be chosen to match $\chi(t_c)$ and $\psi(t_c)$ to the accurate current optimum steering angles. For each new integration interval from a new t_c, $\bar{g}$ and $\bar{e}$ will assume new values. Comparison of the equations repreqented by Eq. (5) and Eq. (7) suggested taking

$$g = \frac{\mu}{r^2} - \frac{u^2 + v^2}{r} \tag{9}$$

where u and v are current velocity components and r the current radial position. For all optimum launch trajectories studied, $\bar{g}$ in fact takes values close to g and $\bar{e}$ close to 1, where g, although taken as a constant for each integration, is a function of the state variables at t_c. These correction functions $\bar{g}$ and $\bar{e}$, that enable the FE guidance demands to be matched to the true optimum angles, form in principle the only data that needs precomputation. In fact for all optimum launch trajectories studied, the use of $\bar{g} = g$ and $\bar{e} = 1$ leads to only a few kg loss in mass into orbit.

To appreciate the basic structure of the FE guidance equations it is useful to consider their form in the neighbourhood of injection defined by the requirement that $|\bar{u}/I| \ll 1$. Under this condition the equations simplify to

$$\tan\chi(t_c) = \left[\frac{w - W + \bar{g}\,T'}{\bar{u}}\right] - 6a\left[\frac{R - r + 0.5\,(w + W)\,T'}{\bar{u}^2}\right] \tag{10}$$

$$\tan\psi(t_c) = \left[\frac{v - V}{\bar{u}}\right] - 6a\left[\frac{\bar{e}\,(Z_T - Z) + 0.5\,(v + V)\,T'}{\bar{u}^2}\right] \tag{11}$$

$$T' = T - t_c = -\bar{u}/a \tag{12}$$

where $\bar{u} = u - UR/r$, the u velocity-to-go; u, v, w, r, Z are the state variables of the vehicle at time t_c; U, V, W, R, Z_T are those required at cut-off; a is the nominal vehicle impressed acceleration; and T is the predicted cut-off time. It can be seen that Eq. (10) and Eq. (11) both consist of two components, the first controlling the velocity error at cut-off and the second the corresponding position error.

The FE guidance law aims to achieve the required cut-off conditions exactly (i.e. no tolerances on the injection state variables) and so will attempt to correct even very small injection errors. If vehicle performance perturbations are present they can result in large steering angles being commanded in an attempt at 'last minute' manoeuvres. The vehicle can therefore be forced by the guidance law to perform violent steering manoeuvres in order to remove only small errors at injection. In order to avoid these unnecessary and perhaps harmful commands it is advisable to relax the stringent injection requirements somewhat. As very small position (r and Z) errors are unimportant in practice, shortly before cut-off the FE equations are replaced by a guidance law formulated without the position constraints. This is accomplished by removing the second components of Eq. (10) and Eq. (11), leaving the so called injection guidance equations

$$\tan\chi(t_c) = \frac{w - W + \bar{g}\,T'}{\bar{u}} \tag{13}$$

$$\tan\psi(t_c) = \frac{v - V}{\bar{u}} \quad . \tag{14}$$

The equations have so far been formulated for 5 terminal boundary constraints corresponding to the case of injection into a circular orbit. This change to the injection guidance equations provides the opportunity for a slight modification to the equations to allow for the case of injection into an

ellipse. Here there is no need to impose a constraint on the cut-off height, provided the cut-off velocity in the orbital plane becomes functionally dependent on this height (i.e. on r). The u cut-off value is already functionally dependent on r in the form required (viz. UR/r), hence it only remains to replace the constant W by

$$W = -\sqrt{\frac{2\mu}{r^2}\frac{(R_A - r)(r - R_P)}{R_A + R_P}} \qquad (15)$$

where R_A and R_P are the apogee and perigee radii of the required ellipse.

Application to the ELDO vehicle

FE guidance was formulated to meet the requirements of high injection accuracy, fuel optimality, minimum pre-computed data, and high mission flexibility. It has been satisfactorily programmed by Elliott for the 920M and is suitable for real time operation within the 200 ms cycle. The flight program (9) is set up to guide the state variables of the vehicle, throughout stage 2 and stage 3A, to those corresponding to the nominal values at the end of the 3A phase. Having achieved this state the main motors are cut and the vehicle is then guided under the vernier thrust up to the true injection state variables. This procedure is adopted to avoid the possibility of the vehicle being left at the end of the main phase in a position from which the verniers are too weak to achieve the final injection.

The flight program contains the guidance programmed in such a way that all likely ELDO missions can be covered without any actual reprogramming, allowable changes being restricted to data alone. This data consists of parameters specifying the launch site, the required orbit, the gravitational environment, the mission sequencing plus the guidance parameters, and various other systems parameters.

5 CONCLUSIONS

This paper has described the use of a small on-board digital computer as part of the inertial guidance system of the ELDO launcher. Reasons have been given for the choice of the Elliott 920M computer and a description made of its performance and role in the ELDO inertial guidance system. In particular it is shown how a guidance law giving high orbital injection accuracy, fuel optimality, minimum pre-computed data, and high mission flexibility, has been successfully formulated and programmed as part of the computer's inertial guidance flight program.

REFERENCES

No.	Author	Title, etc.
1	A. Riley	The Inertial Guidance System for the Europa Satellite Launcher - 19th Congress of the IAF - New York 1968
2	M.J.W. Gage	The guidance of satellite launcher vehicles with particular reference to the ELDO vehicle - BIS Journal Vol.23, No.5, pp.75-92, 1970
3	J.P. Long	Investigation of the suitability of the Elliott MCS 920M digital computer for the navigation and guidance of the ELDO satellite launch vehicle - Elliott Space and Guided Weapons Division, 1966
4	A.H. Cairns	The Elliott MCS920M computer, with particular reference to its application to the inertial guidance of the CECLES-ELDO launcher - CNES International Conference on Aerospace Computers in Rockets and Spacecraft - Paris, 1968
5	A.H. Cairns	Autopilot compatibility of the digital inertial guidance system used in the ELDO 'Europa' launch vehicle - 2nd International Conference on Space Engineering - Venice, 1969
6	A. Riley	Checkout procedures for the computer program used in the ELDO inertial guidance system - AIAA Aerospace Computer Systems Conference - Los Angeles, 1969
7	M.J.W. Gage	An explicit guidance law for satellite launcher vehicles. R.A.E. Technical Memorandum Space 82, 1966
8	M.J.W. Gage	FE guidance and its application to the ELDO launcher vehicle. R.A.E. Technical Report 68201, 1968
9	M.J.W. Gage	The inertial guidance law of the ELDO launch vehicle. R.A.E. Technical Report 70032, 1970

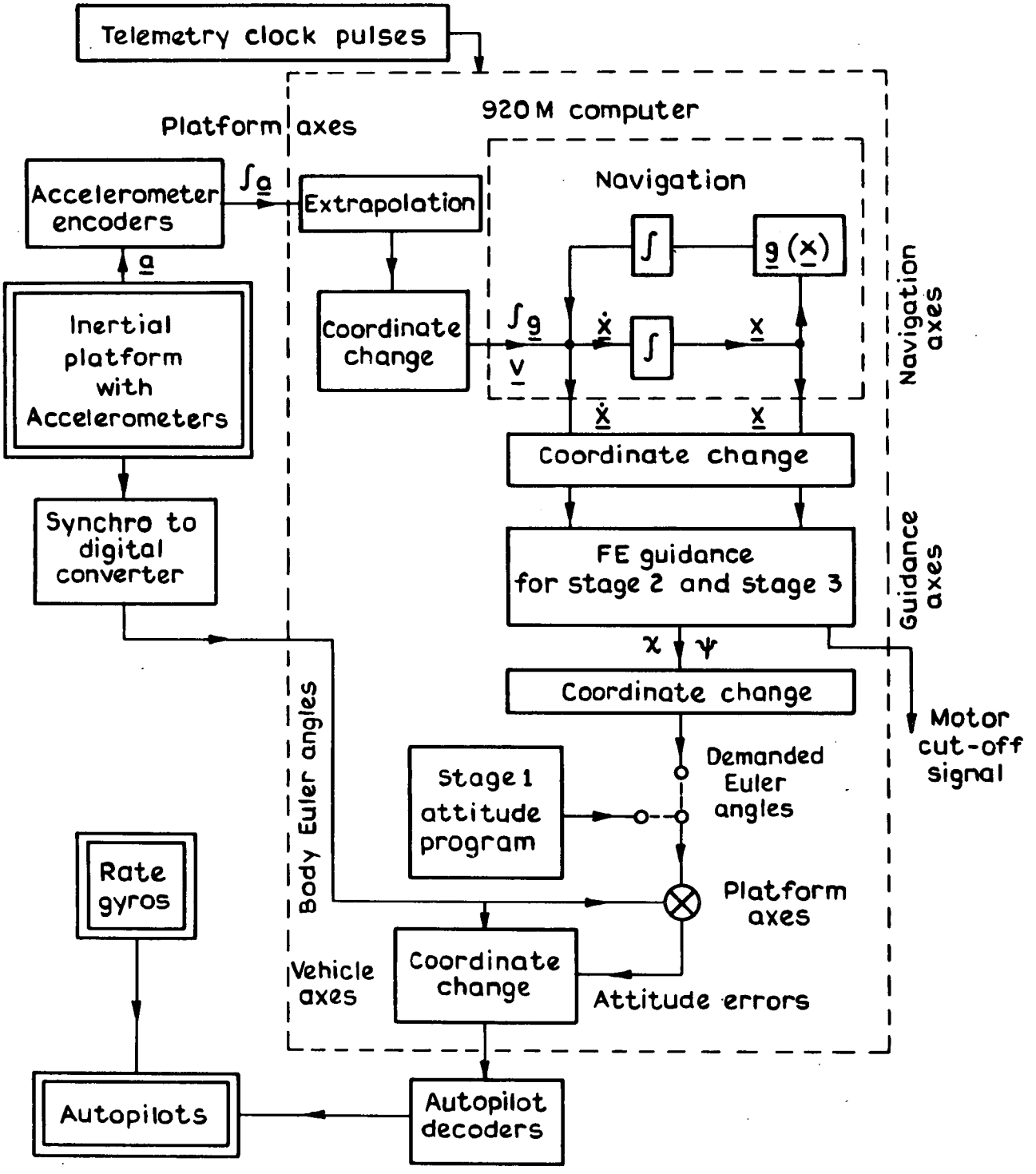


Fig.1 Eldo inertial guidance system

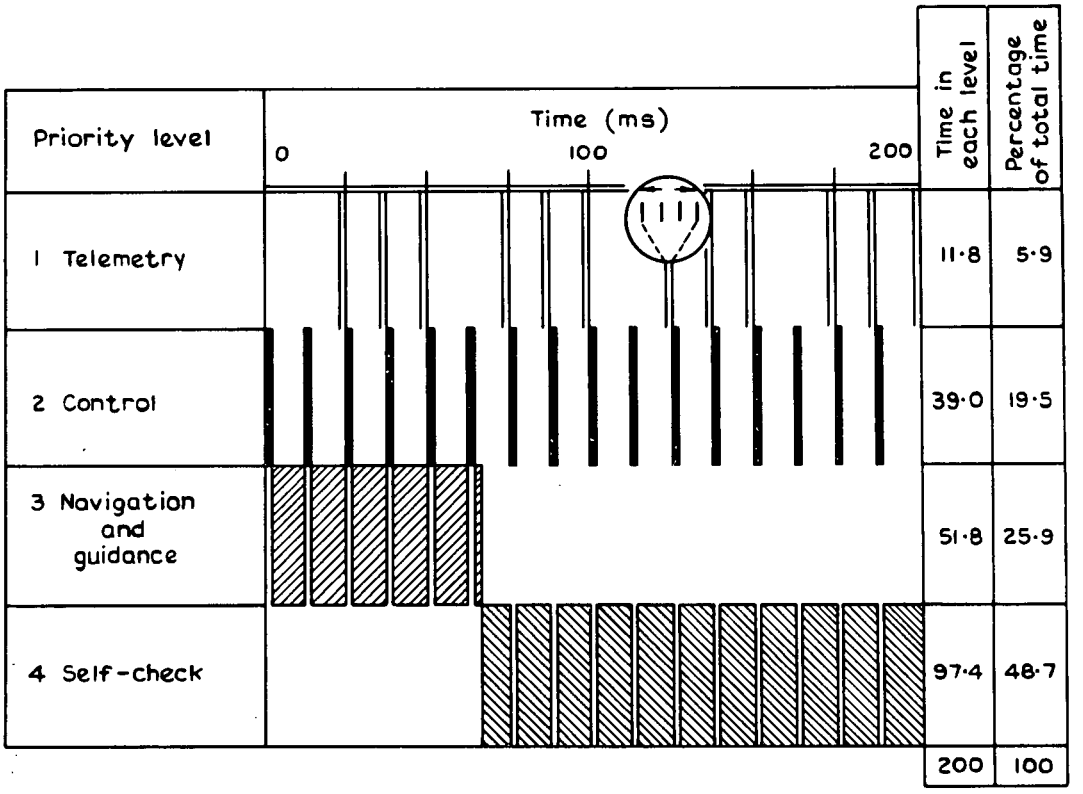
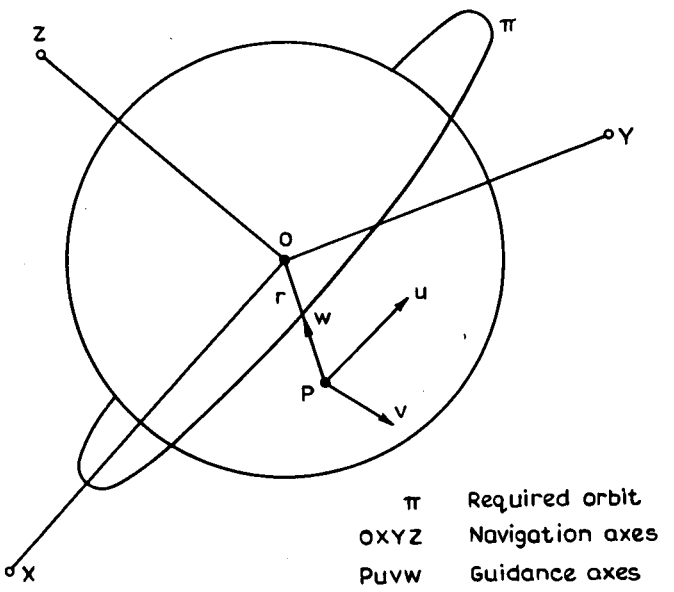


Fig.2 Program timing



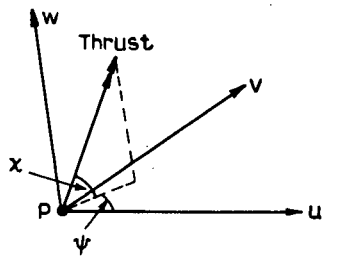


Fig.3 Navigation and guidance axes

THE STAR COMPUTER: A SELF-TESTING-AND-REPAIRING COMPUTER FOR SPACECRAFT GUIDANCE, CONTROL, AND AUTOMATIC MAINTENANCE*

Dr. Algirdas Avižienis†, Member of Technical Staff
Jet Propulsion Laboratory, California Institute of Technology
Pasadena, California, USA

*This paper presents the results of one phase of research carried out at the Jet Propulsion Laboratory, California Institute of Technology, under Contract No. NAS 7-100, sponsored by the National Aeronautics and Space Administration.

†Also, Associate Professor, Computer Science Department, School of Engineering and Applied Science, University of California, Los Angeles, California, USA.

SUMMARY

The success of long-term missions of unmanned spacecraft in the exploration of the solar system depends on the survival of the spacecraft during long intervals of interplanetary flight. Missions of up to 12 years duration are being considered for the "Grand Tour" multiple-planet flyby opportunity in the late 1970's. The long unattended life requirement and the extreme communication distances impose immense reliability requirements on the onboard digital computer which performs the guidance and control computations. The development of such a computer has been the objective of a research and advanced development project which was initiated in 1961 and is continuing with an increasing scope at the Astrionics Division of the Jet Propulsion Laboratory. An experimental aerospace computer with built-in automatic maintenance features is described in this paper. The computer is called the STAR (Self-Testing-And-Repairing) computer. The hardware, the software, and the estimation of reliability of the STAR computer are discussed. An extension of the STAR concepts to automatic maintenance of other systems of an unmanned interplanetary spacecraft is also considered.

1. INTRODUCTION

Successful planetary exploration missions have been completed by unmanned Mariner spacecraft, which were designed at the Jet Propulsion Laboratory. Mariner II completed a 109-day flyby mission to Venus on December 14, 1962. This mission was followed by the flyby of Mars and return of 21 historic photographs of its surface by Mariner IV in July 1965. Further successful missions followed: Mariner V to Venus in 1967, Mariners VI and VII to Mars in 1969. In all of these missions, a Central Computer and Sequencer onboard the spacecraft was used to provide on-board control functions, although complete ground backup was also available.

At the present time attention is being directed towards the unmanned exploration of the outer planets Jupiter, Saturn, Uranus, Neptune, and Pluto [1]. These planets will reach an unusually favorable alignment during the years 1976-80. This alignment, which will not be repeated for about 170 years, permits the use of the gravitational field of Jupiter to deflect and accelerate the spacecraft toward Saturn, and from there toward Uranus and Neptune, or toward Pluto. Much less energy is required to reach the outer planets under these conditions, and launch vehicles that would be limited to Jupiter missions can be employed for the entire "Grand Tour." Mission times of 8-12 years become available, as opposed to direct mission times of 18-40 years.

Even with the favorable alignment of the outer planets, the mission times exceed by on order of magnitude or more the successful nine-month Mariner missions to Mars. An autonomously operating on-board digital computer becomes an absolute necessity because of three factors. First, the length of the mission makes continual ground support economically unfeasible. Second, the extreme communication distances (for example, 4 hours one-way time for a message from Neptune) require self-contained response to failures and abnormal conditions. Third, full reprogrammability is needed to make best use of information gained as the mission progresses and to adapt to partial failures of the spacecraft system. The computer permits the final encounter decisions to be made a few days before arrival at a planet. A completely new set of programs can then be devised during the years-long cruise to the next planet.

The unattended survival requirement for an 8 to 12 year period calls for a computer which possesses a self-test and self-repair function. Only a computer of this type can serve as both the "pilot" and the "automatic repairman" for the entire spacecraft. Even as the first Mariner spacecraft was being readied in 1960-61, it became evident that its modest sequencer would have to be succeeded by a self-maintaining computer as more and more ambitious missions would follow in the future. A research effort intended to lead to the development of a Self-Testing-And-Repairing (STAR) spacecraft computer was initiated in the Flight Computers and Sequencers section at the Jet Propulsion Laboratory in early 1961 and has been continued with increasing levels of effort until the present.

The first four years of the STAR effort (1961-1964) were devoted to theoretical studies of protective redundancy and to preliminary designs to evaluate competitive schemes of fault-tolerance. The conclusions strongly favored a replacement system, and beginning with 1965 the work was concentrated on the development of a spacecraft guidance and control computer with provisions for the detection of transient and permanent (simple as well as catastrophic or multiple) faults and with spare unpowered functional units for repair of permanent damage [2]. The construction of an experimental laboratory model ("breadboard") of the STAR computer was initiated in 1967, and the first program was successfully carried out in March 1969 by a subset of the complete system. At the present time the STAR computer is in limited operation, and additional functional units are being constructed and added. The following sections of this paper present a general treatment of the problems of fault-tolerant computers, followed by a discussion of the hardware, software, and reliability assessment of the STAR computer. The paper is concluded with some considerations of the extension of automatic maintenance to the entire spacecraft.

2. FAULT-TOLERANT DIGITAL COMPUTERS

The idealized or "perfect" digital computer can serve as a reference point in the discussion of fault tolerance. Two-state logic circuits (storage elements and operator elements) are the elementary building blocks of a digital computer. The logic design of the computer specifies how these circuits are interconnected and which one of the two possible logic values ("true" and "false", most frequently designated by "1" and "0" respectively), will exist at the input and output points of every logic circuit at every defined instant of time. The "perfect" computer always functions according to the specifications of its logic design. It has been observed, however, that digital computers occasionally deviate from the design specifications. The logic value(s) at one or more points of the computer logic become opposite to the specified value(s). The deviation of a logic variable from its prescribed value is called a logic fault, or more concisely, a fault. Most faults will cause an error in the program being executed by the computer: either an instruction is not executed correctly, or an incorrect result is computed. Both types of errors may be caused at once by some faults. The exact nature of a fault depends on whether a permanent failure, or a temporary (transient) malfunction of one or more components (resistors, transistors, connections, etc.) or the computer is the cause.

Transient faults are temporary deviations of logic values from design specifications. Two main causes of transient faults are intermittent component malfunctions and external interference with the operation of the computer. Such interference is caused by irregularities of the power supply, stray electromagnetic radiation, severe environments, and similar events. Transient faults cause errors in computation without leaving a permanent record, and their occurrence will not be detected by the periodic checkouts of the computer, that are sufficient to detect the presence of permanent faults. Permanent faults are caused by permanent component failures. As a result the logic value at a certain point remains constant ("stuck on zero", or "stuck on one") regardless of design specification. Beside such determinate faults, occasionally an indeterminate (or "stuck on X") permanent fault may occur, in which the logic value varies between "1" and "0" but not according to the design. Such faults can be caused by drift of component values and similar phenomena. Logic faults also differ as to the extent to which they affect the computer. Independent, or local, faults affect only one logic circuit in the computer. They are caused usually by random failures of the components. Related, or catastrophic, faults simultaneously affect two or more logic circuits. They are very likely to occur in the case of physical damage or external interference to

the computer. The recent advances in the large-scale integration of electronic circuits lead to very close placement of logic circuits and make related faults more probable than is the case when discrete components are used.

The preceding discussion of logic faults now permits the definition of a fault-tolerant computer. It is a computer that can carry out error-free programs in the presence of logic faults [2]. The classes of faults that are tolerated and the parts of the computer in which they may occur must be identified in every claim of fault tolerance. Fault tolerance in digital computers is attained by means of protective redundancy. In this report, protective redundancy is defined as all additional programs (software redundancy), repetition of operations (time redundancy), and additional circuits (hardware redundancy), that are not needed in the "perfect" computer, but serve to provide fault tolerance in the physical implementation of the same logic design. Software redundancy includes emergency action programs that are used when a fault has been detected. Also included are diagnostic programs that are executed (periodically or upon request) to test all logic circuits of the computer for the presence of permanent faults. Time redundancy includes the repetition of a program or a segment of a program after a fault has been detected or suspected. Repetition of programs to compare the results, the inclusion of duplicate instructions, and the "reasonableness checks" also fit into this category.

Hardware redundancy includes the components and/or circuits that serve to provide fault tolerance. Two distinct approaches to the use of hardware redundancy are masking and recovery. The masking, or static, approach uses a "massive" replication of each component or circuit to two or more copies. All copies are permanently connected and receive power. Component failures or logic faults are masked by the presence of other copies of the same item. The fault masking occurs instantaneously and automatically; however, if the fault is not susceptible to masking and causes an error, a delayed recovery is not possible in the masking scheme. The two most used variants of masking are component quadding for individual electronic components (Fig. 1) and triple modular redundancy (TMR) with voting (Fig. 2) for logic circuits or larger parts of a computer. Several other variants of masking redundancy have been studied but were not found practical because of various drawbacks (excessive cost and lack of practically realizable special components). The recovery, or dynamic, approach in hardware redundancy requires two consecutive actions. The presence of a fault is first detected, then a recovery action either eliminates the fault or corrects the error which was caused. The redundancy is usually introduced in a selective, rather than massive, fashion. The means of error detection are error-detecting codes, monitoring circuits, synchronization checks, duplication, and comparison of critical functions. Software and time redundancy are also applicable to the error detection. The recovery may be a repetition of a program segment, the application of an error-correcting code, or the replacement of a faulty part by a spare.

The reconfiguration of a computer into a new system without the failed part has also been investigated. It must, however, be recognized here that the computing capacity of the system has been reduced, therefore the fault has been only partially tolerated. Such computers can be designated as "partially fault-tolerant" and, therefore, not suitable in applications that require a constant computing capacity for a prescribed time.

3. APPLICATION OF FAULT-TOLERANCE IN AEROSPACE COMPUTERS

The discussion is now directed to the special problems of introducing fault tolerance into aerospace computers that carry out computations required for the guidance and control of aircraft, spacecraft, and related vehicles. On many occasions, the same computer is programmed to perform the reduction and storage of telemetry and scientific data, as well as to monitor the performance of various other subsystems. The choice among the methods of fault tolerance is strongly affected by these requirements and by the constraints imposed by the overall system design.

Ground-based digital computer installations depend on extensive diagnostic programs and on the availability of maintenance experts to restore proper functioning after a fault has been detected. In cases where a longer interruption cannot be tolerated, a second computer is held in readiness with the same set of programs as the first. Intermediate results of computation are periodically transferred from the primary to the backup computer memory. As long as the primary computer functions correctly, the backup unit performs auxiliary computing. When an error is the primary unit is detected, the backup unit assumes the role of the primary unit. In real-time operation, both computers carry out the primary programs, and frequent comparisons are used for error detection. The Electronic Switching System (ESS) No. 1 of Bell Telephone is a good example of a real-time, special-purpose computer with duplication and extensive diagnostic programs. The typical aerospace computer operates in a more constrained environment than the ground based installation. A human maintenance specialist is not available during an actual mission (the interval of time $0 \leq t \leq T$ during which the computer must remain operational) nor is storage for large diagnostic programs available. Computing at the full capacity and speed is required at critical times during the mission, with one of these times usually coming near the mission's end. There are strict power, weight, and volume constraints to be observed in this environment. An example where all of these limits are encountered is in the computer aboard an unmanned space vehicle intended for the exploration of the outer planets of the solar system.

The first extensive application of protective redundancy (of the masking type) in an aerospace computer is found in the Primary Processor and Data Storage Unit of the Orbiting Astronomical Observatory (OAO) [3]. Discrete electronic components were used in this project, which was initiated in 1960. The individual components of each logic circuit are replaced by two or four copies each, arranged in parallel, series, or "quad" (Fig. 1) arrays. This approach uses structural redundancy at the lowest level at which replication is practical. The choice of this method was based on the initial reliability goal of 0.95 for one year (8760 h) operation in space and a 15-minute launch interval during which failure rates are assumed to be 1000 times higher than during actual operations in space, and on the known failure rates of the components. A pre-launch period of 3240 h was added subsequently to the specification. The resulting reliability prediction was about 0.75 for the entire period of operation (including data storage). The complete Primary Processor and Data Storage unit contains approximately 70,000 component parts and approximately 212,000 magnetic cores for data storage. The Data Storage Unit uses storage in quadruplicate for command words (256 locations) and in duplicate for data words (8192 locations). Storage redundancy is

needed because some memory circuits require high precision and cannot be protected by component replication, which requires wider circuit tolerances.

A second important use of protective redundancy is found in the guidance and control computer for the Saturn V launch vehicle [4]. The design of this computer was initiated in 1961, with deliveries required for the 1964-1968 period. The specified reliability goal was 0.99 for a mission time of 250 h and it led to the choice of triplication of the logic elements with voting at selected locations (Fig. 2). An exception was the core memory that was protected by duplication, parity checking for error detection, and monitoring of drive currents by special circuits. The TMR part of the computer consists of seven modules averaging 13 voted outputs for each module. Extensive computation of the reliability using a Monte Carlo simulation was used to verify the adequacy of the design, because the redundant system is too complex for an analytic calculation of the reliability. The results indicated a 250-hour reliability of 0.9992 for the TMR portion of the computer which had a predicted reliability of 0.973 for the non-redundant design. Ground maintenance and checkout are facilitated by the attachment of disagreement detector (DD) circuits to the voting circuits. A vote of 2 to 1 is indicated by the DD and identifies a triplet of modules in which one module has failed and is being masked by the other two.

It is noted that the first two cases of redundant design used the masking rather than the recovery approach in the processors, as well as a complete word replication in memory. Both processors do not need error-detecting circuitry and have immediate fault-masking without interruption of operation. The conversion from a nonredundant to the "massively" redundant processor is therefore straightforward. Conversion is accomplished when either the individual circuits are redesigned with replication components or locations for voters are chosen and the voters are designed. The application of the recovery approach is more complicated than this conversion. New functions of detection and recovery must be incorporated into the computer. The following discussion relates the problems encountered in the use of masking redundancy in aerospace computers and compares it to the recovery method.

The two computers described above represent current application of hardware redundancy. The usefulness of the component redundancy approach of the OAO depends on the validity of the assumption of independent failures in the components forming a "quad." Miniaturization and integration of electronic circuitry places the components very close together and therefore largely invalidates the independent failure assumption. Another drawback is the very difficult circuit design problem. Proper operation must continue after a failure causes a change in circuit parameters that may shift the operating point or increase dissipation in semiconductors of the circuit. The redundant circuits dissipate more power, are slower, and need higher-precision components when the same performance is required as is of a non-redundant circuit. Pre-mission maintenance and checkout of component-redundant systems also present problems. The life of the system begins with the interconnection of the redundant circuits to form a system. The low-level masking makes component failures undetectable at the system outputs until a complete circuit failure occurs and leads to a system failure. Mission times must be defined as including the entire time interval between assembly and the end of the mission. These serious drawbacks make the extensive use of component redundancy unlikely in the future. An exception to this statement may occur in relation to a few critical circuits of a system that cannot be protected by other methods. These circuits can be built of discrete components and replaced with a new assembly before the start of each mission. Another application of component redundancy is possible when non-electronic, for example fluidic, logic circuits are used.

The TMR method of protective redundancy avoids several of the drawbacks of component redundancy. Circuits remain internally non-redundant, and computing occurs in three identical channels which can be adequately separated in order to retain the independent failure assumption. Pre-mission maintenance and checkout is facilitated by the use of disagreement detectors at the voters. The validity of the independent failure assumption critically depends on the inter-channel isolation and synchronization which must be provided at each voter which receives inputs from all three channels. A fault which can affect two adjacent channels through a voter, or by any other path, is not masked and will cause a system failure. The initial reliability in a TMR system is high, but the number of independent failures which it can tolerate before failing is much smaller than a component-redundant computer, and the reliability drops sharply after the initial period. The difference is emphasized by the comparison of the 8760-h mission of the OAO to the 250-h mission of the Saturn V.

An alternate approach which competes with masking redundancy is a replacement system (RS). The RS provides the required computing capacity as a "standard" computer which consists of a number of operating functional units (arithmetic, memory, etc.). Each unit has one or more spares which are held in a standby (usually unpowered) condition. The RS also has a "monitor unit" which detects the presence of a fault in any operating unit and carries out the recovery operation. Transient faults must be identified and their effects corrected by repetition of a computation. Permanent faults are corrected by the replacement of the faulty unit. It is disconnected and the unit spare is switched into the standard computer. Several advantages of the RS over the masking approach are shown in the use of aerospace computers:

1) All spares of a unit are utilized before system failure occurs.

2) System reliability will increase if unpowered spare units have a lower failure rate (usually, some failure modes are not present when power is not applied), and parts of the standard computer are shut down during idle periods.

3) Power requirements are lower because only one copy of each replaceable unit is powered.

4) The system contains a built-in capacity for premission maintenance and checkout.

5) The number of spares for a given unit may be varied as a function of the unit's reliability estimate, the mission time, and the weight and volume constraints, while the power requirement is not changed.

6) Replacement by switching and power removal can provide strong fault isolation between units.

7) The automatic maintenance features of the RS can be extended from the computer to service the entire aerospace system which contains the computer.

All of the above listed advantages are based on the existence of the monitor unit (often called the "hard core") which carries out fault detection and recovery operations with a sufficiently high reliability and without causing intolerably long interrupts in the current computation. The lack of such monitors has been the main reason why the advantages of the RS have not been realized in current aerospace computers.

4. ARCHITECTURE OF THE STAR COMPUTER

The advantages of a replacement system are vital to the unmanned spacecraft with the mission of exploring the outer planets of the solar system. These advantages and the limitations of masking redundancy which were presented in the preceding section led to the design and construction of a laboratory model ("breadboard") of a fault-tolerant computer called the Self-Testing-And- Repairing (abbreviated "STAR") computer [5]. The STAR computer is a replacement system which provides to the user one standard configuration of functional subsystems with the required computing capacity. The standard computer is supplemented with one or more spares of each subsystem. The spares are held in an unpowered state and are used to replace operating units when a permanent fault is discovered. The principal features of the STAR system which are used to impletment error diagnosis and recovery are listed below:

1) All machine words (data and instructions) are encoded in error-detecting codes to provide concurrent fault diagnosis, i.e., fault detection occurs concurrently with the execution of the programs.

2) The computer is subdivided into a number of replaceable functional units containing their own operation code decoders, and sequence generators. This decentralization of the system allows simple fault location procedures and simplifies system interfaces.

3) Fault detection, recovery, and replacement are carried out primarily by special purpose hardware. In the case of memory damage, software is used to augment the recovery hardware.

4) Transient faults are identified and their effects are corrected by the repetition of a segment of the current program; permanent faults are eliminated by the replacement of faulty functional units.

5) The replacement is implemented by power switching: units are removed by turning power off, and connected by turning power on. The information lines of all units are permanently connected to the busses through isolating circuits; unpowered units produce only logic "zero" outputs.

6) The error-detecting codes are supplemented by monitoring circuits which serve to verify the proper synchronization and internal operation of the functional units.

7) The "hard core" test-and-repair hardware is held to a small size and protected by complete replication and immediate replacement.

The block diagram of the STAR computer is shown in Fig. 3. The replaceable units of the system are shown in a circular arrangement. Communication between the units is carried out on two four-wire busses: the Memory-Out (MO) Bus, and the Memory-In (MI) Bus. The three-letter abbreviations designate the following units:

COP - the Control Processor - contains the location counter and index registers and performs modification of instruction addresses before execution.

LOP - the Logic Processor - performs logical operations on data words (two copies are powered).

MAP - the Main Arithmetic Processor --performs arithmetic operations on data words.

ROM - the Read Only Memory, 16384 permanently stored words.

RWM - a Read-Write Memory unit with 4096 words of storage (at least two copies always powered, twelve units are directly addressable).

IOP - the Input/Output Processor - contains a transfer register and peripheral unit buffer registers.

IRP - the Interrupt Processor - provides communication links with systems outside the computer.

TIP - the Timing Processor - acts as an interrupt-generating interval timer.

The choice of the functional units represents a typical set of functions required in an aerospace computer. The functional units of the STAR computer may be thought of as a collection of small special purpose computers tied together by the MI and MO (4-wire) information busses. Computer words of 32 bits are transmitted on these two busses as 8 bytes of 4 bits each. Three control signals are sent from the Test-And-Repair Processor (TARP) through the 3-wire Control Bus to synchronize the operations of the functional units. Otherwise the functional units operate autonomously.

The monitor of the STAR system is shown in the center of the circle in Fig. 3 and is designated as TARP (Test-And-Repair-Processor). The TARP monitors the operation of the STAR computer by two methods. In the first method, an error-detecting code has been applied to all data and instruction words which are transmitted on the MI Bus and MO Bus. The TARP contains two "Bus Checkers", which test every word for validity of its code. In the second method, a four-wire unit status code is received on the Status Lines from each powered unit. The unit status codes indicate the operational status of the unit ("On", "Active", "Complete"), report "Special" conditions and also issue "Internal Fault" warnings detected by monitoring circuits within the unit. By recording the current instruction and its location, the TARP internally

reconstructs the proper status for each unit. A deviation from normal operation causes an interruption of normal computing and an entry into the recovery mode.

Unless otherwise noted, one copy of each unit is powered at any given time. The decentralized organization allows a simple standard interface between each unit and the remainder of the computer. This organization also simplifies the control of the other units by the TARP and provides simple error detection procedures. Each STAR computer unit interfaces with the computer by the means of 16 wires. These wires in both active and spare units are permanently connected to their respective computer system busses, but are isolated in such a fashion that removal of power from a given unit removes its ability to affect the busses. The external connections of a typical unit are shown in Fig. 4. The four "Input" and four "Output" lines are connected to the information (MI and MO) busses. They receive and send coded machine words in four-bit bytes. The SWITCH ON command input closes the power switch. The three "Control" input signals are: CLOCK, a 1 mHz timing input; SYNC, a periodic synchronization signal; and RESET, a signal which forces the functional unit into a standard initial state. The "Unit Status" lines supply the unit status signals to the TARP. The TARP initiates all recovery and replacement actions on the basis of the status signals received from the powered functional units and from the Bus Checkers of the system. Each functional unit is autonomous and contains its own sequence generator as well as storage for the current operation code, operands, and results. The internal design of a unit may be altered without affecting other units as long as the interfacing specifications are observed.

All numeric operand words and instruction words in the STAR computer are encoded with error-detecting codes, as shown in Fig. 5. The numeric operand words are represented as 32-bit residue-coded numbers consisting of seven 4-bit bytes (b(0) through b(6)), which is the binary representation of the word b, plus a 4-bit check byte c(b). The check byte is established by

$$c(b) = 15 - (15 \mid b)$$

where "15 | b" means "the modulo 15 residue of the number b." This check byte results in the complete 32-bit word being a multiple of 15. Checking of a numeric operand word is accomplished by computing the modulo 15 residue of the word. A zero residue, represented by "1111" indicates a correct word; all other values of the residue indicate a fault. A 32-bit instruction word consists of a 12-bit operation code and a 20-bit residue-coded address part. The address part is encoded in the residue code with the check modulus 15. The address part consists of a 16-bit binary number a and a 4-bit check symbol c(a) such that $c(a) = 15 - (15 \mid a)$. The address part is checked using the same checking algorithm that is applied to numeric operand words. The operation code is divided into 3 bytes of 4 bits each. Each byte is encoded in a "two-out-of-four" code which allows byte-wise detection of op-code errors. There are six valid combinations for every four-bit byte, or a total of 216 variants of op-codes.

Residue codes were chosen for error detection in the STAR computer because they have the property that the residue check byte can be operated upon independently in arithmetic operations. The residue coding can be used to check arithmetic processing as well as data transmission. An earlier version of the arithmetic unit employed a product code y = 15x for operand words. The residue code was substituted because of its more convenient double-precision algorithms. An extensive study of arithmetic codes preceded the design and demonstrated that these codes need only a very simple Bus Checker for validation and have fully satisfactory error detection properties [6].

5. OPERATION AND RECOVERY PROCEDURE

The STAR computer has two modes of operation: the working mode and the recovery mode. During the working mode the stored programs are carried out. The TARP processor issues the principal 1 mHz CLOCK signal and the SYNC signal which occurs when a new step is initiated in the execution of an instruction. One standard instruction cycle is executed in three steps. In the first step, the address of the instruction is sent from the Location Counter in the COP to the Memory (ROM and RWM) units. In the second step, the addressed Memory unit broadcasts on the MO bus the operation code and address of the instruction to all functional units. The address is indexed in the COP which transmits it to the MI bus if necessary. The appropriate units recognize the operation code, store the address, and initiate execution. In the third step the instruction is executed, an operand is placed on the appropriate bus and accepted by the destination unit. The Bus Checkers test every word transmitted on the busses for proper encoding. An interrupt cycle omits the first step. During the second step an instruction is broadcast by the interrupting unit (IRP or TARP). In order to distinguish between "standard" and "interrupt" cycles, a 2-out-of-4 encoded "step-code" byte is issued to the information bus by the TARP simultaneously with the SYNC signal at the beginning of each step.

The Read-Only Memory (ROM) contains the permanent programs and the associated constants for a given mission. The experimental model uses a "braid" assembly of transformers and wires for the permanent storage of binary information. Complete replicas of the memory are used as replacements. Each Read-Write Memory (RWM) unit has two modes of operation. In the standard mode a RWM unit recognizes its own wired-in unit address. In the relocated mode, a RWM unit records and from then on responds to the unit address of another ("main") unit. The relocated unit stores and reads out the same words as its main unit. Each unit internally compares its readout to the word appearing on the MO bus. In case of a disagreement, a SPECIAL status signal is sent to the TARP which also receives the Checker result and identifies the RWM unit supplying a damaged word. The relocated mode provides duplicate or triplicate storage for critical data. In the case of failure of a RWM module, its replacement unit can be assigned the same address, avoiding a discontinuity in addresses. Assignment and cancellation of relocated mode is performed under program control; this provision allows for efficient memory utilization as well as for selective redundancy of storage. A record of RWM module assignment is retained (in non-volatile storage) in each copy of the TARP.

The recovery mode of operation is initiated by the RESET signal to all units from the Test-And-Repair Processor (TARP) when a fault is indicated by the Bus Checkers or by the "Unit Status" inputs. Then the TARP issues an unconditional transfer instruction and a segment of the current program is repeated. If the fault recurs, the faulty unit is replaced with a spare. A fault requires one of two procedures: either only one or a sequence of instructions need to be repeated. To provide recovery in the latter case the

STAR computer has an instruction which stores a "rollback" address in the TARP. At the time of setting the "rollback" register all computer words which will be needed by the program in the future must be in nonvolatile and redundant storage (at least two separate RWM memory modules). The TARP also has registers for the storage of the following data which is needed to decide whether all units operate when required: the present instruction, its location, RWM module assignments, and the source and type of an observed fault. For cases of temporary power loss and irresolvable fault conditions, the TARP contains a wired-in "Disaster Restart" procedure which begins with all other units in unpowered condition.

The TARP is the "hard core" of the system. Three fully powered copies of the TARP are operated at all times together with n standby spares. All outputs of the TARP are decided by a two-out-of-(n +3) threshold vote. When one powered TARP disagrees with a voted output, it is immediately returned to the standby condition and one of the standby units receives full power and joins the powered triplet. The removed one is kept as a standby unit and reuse is attempted when other spares are exhausted. Reuse succeeds if previous removal had been caused by a transient fault. Two spares (n = 2) are used presently. Because of the three unit requirement, design effort has been concentrated on reducing the TARP to the least possible complexity. Experience with the present model is expected to lead to modifications of the design. The replacement of faulty units is commanded by the TARP vote and is implemented by power switching. It offers several advantages over the switching of information lines which connect the units to the bus. The number of switches is reduced to one per unit, power is conserved and strong isolation is provided for catastrophic failures. Magnetic power switches have been developed which are part of each unit's power supply and designed to open for most internal failures. The information lines of each unit are permanently connected to the busses through isolation circuits. The signal on the bus is the logic OR of all inputs from the units. The power switch and the busses utilize component redundancy and careful electronic design for protection against fatal "shorting" failures.

6. STAR COMPUTER SOFTWARE AND RELIABILITY ASSESSMENT

Two major software research and development efforts were initiated concurrently with the design and construction of the STAR computer breadboard [7]. First, a software effort was initiated in 1967 and is presently well under way; the completed systems programs include an assembly program, loader, and functional simulator. The operating system is presently being designed. A subset of the STAR breadboard successfully carried out its first program in March of 1969. Second, a study of reliability estimation for STAR-like fault-tolerant replacement systems is nearing completion. The study has led to the development of the CARE (Computer-Aided Reliability Estimation) program, which can be used for interactive evaluation of various fault-tolerant designs.

The STAR Computer Software System is partitioned into two separate subsystems. The first subsystem, the programming subsystem, consists of three modules: an assembler, a loader, and a functional simulator. The second subsystem, the operating subsystem, consists of two modules: the resident executive module and the applications programs module. The modules of the programming subsystem are static in nature: once they are complete they require little change. These modules are being developed on the UNIVAC 1108 system of the Scientific Computing Facility at JPL. An executive program has also been developed to facilitate coordinated use of the assembler, loader, and simulator. The modules of the operating subsystem are dynamic in nature: they will be constantly changed as the STAR computer hardware-software system is developed. The modules for a typical space mission would consist of a resident executive module and a module comprised of several applications programs, each of which would perform the computation for a particular aspect of the mission. Facilities are included in the operating subsystem which implement the timekeeping functions required on a spacecraft computer.

The first module of the STAR Computer Software System is SCAP: the STAR Computer Assembly Program. Programs for the STAR computer are written in SCAL: the STAR Computer Assembly Language. The input to SCAP consists of a source program written in SCAL; the output consists of binary card images in an intermediate language used as input to the loader. SCAP is a traditional two-pass assembler incorporating machine instructions, pseudo-operations, macro facilities, and a unique COMPILE pseudo-operation which implements automatic compilation of simple arithmetic statements by the assembler. For instance, on a single-address machine like the STAR computer, incrementing a counter usually requires three instructions: a load or clear-and-add type, an add, and a store. These three instructions could be automatically compiled by writing COMP Y = Y + 1 in SCAL. Any valid statement involving =, +, -, *, /, and parentheses is allowed. This pseudo-operation is a powerful feature of the STAR Computer Assembly Language. The second module of the STAR Computer Software System is LOAD: the STAR Computer LOADer. This phase is unique in that it is the only phase programmed on two different computers. Initially LOAD is written for the same computer as the assembler and simulator (the UNIVAC 1108). This is essential since the loader is the interface between the assembler and the simulator. LOAD is also programmed for the STAR computer itself so that various binary decks (and possibly patches) may be put together and loaded immediately into the STAR computer without an additional run on the 1108. Thus two alternatives are available to the STAR computer programmer: programs may be assembled and loaded onto a binary magnetic tape or binary punched cards by the 1108 ready for immediate use by the STAR computer; or programs may be assembled by the 1108 and loaded by the STAR computer. The third module of the STAR Computer Software System is the functional simulator. The STAR computer is being constructed as a breadboard to permit experimentation with the self-testing-and-repairing features and the associated algorithms. A functional simulator is being included in the programming subsystem so that programs may be developed for the STAR computer before hardware construction is completed. Furthermore, by using the simulator to debug programs before they are run on the STAR computer, hardware debugging of the computer can be simplified.

An executive program has been developed for coordinated use of the assembler, loader, and simulator. Using this executive, source and relocatable binary decks can be combined to form a load module for the STAR computer. This load module may be written on tape, punched on cards, and input to the simulator for simulated execution. Thus, all modules of the programming system can be utilized during a single run on the 1108 computer. Control cards may be included in the input deck to select the various options available. The modules of the operating subsystem of the STAR Computer Software System, consisting of the resident executive module and applications programs module, are currently being developed. The fourth module of the STAR Computer Software System is the resident executive module. The most important function of this

module is to provide software required to implement program resumption after error detection. Other functions implemented in this module are scheduling, input-output control, interrupt processing, and diagnostics. The fifth module of the STAR Computer Software System is the applications programs module. Common mathematical subroutines such as floating-point, double-precision, sine, cosine, square root, etc., may be recorded in STAR computer read-only memory. The read-write memory will be used for developing programs on the STAR computer. It is anticipated that guidance and control programs can be written and run on the STAR computer by simulating sensor and control signals at the STAR Computer Input-Output Processor.

The projected interplanetary mission requirements for digital computers designed to perform in space environment are in the order of ten or more years. In order to determine, evaluate, and compare the reliability of STAR-like configurations, the methodology for reliability evaluation of such systems needed first to be developed. Since the current methods (based either on simulation or on parts count) of measuring the expected reliability of a computer system do not cover all aspects of the STAR computer organization, a reliability estimation study was initiated for the STAR project. This study is concerned with several interrelated tasks. One is the development of reliability equations which represent the reliability of STAR-like fault-tolerant configurations. The second is to analyze with respect to significant parameters the various redundancy techniques which have been utilized. Third critically compares the various redundancy techniques which are available and determines figures of merit and guidelines for their optimum usage. These tasks lead to the development of a mathematical reliability model of generalized STAR-like computers which will permit the analytic evaluation of the significant reliability parameters [8]. The foregoing reliability study led to the development of a reliability design tool called the CARE (Computer-Aided Reliability Estimation) program. CARE is a software package currently being developed on the UNIVAC 1108 computer system at JPL. CARE may be interactively accessed by a designer from a teletype console to calculate his reliability estimates. The input is in the form of a system configuration description followed by queries on the various reliability parameters of interest and their behavior with respect to mission time, fault-coverage, failure rates, dormancy factors, allocated spares, and cost in hardware. The CARE subroutines are periodically updated to reflect refinements made on existing reliability models and enlarged as further data becomes available.

The preliminary reliability estimates of the STAR computer were based on the following reliability model. The functional processors comprising the STAR are considered to be in series for their reliability estimation. The TARP unit which is the "hard-core" of the STAR organization is a triplicated configuration with a number of spares. The remaining functional processors are in a replacement mode utilizing standby spares. The reliability model reflecting this organization was extended by incorporating additional parameters to reflect dormancy effects of the spare units. Figure 6 shows the comparative survival probabilities of the STAR computer and a non-redundant computer having the same computational capability as the STAR. The bounds on the reliability versus mission time (in years) are shown corresponding to the cases of no failures of the spares (upper bound) and equal failure rates for spare and active units (lower bound) for the STAR computer with an allocation of three spares for each processor. The mathematical models on which these results are based are undergoing further refinement to reflect the reliability of the STAR computer with greater exactitude. For the above reasons the computer figures provided at this time are to be considered as a first-cut estimate.

7. AUTOMATIC SPACECRAFT MAINTENANCE: EXTENSIONS OF STAR TECHNIQUES

An immediate consequence of the automatic maintenance provided in the STAR computer is the reduction of the ground checkout equipment to a special functional unit (an expanded TARP) which contains programs for the checkout of all copies of the STAR function units. Periodic checkout of spare units during a mission is also readily implemented by an on-board program. These features lead to the conclusion that automatic maintenance (testing and replacement) of subsystems may be extended beyond the computer to include other parts of the spacecraft. Sensors and other parts of the guidance system, communication equipment, and scientific instruments can be monitored, periodically tested, and replaced by spares (using power switching) following the techniques used for the replaceable processor and storage units of the STAR computer. The fault-tolerant STAR computer can be used as the monitor and "automatic repairman" for the entire spacecraft. Its memory units contain the calibration, diagnosis and emergency action programs for the maintenance of the entire spacecraft. The redundancy provided by unpowered spare copies of spacecraft subsystems offers very important advantages over other types of redundancy. First, the power requirements are not increased by the introduction of redundancy. Second, the unpowered spare copies will have a lower failure rate whenever additional failure modes are introduced by the presence of power in a subsystem. Both properties are exceptionally important in the missions to the outer planets of the solar system, which require a very long survival of the spacecraft (over 10 years), and which are subject to very strict power constraints [1].

The practical implementation of automatic checkout and continuous in-flight maintenance of the entire spacecraft poses several additional requirements on the designers. First, methods of internal fault detection which are most effective for each "black box" subsystem of the spacecraft must be incorporated within the subsystem. Second, a standard interface with the STAR computer (i.e., its Interrupt Processor and I/O Processor) must be devised, which will permit the STAR computer to perform the necessary monitoring and test operations. Third, power switching must be adapted for each "black box." Fourth, test and calibration procedures must be devised and test programs must be written and stored in the Read-Only Memory unit of the STAR computer for each one of the maintainable (replaceable) parts of the spacecraft. It must be noted that all of the above functions (except power switching) are presently implemented by ground checkout equipment which is used to prepare the spacecraft for its mission, and that this accumulated experience is applicable to the development of in-flight testing and replacement. A study of the automatic in-flight maintenance of the spacecraft is presently under way, and several promising applications have already been identified.

To illustrate the application of automatic maintenance principles, we consider the monitoring of the propulsion system performance in a spacecraft. The sensors which provide data to the computer indicate the pressures in the fuel line, the oxidizer line and in the combustion chamber, as well as the combustion chamber temperature. If an abnormal fuel or oxidizer flow condition is detected, the motor is shut down and an alternate valve is operated. A record of the event is stored for future reference and for telemetry.

The chamber temperature and the accelerometer outputs are also monitored by the computer, and if abnormal conditions are detected, an adjustment is made or the motor is shut down, according to the maintenance program. Another example is the monitoring of gas leakage in the attitude-control system; a standby valve can be employed when excessive gas leakage occurs.

The application of automatic maintenance offers a significant improvement in the expectation of mission success and should also be useful in many other critical applications. High-speed aircraft and automated systems for the monitoring of patients in hospitals of the future are examples of this usage.

8. PRESENT STAR COMPUTER STATUS AND CONTINUING EFFORTS

The STAR computer employs a balanced mixture of coding, monitoring, standby redundancy, replication with voting, component redundancy, and repetition to attain hardware-controlled self-repair and protection against transient faults. The principal objective of the design is to attain fault-tolerance for the widest possible variety of faults: transient, permanent, random, and catastrophic. The actual construction (rather than simulation) of the STAR breadboard has two significant advantages. First, the design process has uncovered interesting new problems and led to numerous improvements. Second, the computer is intended to serve as a vehicle for further experimentation and refinement of the recovery techniques, (especially in the protection of the TARP and in the design of buses and power switches). An extensive testing and validation program is being planned for the STAR breadboard. A better understanding of recovery from transient faults will be sought by actual injection of such faults and observation of the recovery procedure. Automatic test data collection and reduction will be provided. The design of hardware and software for this purpose has been initiated.

At the present time (April 1970) the STAR computer breadboard has been in limited operation for a year (Fig. 7). The present breadboard contains one complete set of functional units shown in Fig. 3, with the exception of the Timing Processor (TIP). The TARP contains a subset of parts which control the "working" mode of operation; the hardware to control the "recovery" mode is nearing completion. Two complete Read-Write Memory units are included in the breadboard. The Input-Output and Interrupt processors consist of laboratory I/O equipment: card reader, typewriter, and magnetic tape unit. Very extensive manual controls for pulse-by-pulse operation have been provided for every functional unit. Display lights are provided for all the principal registers of the units. The display outputs also provide points for the automatic collection of performance data during dynamic testing of recovery operations. An all-magnetic transformer-type power switch has been constructed and tested; work continues on further refinements of the switch.

Design of several improved "second generation" functional units is under way. They include a new Arithmetic Processor, a Control Processor for medium-scale integrated circuit implementation, and a "shared" Read-Write Memory module for the storage of automatic maintenance information from the spacecraft telemetry system. The "shared" RWM is the result of a concentrated study of automatic spacecraft maintenance under control of the STAR computer, and it serves as a very effective Input/Output Processor for the monitoring of spacecraft performance. Analysis of automatic maintenance algorithms and design of a Command/Data bus for their implementation are under intensive study. The second version of the programming subsystem in the STAR Computer Software System has been completely implemented on the UNIVAC 1108 computer at JPL. Work is currently under way on the resident executive program. A few application programs, including a floating point subroutine, have been completed; others are under way. The CARE (Computer-Aided Reliability Estimation) program is operational; further refinements are continuously being introduced.

The investigations of the STAR computer have stimulated new research efforts in fault tolerance. Current advanced investigations are concerned with the following areas:

1) Hardware-software interaction in a fault-tolerant system with recovery, such as the interaction of the TARP and the operating system, the automatic verification and insertion of "rollback" points in all programs, and auxiliary diagnostic programs.

2) Studies of advanced recovery techniques, i.e., post-catastrophic restart, TARP replacement schemes, recovery from massive interference.

3) Computer-aided reliability prediction for STAR-like fault-tolerant systems.

4) Advanced component technology, especially methods to attain bus and power switch immunity to faults.

5) Formulation of a theory of fault-tolerance by interpretation of extensive experiments with the STAR breadboard as the instrument.

6) The design of a "Super-STAR" computer with "microprogrammable" processor and storage modules, and their implementation by large-scale integration.

Another area of utilization of the STAR breadboard is in the planning of future space missions. The breadboard serves as the model for the construction of flighworthy prototypes and as the training device for future users, including the designers of other systems of the spacecraft. In 1969 the Thermoelectric Outer Planet Spacecraft (TOPS) research project was initiated at the Jet Propulsion Laboratory. Its objective is the design of the prototype of a spacecraft suitable for missions to the outer planets of the solar system. An effort to design a STAR-type computer for the guidance, control, and automatic maintenance of this spacecraft is currently in progress.

On the basis of current results, it may be expected with reasonable certainty that the STAR computer project will yield both a more refined insight into the problems of computer self-repair and a model for advanced spacecraft computers which will perform guidance, control, and maintenance operations on missions exploring the most remote realms of the solar system as well as the interstellar space.

9. ACKNOWLEDGMENTS

The research and development related to the STAR computer has been performed in the Flight Computers and Sequencers Section of the JPL Astrionics Division, and recognition is due to most of the Section's members for support in their respective specialties. The STAR concept has been devised by A. Avižienis, who has directed the overall technical effort. The hardware effort is directed by D. A. Rennels, the software effort--by J. A. Rohr, reliability estimation--by F. P. Mathur, and the implementation of spacecraft automatic maintenance--by G. C. Gilley. Major technical contributions to the design have been made by M. T. Ghosn, P. H. Sobel, and A. D. Weeks, and extensive consultation has been contributed by R. K. Caplette, G. R. Hansen, E. H. Imlay, G. R. Kunstmann, G. C. Milligan, J. Nievergelt, D. K. Rubin, and J. J. Wedel. R. A. Easton of the Data Handling System Section has made valuable contributions in the development of interfacing with the Computer-Accessed Telemetry System. Administration of the research has been directed by J. R. Scull (Astrionics Division Manager), W. F. Scott (Flight Computers Section Manager) and J. J. Wedel (Research Group Supervisor). The power switch has been developed by the Stanford Research Institute, Menlo Park, California, and a fault-tolerant Read-Only Memory has been designed by the M.I.T. Instrumentation Laboratory, Cambridge, Massachusetts, under subcontracts from JPL.

A special acknowledgment is due to R. V. Powell, Manager for Electronics of the JPL Research and Advanced Development Program Office and Messrs. F. J. Sullivan (Director, Electronics and Control), J. I. Kanter, G. A. Vacca, J. L. East, and T. S. Michaels of the NASA Office of Advanced Research and Technology, Washington, D.C., for their continued advice and encouragement of the STAR computer research and development efforts.

10. REFERENCES

[1] Long, J. E., "To the outer planets," Astronautics & Aeronautics, Vol. 7, No. 6 (June 1969), 32-47.

[2] Avižienis, A., "Design of fault-tolerant computers," AFIPS Conference Proceedings, Vol. 31 (Fall Joint Computer Conference 1967), Thompson Books, Washington, D.C., 733-743.

[3] Lewis, T. B., "Primary processor and data storage equipment for the orbiting astronomical observatory," IEEE Transactions on Electronic Computers, Vol. EC-12, No. 6 (December 1963), 677-686.

[4] Anderson, J. E., and Macri, F. J., "Multiple redundancy applications in a computer," Proc. 1967 Annual Symposium on Reliability, Washington, D.C. (January 1967), 553-562.

[5] Avižienis, A., "An experimental self-repairing computer," Proceedings of the 1968 Congress of the International Federation for Information Processing, Edinburgh, Scotland (August 1968), Preprint Booklet E, pp. E29-E33.

[6] Avižienis, A., "Concurrent diagnosis of arithmetic processors," Conference Digest of the 1st Annual IEEE Computer Conference, Chicago, Illinois, 1967, 34-37.

[7] Avižienis, A., Mathur, F. P., Rennels, D., and Rohr, J., "Automatic maintenance of aerospace computers and spacecraft information and control systems," Proc. of the AIAA Aerospace Computer Systems Conference, Paper 69-966, Los Angeles (September 8-10, 1969), 1-11.

[8] Mathur, F. P., and Avižienis, A., "Reliability analysis and architecture of a hybrid-redundant digital system: generalized triple modular redundancy with self-repair," AFIPS Conference Proceedings, Vol. 36 (Spring Joint Computer Conference 1970), AFIPS Press, Montvale, N. J.

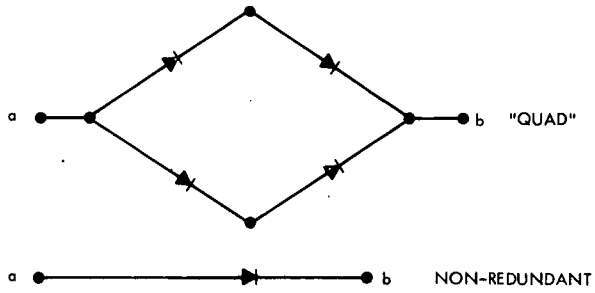


Figure 1. "Quadded" components (diodes)

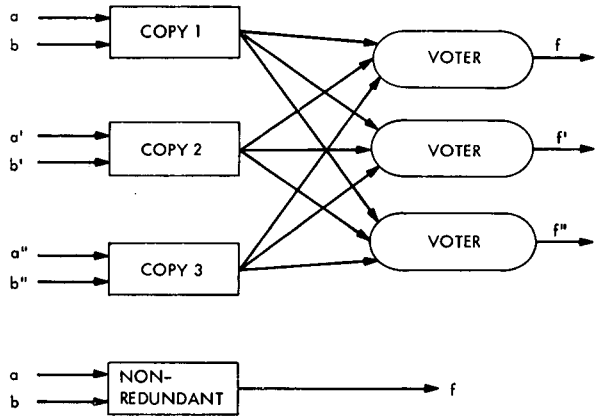


Figure 2. Triple-modular redundancy with voting

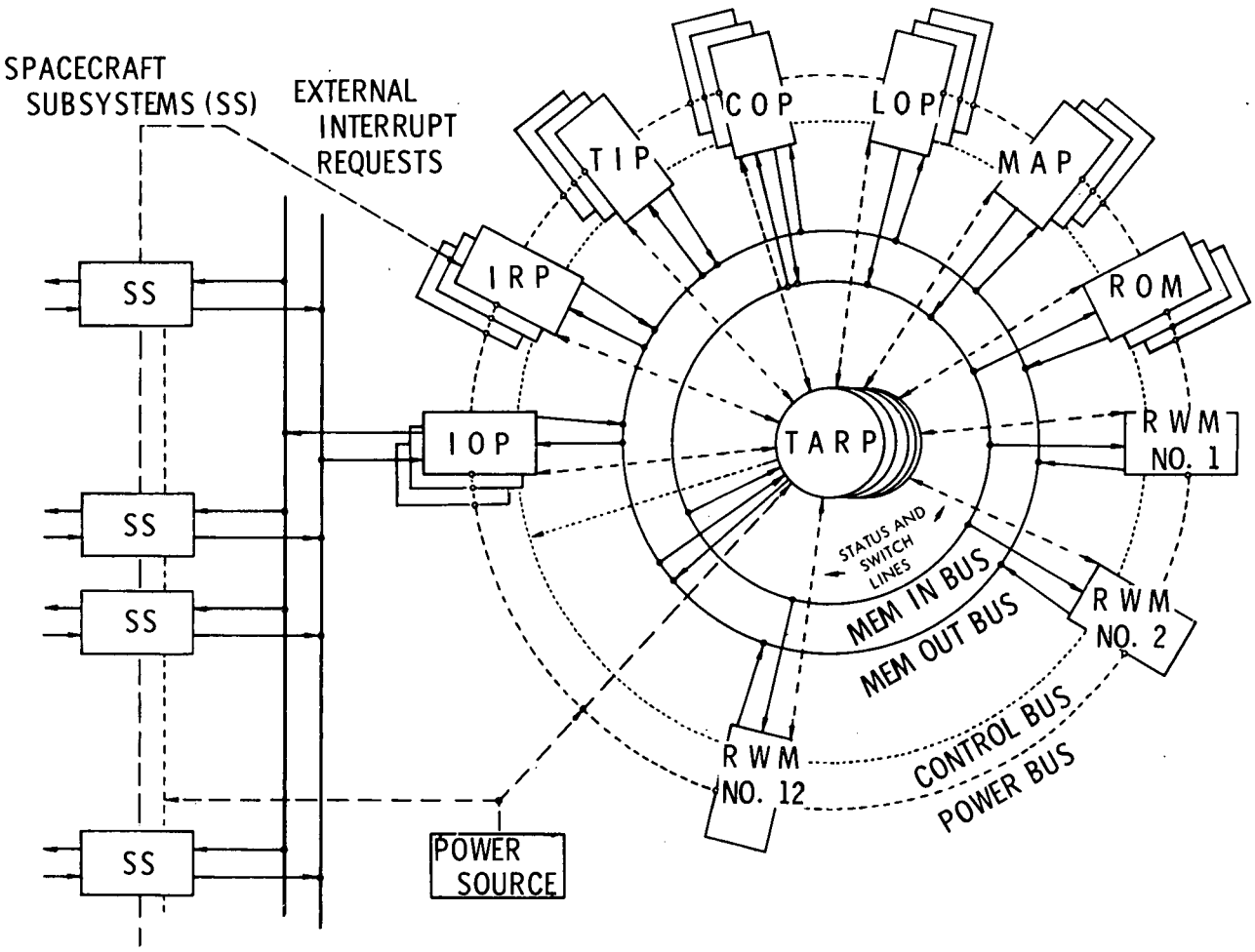


Figure 3. STAR computer block diagram

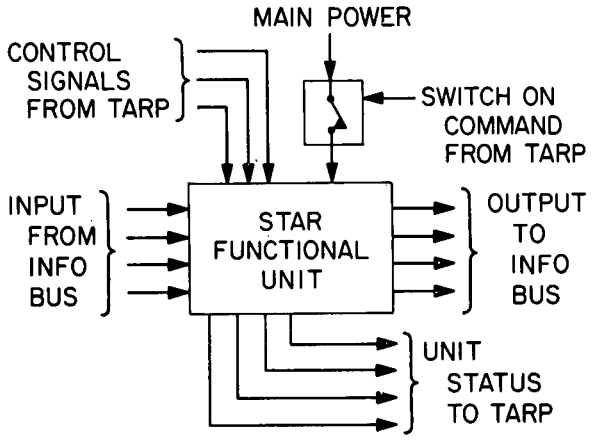


Figure 4. Typical STAR functional unit

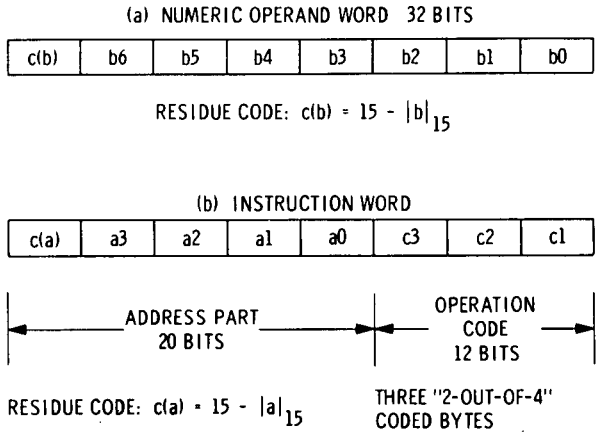


Figure 5. STAR computer words

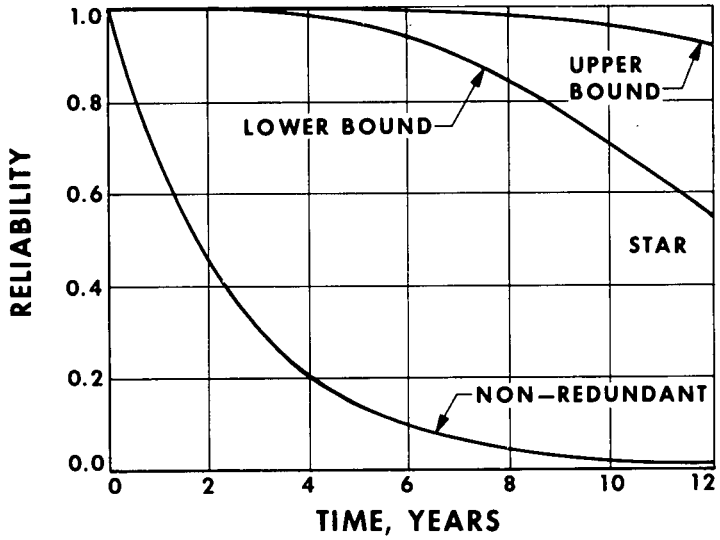


Figure 6. Upper and lower bounds on the reliability of STAR computer (3 spares)

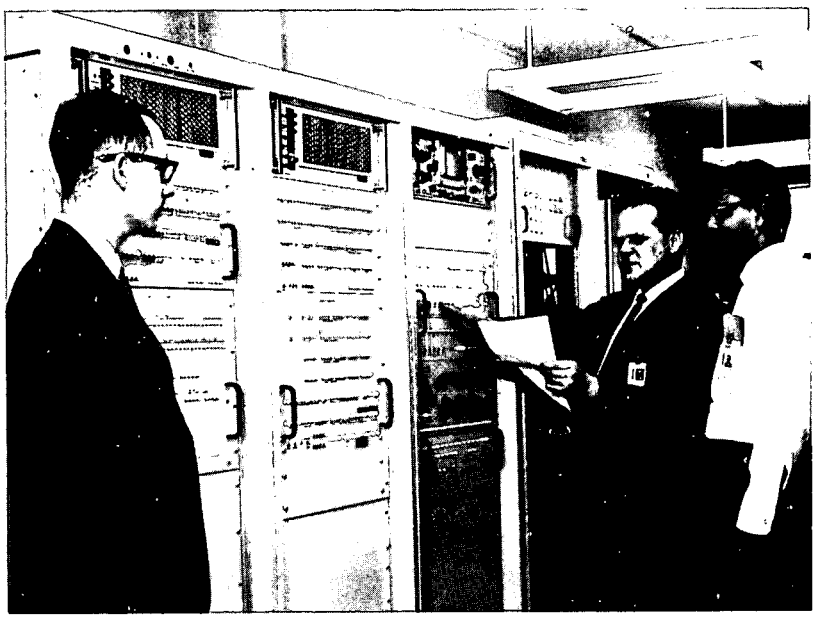


Figure 7. STAR computer breadboard

SENSITIVITY ANALYSIS OF APPROXIMATE OPTIMUM GUIDANCE PROCEDURES FOR ON-LINE OPERATION

by

Eveline Gottzein and Helmut Bittner

Messerschmitt-Bölkow-Blohm GmbH
Ottobrunn, Germany

SUMMARY

The paper is concerned with simplified guidance computation schemes for on-line operation. Under the aspect of onboard or ground station computer organisation explicit and implicit guidance methods are discussed. Based on computational requirements, possibilities for approximate explicit guidance procedures are compared.

SENSITIVITY ANALYSIS OF APPROXIMATE OPTIMUM GUIDANCE PROCEDURES FOR ON-LINE OPERATION

Eveline Gottzein and Helmut Bittner

1. INTRODUCTION

The fast progress in the development of low-weight, high-speed digital computers already achieved and still continuing has caused new aspects to become attractive for the application in space and missile guidance and control tasks. The fundamental situations to be encountered in practice are schematically indicated in the block diagrams of Figure 1. The simplest method of guiding a vehicle on its trajectory uses a precalculated nominal attitude program, which is often sufficient for the first stage flight of a multistage vehicle. In order to transfer maximum payload into a given orbit and to ensure sufficient injection accuracy in the presence of perturbations, closed loop guidance procedures have to be applied, at least for the flight phases of the upper stages. Depending on whether the guidance and control problems to be solved can be locally separated from each other or have to be solved at the same place i.e. on board of the vehicle, either radio guidance or inertial guidance may be used (see Figure 1). The first possibility allows for a big and fast guidance computer but is heavily restricted by the geographical conditions i.e. the availability of a point on the earth surface within the visibility range of the most important part of the trajectory. This way is completely out of the question, if a mobile launching point is required. The inertial guidance case does not suffer from these restrictive conditions but entrains obviously limitations on computer size and storage capacity.

Advanced vehicle concepts and the complex onboard systems associated with them call for the implementation of a digital processor for various reasons. Once it is decided that a digital computer is to be installed on board of the vehicle there are a large number of tasks to be performed conveniently by the computer apart from the navigation (purely inertial or hybrid) and guidance aspects. Disregarding for the moment its application within the prelaunch phase for

- platform alignment
- check-out assistance and
- propellant loading control,

the availability of an onboard digital computer may be taken advantage of during the actual flight phase to solve the following problems:

- attitude control
- auxiliary attitude control
- navigation, guidance and optimization
- digital filtering of the measured values of the inertial measurement system, estimation of vehicle parameters
- control of thrust and propellant feed system
- coordination of guidance and control systems
- flight sequencing
- sequencing of data transfer
- data compression and preliminary data analysis for telemetry.

In order to cope with these tasks it is required to reduce the computational efforts for each of these problems by convenient approximations. From the following we are going to discuss the aspect of guidance computation schemes suited for on-line computer operation.

2. GENERAL REQUIREMENTS

Let us first state in general form what requirements have to be met in order to fit the guidance procedure into the on-board computer organisation. The guidance procedure is expected to perform the following tasks:

- evaluation of the optimal attitude law based on the knowledge of the present state vector and the injection point to be achieved.
- prediction of the time to go in order to provide the engine cut-off signals.

Figure 2 shows a block diagram of the EUROPA 2 inertial guidance system and its interlinkage with the vehicle attitude control. The vehicle is equipped with an inertial platform providing three Euler angles for attitude

control purposes and three acceleration components in space fixed axes. For the particular example considered, the analog accelerometer signals are integrated to velocity increments and converted into digital form within the acceleration encoder and then fed into the onboard computer. Since the accelerometers measure the acceleration due to all external forces except the gravitational force, the velocity increments due to gravitation have to be calculated from position and added within the computer integration routine. The resulting position and velocity of the vehicle is converted into a coordinate system suited for guidance computations. A detailed discussion of the EUROPA 2 inertial guidance system is given in Reference 4.

When establishing guidance procedures, the following general aspects have to be taken into account from the computer point of view:

- The tasks listed in Section 1 have to be performed in real time.

  The various signals to be processed demand different sampling rates and computer resolution.

  The computer therefore has to be equipped with input and output capability of different levels like Output Control Pulses, Sense Lines, Priority Interrupts besides the data communication.

  The provision of these above mentioned control signals allows for the necessary multiprogramming scheme of different priority levels. The guidance signals having a low bandwidth only will obviously have to be calculated at a very low priority level.

- Each of the individual tasks has to be performed within fixed frame times and in order not to waste computation time the duration of the various operations should be predetermined as far as possible and only subjected to very small variations. In Figure 3 a typical diagram for onboard computer timing with different priority levels and frame times is given.

- The storage requirements for the essential program parts should stay within limits, since for reasons of reliability it is advisable to store this information twice in different memory banks.

- A reasonable compromise as for computer wordlength has to be found since the various tasks of Section 1 require different computer resolution. While for some tasks two or even one bit is sufficient and the control tasks require 7 - 8 bits, only, insufficient computer resolution for navigation and guidance might produce guidance noise at critical frequencies and excite the vehicle. This difficulty could be coped with by using multiple wordlength of course but only on cost of longer computation time.

  A word length of 18 - 24 bits seems appropriate.

3. POSSIBLE SOLUTIONS OF THE GUIDANCE PROBLEM

From the following the methods commonly used to solve the optimal guidance problem will be briefly characterized and their advantages as well as their practical limitations will be illuminated. It is to be understood, that irrespective of the particular coordinate system chosen to mathematically describe the motion of the vehicle, the injection conditions have to be expressed by a suitable set of invariant orbital parameters. Convenient formulations of the problem will therefore use coordinate systems which ensure simple relations between state variables and orbital parameters.

3.1 Explicit Guidance

The exact optimal guidance conditions represent a nonlinear two-point boundary value problem, which is to be solved at any instant of vehicle flight, where guidance commands are to be calculated. Since the necessary optimality conditions involved are too complicated to allow for an analytical solution, an iterative procedure is required. In general this is an elaborate and time consuming task. It implies, that the simultaneous nonlinear differential equations of state variables and adjoint variables have to be integrated several times over the whole time of flight for initial conditions of the adjoint variables altered in turn. From the variations of the end points of the state variables as a function of variations of the adjoint variables at the current time a sensitivity matrix can be constructed which allows to evaluate corrections of the initial values of the adjoint variables. Functional relationships of the adjoint variables determine the optimal attitude programme. The most critical drawbacks of this iterative procedure for on-line computations of the optimal guidance commands in view of the requirement of a fixed frame time are:

- variable time of integration depending on the current state of the vehicle with respect to the injection point
- the number of reoptimization cycles required to calculate the optimal attitude commands is not predetermined
- the computation time needed for each reoptimization step is fairly high.

The advantages of the explicit guidance method to solve the two-point boundary value problem are:

- the computer storage capacity required is small
- the guidance scheme is to a far extent independent of the particular orbit or mission
- the guidance procedure is flexible with respect to changes in the launch site
- the guidance method is fairly insensitive to deviations of vehicle parameters from nominal so that in most cases no updating of the vehicle model is necessary

- precalculations are only required to adjust the initial values of the adjoint variables.

Explicit guidance will be dealt with in more detail in Section 4.

3.2 Implicit Guidance

In the case of implicit guidance the nominal optimal trajectory is precalculated by means of some explicit guidance procedure. For a certain number of points along this optimal trajectory correction matrices are evaluated under the assumption that in the vicinity of the optimal trajectory the corrections of the optimal attitude commands or rather the adjoint variables and time to go are linear functions of the state variables (expressed in suitable orbital parameters). Associated sets of numerical values of flight time, nominal trajectory data (orbital parameters), nominal adjoint variables and correction matrices are stored in the computer for on-line operation.

The actual computations to be performed in real time are reduced in this guidance procedure to the following linear algebraic operations:

- evaluation of the deviation of actual from nominal trajectory (expressed in orbital parameters e.g. osculating ellipse)
- selection of "adjacent" correction matrices on the basis of flight time and computation of a proper interpolation factor
- adjustment of the adjoint variables stored in the adjacent correction matrices by means of the deviations of actual from nominal trajectory
- linear interpolation between the adjusted adjoint variables, computation of associated optimal attitude command and corrected time to go.

In Figure 4 the attitude angle as derived from such an implicit guidance procedure has been plotted. The corner points marked by arrows represent the moments for which guidance correction matrices have been stored and the bows in between are due to the linear approximation used. As is obvious from this figure the number of correction matrices required depends on the linearity range which decreases towards injection[2].

The dynamic behaviour of the carrier vehicle represented by a detailed hybrid simulation model under the influence of such implicit guidance commands is shown in Figure 5. Commanded (W_ϑ, W_ψ) and actual vehicle attitude in pitch ϑ and yaw ψ have been recorded as well as body-fixed translational acceleration (a_x, a_y, a_z) over the optimal guided trajectory of the 3rd stage of EUROPA 1 carrier vehicle. Fuel sloshing has only been included in the simulation of the pitch plane and the associated vehicle accelerations show up in the a_z-component. In the yaw plane the effect of the rigid body limit cycle (a_y-component) can be observed.

Coming back to the computational aspects of the implicit guidance procedure, the great advantage of this method is – a very short time required for on-line computations. This advantage must be paid for by the following drawbacks:

- large storage capacity is required
- extensive precalculations are necessary for changes in orbit and mission
- the guidance scheme is not flexible with respect to changes in the launch site (equivalent to a change in mission).

4. APPROXIMATE EXPLICIT GUIDANCE SCHEMES

From the discussion of the previous chapter explicit guidance procedures are more attractive for inertial guidance applications mainly from the flexibility and storage capacity point of view, provided it were possible to fit the on-line computations required into a fixed frame time of reasonable length by introducing suitable approximations.

4.1 Flat Earth Guidance

In order to circumvent the computational difficulties involved in solving the nonlinear two point boundary value problem, in the flat earth guidance procedure some simplifying approximations are made, which allow the differential equations representing the necessary conditions for optimality to be solved partly analytically[4]. The characteristic features are:

- the optimization problem is formulated in spherical polar coordinates, a rotating coordinate system with the origin in the centre of gravity of the vehicle one axis pointing to the gravitation centre, the second being parallel to the nominal orbit plane pointing into the "forward direction" and the third pointing out of the orbit plane. In these coordinates the cut-off boundary conditions assume a very simple form.
- the motion is assumed to take place in a unidirectional – hence the name "flat earth" – and constant force field. Under this assumption the differential equations of state variables and adjoint variables are not coupled with each other, the adjoint system can be integrated analytically and the optimal control law (steering angles) can be formulated in terms of the integration constants of the adjoint system.

- by means of suitable corrective functions the integration constants of the flat earth optimal control law are matched to the curved earth optimal control law such as to satisfy the initial and terminal boundary conditions of the curved earth optimal trajectory.

The above mentioned corrective functions have to be precalculated and stored in the computer. Commonly they are not only used to correct for the assumption of a constant unidirectional instead of a central inverse square distance force field but also to additionally make the on-line computations as simple as possible.

The flat earth guidance law meets the requirements for onboard computer applications. It has been successfully implemented for instance in the Apollo space program[6]. The guidance computations can be performed within a reasonable frame time and the procedure is fairly insensitive to perturbations, though not independent from the mission and changes in the launching site.

4.2 Iterative Procedures

Comments on computer mechanisation:

In this section it will be shown how the most critical drawbacks of iterative explicit guidance procedures can be eliminated. The means to achieve this are the following:

(a) Sizable computation time required for each integration cycle of Euler-Lagrange equations: This drawback can be coped with by:

- proper choice of numerical integration procedure, e.g. Adams,
- normalisation of state variables using the desired nominal orbit parameters as reference conditions,
- linearisation around the injection point,
- approximation of trigonometrical functions of attitude angles, by assuming small variations of these angles in certain flight phases,
- approximation of trigonometrical functions of attitude angles by first order functions.

(b) Variable time requirements for integrating the trajectory between the present state and the desired end state: In order to cope with this drawback the usefullness of integration step control was investigated. Instead of using a varying number of intervals of fixed time length, a fixed number of integration steps of varying time length was chosen.

Three integration steps were sufficient in the following examples:

- Injection of an upper stage of a three stage vehicle into 200 km and 550 km circular orbits,
- injection of the upper stage of a two stage vehicle into an elliptical orbit of 200 km perigee and 36,000 km apogee.

(c) Undetermined number of reoptimization cycles for each updating of attitude commands: Updating of the attitude law during the flight becomes necessary for two reasons:

- Disturbances (external or caused by the system) such as wind, turbulence, start-up and cut-off of engines, staging, jettisoning of fairings etc.

  In order to cope with these disturbances, reoptimization would be required only shortly after the particular event. This type of disturbance occurs only at special instances. It would be sufficient to reoptimize five or ten times during a flight of 500 to 700 seconds.

- perturbations of system parameters from the assumed nominal values. These are more critical with regard to injection accuracy and payload in orbit. Therefore updating of the attitude law in much shorter time intervals is required, e.g. two to five times a second. The most critical vehicle parameters are thrust magnitude, thrust misalignment, mass flow rate and initial mass. It is understood that the convergence of an explicit guidance procedure is improved, i.e. the number of reoptimization cycles is reduced if the vehicle model used in the equations of motion is close to reality. Updating of the most critical system parameters in flight from the inertial measurement data by means of appropriate parameter estimation techniques (Kalman filter) is therefore very useful and in many cases demanded anyway from other considerations like performance optimization and analysis.

Demonstration of iterative explicit guidance:

In order to demonstrate how the iterative explicit guidance procedure works, an example is given in the Appendix. The symbols and the coordinate systems used can be understood from Figure 6 and Equations (1). Equations (2) describe the motion of the vehicle. Equations (3) and (4) describe how rationalized relative coordinates are introduced for the case of injection into a circular orbit. Using this normalisation and substituting the nominal orbit parameters, the equations of motion take the form of Equations (6). Equations (7) are the differential equations of the associated adjoint variables. The optimal thrust program is given in Equations (8).

The optimization procedure is illustrated by Figure 7 for a motion in the orbit plane. Here the integration of the Euler-Lagrange equations was performed using a fast logic controlled analogue program. Sufficient accuracy at injection even with analogue components was achieved by use of rationalized relative coordinates. The iterative updating of the initial conditions of adjoint variables was done digitally. The matrix used for updating the adjoint variables p_i is given in Equation (9). The digital computer was also used to calculate the motion of the centre of gravity of the vehicle.

In the simplified case of Figure 7 (motion in the nominal orbit plane) the three state variables, radial distance R_0 , radial velocity V_r and tangential velocity V_θ describe the system completely. Figure 7 shows the deviations of the state variables ($\Delta R_0, \Delta V_\theta$) from their nominal values in rationalized relative coordinates. The spikes to be noticed arise from fast integrations of the Euler-Lagrange equations in the logic controlled analogue program. The adjoint variables determine the optimal thrust program. The first spike in each group of three integrations represents a test run with estimated initial conditions of adjoint variables. At the end of this test run the injection point achieved is compared with the desired injection conditions. If the injection point is not achieved with the required accuracy, the integration of the Euler-Lagrange equations has to be repeated twice in order to construct the updating matrix (2nd and 3rd spike). The initial conditions of the adjoint variables have to be systematically varied for this purpose. The formulae used for the computation of the partial derivatives of the updating matrix are listed under Equations (11). The use of the runs of Equation (11) for the construction of the approximated updating matrix can be seen from matrix No. 12.

The partial derivatives are constructed from the deviations, which occur in the state variables at injection, resulting from a systematic variation of the initial conditions of adjoint variables. The updating matrix is used to improve the initial conditions of adjoint variables by iteration. The optimization cycle described has to be repeated until the test run (first spike) delivers state variables which meet the desired injection accuracy.

From the initial conditions of adjoint variables thus determined the optimal attitude law to guide the launch vehicle is derived. After a time interval Δt the launch vehicle deviates from the predicted optimal flight path mainly due to perturbations. Therefore reoptimization is required in order to determine a new optimal attitude law. In Figure 7 five optimization cycles are necessary at the beginning of the upper stage flight due to disturbances originating from other stages, separation and start-up of the engines. Afterwards only three optimization cycles are required to cope with the perturbations in vehicle parameters. (Here thrust magnitude, thrust direction and truncation errors.) Reduction of the reoptimization cycles required from 3 to 1 can be obtained by updating the parameters of the vehicle model used in the optimization procedure as was already mentioned.

Integration step control:

In Figure 9 the efficiency of integration step control is demonstrated for the injection of the upper stage of a three stage launch vehicle into a 550 km circular orbit for the nominal case as well as for perturbed cases. In Figure 9.1 the results obtained by integration of the Euler-Lagrange equations with fixed step size of $\Delta t = 5$ sec and by integration step control with 3 steps are compared for the nominal case. No difference neither in attitude law nor in performance (time to go) is to be noticed. In Figures 9.2 and 9.3 perturbations in thrust direction and thrust magnitude were investigated. It is to be noticed, that perturbations in thrust direction mainly influence the attitude law and not so much the performance (total mission time t_g). Deviations in thrust magnitude of course influence the total time of flight but not so much the attitude law. In Figures 9.2 and 9.3 the demanded attitude changes very rapidly towards the end. This phenomenon is avoided by changing weighting factors of injection conditions in the final flight phase (or omitting all position constraints like radial distance).

Simplification of Euler-Lagrange equations:

In Figure 8 approximations used in order to simplify the Euler-Lagrange equations are listed. By these simplifications computation time requirements on the onboard computer were relieved. Figure 8 shows that the simplifications proposed do not lead to any sizable increase in time to go for the nominal case. Figures 10 and 11 show the attitude laws to be expected.

Figure 8; No. 7: $z, \rho \ll 1$

The equations of motion can therefore be linearised around the desired end conditions. Attitude law see Figure 10.1.

Figure 8; No. 8: $z, \rho \ll 1 ; \Delta\psi_2 \ll 1$

An end law is used 40 seconds before injection in order to avoid rapid changes in attitude demands towards the end (see Figure 10.1).

Figure 8; No. 9: $z, \rho \ll 1 ; \Delta\psi_1 \ll 1 ; \Delta\psi_2 \ll 1$

Attitude law see Figure 10.2. No end law was used in order to smooth out the final attitude demands.

Figure 8; No. 10:

Corresponds to the above No. 9 except for the additional assumption that p_2 is constant in the linearised form (see Figure 10.2).

Figure 8; No. 11:

Corresponds to No. 10 except for the use of an end law in order to reduce bandwidth requirements near injection (see Figure 10.2).

Figure 8; No. 12:

In this case the solution of the differential equations of adjoint variables was eliminated by approximating $\cos\psi_1 = K_{11} + K_{12}\tau$; $\sin\psi_1 = K_{21} + K_{22}\tau$.

Instead of varying the initial conditions of adjoint variables, the coefficients of the attitude law (K_{11}, K_{12}, K_{21}, K_{22}) were varied instead in order to meet the injection requirements (see the Appendix, Equations (14); for the updating matrix see Equation (15)).

Even these very severe simplifications do not lead to any considerable loss in performance as can be seen from the time to go. The attitude law is shown in Figure 11.1 where other cases are repeated for comparison.

In Figure 11.2 the predicted "times to go" as a function of flight time for the nonlinear case and this very much simplified case are compared. The differences in time to go are negligible. The fast changes in attitude demands towards the end (Fig.11.1) can again be avoided by using an appropriate end law.

Injection into elliptical orbit:

Rationalized relative coordinates, subsequent linearisation of the equations of motion and integration step control have been applied for guidance of launch vehicles into elliptical orbits as well. The nominal attitude law as obtained from the guidance of the 2nd stage of a two stage launch vehicle into an elliptical orbit of 200 km perigee and 36.000 km apogee is shown in Figure 12. For comparison the attitude law for injecting the same upper stage into a 200 km circular orbit is drawn. For circular orbits and elliptical orbits of the same perigee, similar bandwidth requirements exist, if injection takes place near the perigee. When injecting into other regions of the ellipse, the bandwidth requirements and guidance noise produced near injection are less severe. The following sensitivity investigation to establish the bandwidth requirements on the vehicle for the nominal as well as for perturbed cases are therefore shown for injection into a 200 km circular orbit only.

Interaction between guidance and vehicle bandwidth:

Preliminary guidance investigations are necessary in order to establish the bandwidth requirements particularly on the upper stages of launch vehicles. As an example loss in performance and changes in attitude law as a result of vehicle bandwidth were investigated for the 2nd stage of a two stage launch vehicle. Injection into a 200 km circular orbit was considered. The case with no perturbation and vehicle transfer function unity is referred to as nominal case. In order to perform the investigations, the following perturbations in vehicle parameters were assumed:

(a) in-plane thrust vector misalignment of $\Delta\alpha = \pm 1$ degree,

(b) distance of vehicle centre of gravity from the symmetry axes: $\Delta l = \pm 30$ mm,

(c) error in specific impulse of $\pm 1\%$.

The 2nd stage under attitude control was approximated by a mathematical model of 2nd order:

$$\frac{\theta}{W_\theta} = \frac{1}{T^2 s^2 + 2DTs + 1} .$$

Time constants in the range between $0 \leq T \leq 5$ sec have been considered for $D = 0.7$. Evaluation of the results shows that the most severe loss in payload occurs when thrust misalignment and deviation of centre of gravity act in one direction, resulting in a negative thrust vector misalignment (see Figure 13). In the case mentioned:

$$\Delta\alpha = -1^\circ ; \quad \Delta l = -30 \text{ mm}$$

the following additional payload loss Δm occurred due to the response of the 2nd stage:

$$T = 1 \text{ sec} ; \quad D = 0,7 ; \quad \Delta m = 6,1 \text{ kg}$$

$$T = 5 \text{ sec} ; \quad D = 0,7 ; \quad \Delta m = 65,1 \text{ kg}.$$

In Figure 13 the loss in performance (payload Δm) for different bandwidth of 2nd stage and perturbations in thrust direction has been plotted. Figures 14.1 and 14.2 show the changes in attitude law due to perturbations in thrust direction (positive and negative angles), and vehicle bandwidth. The bandwidth requirements towards the end have to be relieved by modifying the guidance law or introducing appropriate weighting factors on injection conditions for the final flight phase.

5. CONCLUSIONS

Due to the fact that analytical solutions of the nonlinear optimal guidance are not available, for most practical applications approximate numerical solutions are used. The discussion of the procedures commonly implemented has shown that explicit guidance is preferable for its independence of mission and vehicle parameters, provided it is possible to circumvent the characteristic drawbacks associated with it like large computation time required and variable frame time. It has been shown in the paper that even iterative explicit guidance can be applied successfully for on-line optimization by introducing proper means to avoid the above mentioned difficulties. The remedies are:

- proper integration step control
- inflight estimation of vehicle parameters

- use of rationalized relative coordinates referred to the injection point
- subsequent linearisation of the Euler-Lagrange equations
- piecewise approximation of trigonometrical functions of the attitude angles by first order equations, the coefficients of which are iterated in order to meet the desired injection conditions.

ACKNOWLEDGEMENT

The authors are indepted to Col.K.P.Davies of ELDO who promoted systematic studies of the ELDO launch vehicles in the closed guidance loop and to R.Cosaert of ELDO for fruitful discussions.

REFERENCES

1. Vigneron, M. — *La Station de Guidage ELDO.* Revue MBLE/Vol.IX, No.1 (10e année).

2. Polack M. — *Lanceur de Satellite ELDO – Calcul de Guidage et de Localisation.* SEREB report.

3. Albery, J.W. — *Assessment of the Performance of the ELDO First Program Attitude Reference Unit in an Inertial Role.* Royal Aircraft Establishment, Technical Report No.66314.

4. Gage, M.J.W. — *FE-Guidance and its Application to the ELDO Launcher Vehicle.* Royal Aircraft Establishment, Technical Report No.68201.

5. Mattin, R.B. — *Inertial Guidance for ELDO Launcher.* ELDO TM 109 (E 28/10).

6. Haeusserman, W. — Survey paper, 3rd IFAC-Symposium on Automatic Control in Space, Toulouse, March 2nd - 6th, 1970.

7. Riley, A. — *The Inertial Guidance System for the EUROPA Satellite Launcher.* 19th Congress of the International Astronautical Federation, New York, 1968.

8. Cosaert, R.
Gottzein, E. — *Closed Loop Simulation of Optimum Guidance of Space Vehicles by Means of Analogue Computer Techniques.* ELDO Technical Review 1966 1, pp.191-231.

9. Cosaert, R. — *On the Optimum Guidance Law for Both Thrust-Coast-Thrust and Continuous Thrust Transfer to Orbit of a Space Vehicle.* Part 1 and 2, Report No.37, pp.5-13,14.

10. Cosaert, R.
Gottzein, E. — *Regelung von Trägerraketen.* Lecture VDI Deutscher Ingenieurtag Braunschweig 1969.

APPENDIX

Definitions and Coordinate System

Unity vectors n_1, n_2, n_3

$$\begin{aligned} w_r &= V_r n_1 \\ w_\theta &= V_\theta n_2 \\ w_z &= V_z n_3 \end{aligned} \qquad w = w_r + w_\theta + w_z = V_r n_1 + V_\theta n_2 + V_z n_3$$

$$\begin{aligned} \mathcal{R} &= R n_1 \\ \mathcal{Z} &= z n_3 \\ \Omega &= \omega n_3 \end{aligned} \qquad \varrho = \mathcal{R} + \mathcal{Z} = R n_1 + z n_3 \qquad (1)$$

$$g = -\frac{Mk}{|\varrho|^3}\varrho = -\frac{Mk}{|\varrho|^3}(R n_1 + z n_3) \qquad - \quad |g| = \frac{Mk}{|\varrho|^2}$$

$$\gamma = (S\cos\varphi_1\cos\varphi_2)n_1 + (S\cos\varphi_2\sin\varphi_1)n_2 + S\sin\varphi_2 \cdot n_3$$

$$|\gamma| = S$$

Equations of Motion

$$\frac{d\varrho}{dt} = w = V_r n_1 + V_\theta n_2 + V_z n_3$$

$$\frac{dw}{dt} = \frac{d^2\varrho}{dt^2} = \frac{\partial w}{\partial t} + [\Omega w] = \frac{\delta \breve{R}}{M}$$

substituting: $\quad M = M(0) - \dot{m}t$

$$|\varrho|^3 = (R^2 + z^2)^{3/2}$$

$$\omega = V_\theta / R$$

gives:

(2)

$$\frac{dR}{dt} = V_r$$

$$\frac{dV_r}{dt} = \frac{V_\theta^2}{R} - \frac{kR}{(R^2+z^2)^{3/2}} + \frac{S}{M(0)-\dot{m}t}\cos\varphi_1\cos\varphi_2$$

$$\frac{dV_\theta}{dt} = -\frac{V_r V_\theta}{R} + \frac{S}{M(0)-\dot{m}t}\sin\varphi_1\cos\varphi_2$$

$$\frac{dz}{dt} = V_z$$

$$\frac{dV_z}{dt} = -\frac{kz}{(R^2+z^2)^{3/2}} + \frac{S}{M(0)-\dot{m}t}\sin\varphi_2$$

$$\frac{d\theta}{dt} = \frac{V_\theta}{R}$$

Nominal Orbit Parameters

$$M\frac{V_o^2}{R_o} = M\frac{k}{R_o^2}$$

$$V_o = \omega_o R_o$$

$$V_o\omega_o = \frac{V_o^2}{R_o} = \frac{k}{R_o^2} = g_o \qquad (3)$$

$$\tau = \omega_o t$$

Definition of Normalized Quantities

$$\varrho = \frac{R - R_o}{R_o} \qquad R = R_o(\varrho + 1)$$

$$v_\vartheta = \frac{V_\vartheta - V_o}{V_o} \qquad V_\vartheta = V_o(v_\vartheta + 1)$$

$$v_r = \frac{V_r}{V_o} \qquad V_r = V_o v_r \qquad (4)$$

$$v_z = \frac{V_z}{V_o} \qquad V_z = V_o v_z$$

$$z = \frac{Z}{R_o} \qquad Z = R_o z$$

$$\vartheta = \Theta - \tau \qquad \Theta = \vartheta + \tau$$

Definition of H and p_i

Hamiltonian H $$H = \sum_{i=1}^{5} p_i f_i \,.$$

where $f_i = \frac{dx_i}{d\tau}$ therefore $\frac{dx_i}{d\tau} = \frac{\partial H}{\partial p_i}$ (5)

Adjoint variables p_i $$\frac{dp_i}{d\tau} = -\frac{\partial H}{\partial x_i}$$

Normalized Equations of Motion

$$\frac{d\varrho}{d\tau} = v_r$$

$$\frac{dv_r}{d\tau} = \frac{(v_\vartheta+1)^2}{(\varrho+1)} - \frac{(\varrho+1)}{[(\varrho+1)^2+z^2]^{3/2}} + \frac{1}{g_o}\cdot\frac{S}{M(o)-\frac{\dot m}{\omega_o}\tau}\cos\varphi_1\cos\varphi_2$$

$$\frac{dv_\vartheta}{d\tau} = -\frac{v_\vartheta+1}{\varrho+1}v_r + \frac{1}{g_o}\cdot\frac{S}{M(o)-\frac{\dot m}{\omega_o}\tau}\sin\varphi_1\cos\varphi_2$$

$$\frac{dz}{d\tau} = v_z \qquad (6)$$

$$\frac{dv_z}{d\tau} = -\frac{z}{[(\varrho+1)^2+z^2]^{3/2}} + \frac{1}{g_o}\cdot\frac{S}{M(o)-\frac{\dot m}{\omega_o}\tau}\sin\varphi_2$$

$$\frac{d\vartheta}{d\tau} = \frac{v_\vartheta+1}{\varrho+1} - 1$$

$$b = \frac{1}{g_o}\frac{S}{M(o)-\frac{\dot m}{\omega_o}\tau}$$

Adjoint Variables

$$
\begin{aligned}
\frac{dp_1}{d\tau} &= -p_2\frac{v_\vartheta+1}{(\varrho+1)^2}v_r + p_3\frac{(v_\vartheta+1)^2}{(\varrho+1)^2} + p_3\frac{z^2-2(\varrho+1)^2}{[(\varrho+1)^2+z^2]^{5/2}} - p_5\frac{3z(\varrho+1)}{[(\varrho+1)^2+z^2]^{5/2}}\\
\frac{dp_2}{d\tau} &= p_2\frac{v_r}{\varrho+1} - 2p_3\frac{v_\vartheta+1}{\varrho+1}\\
\frac{dp_3}{d\tau} &= -p_1 + p_2\frac{v_\vartheta+1}{\varrho+1}\\
\frac{dp_4}{d\tau} &= -3p_3\frac{z(\varrho+1)}{[(\varrho+1)^2+z^2]^{5/2}} - p_5\frac{(\varrho+1)^2-2z^2}{[(\varrho+1)^2+z^2]^{5/2}}\\
\frac{dp_5}{d\tau} &= -p_4
\end{aligned}
\tag{7}
$$

Optimal Thrust Programme

$$
\begin{aligned}
\sin\Psi_1 &= \frac{p_2}{\sqrt{p_2^2+p_3^2}} & \sin\Psi_2 &= \frac{p_5}{\sqrt{p_2^2+p_3^2+p_5^2}}\\
\cos\Psi_1 &= \frac{p_3}{\sqrt{p_2^2+p_3^2}} & \cos\Psi_2 &= \frac{\sqrt{p_2^2+p_3^2}}{\sqrt{p_2^2+p_3^2+p_5^2}}
\end{aligned}
\tag{8}
$$

Explicit Guidance

Matrix for up-dating adjoint variables p_i

$$
\begin{bmatrix} d\varrho \\ dv_\vartheta \\ dv_r \\ dz \\ dv_z \end{bmatrix} =
\begin{bmatrix}
\frac{\partial\varrho}{\partial p_1} & \frac{\partial\varrho}{\partial p_3} & \frac{\partial\varrho}{\partial p_4} & \frac{\partial\varrho}{\partial p_5} & \frac{\partial\varrho}{\partial\tau}\\
\frac{\partial v_\vartheta}{\partial p_1} & \frac{\partial v_\vartheta}{\partial p_3} & \frac{\partial v_\vartheta}{\partial p_4} & \frac{\partial v_\vartheta}{\partial p_5} & \frac{\partial v_\vartheta}{\partial\tau}\\
\frac{\partial v_r}{\partial p_1} & \frac{\partial v_r}{\partial p_3} & \frac{\partial v_r}{\partial p_4} & \frac{\partial v_r}{\partial p_5} & \frac{\partial v_r}{\partial\tau}\\
\frac{\partial z}{\partial p_1} & \frac{\partial z}{\partial p_3} & \frac{\partial z}{\partial p_4} & \frac{\partial z}{\partial p_5} & \frac{\partial z}{\partial\tau}\\
\frac{\partial v_z}{\partial p_1} & \frac{\partial v_z}{\partial p_3} & \frac{\partial v_z}{\partial p_4} & \frac{\partial v_z}{\partial p_5} & \frac{\partial v_z}{\partial\tau}
\end{bmatrix} \cdot
\begin{bmatrix} dp_1 \\ dp_3 \\ dp_4 \\ dp_5 \\ d\tau \end{bmatrix}
\tag{9}
$$

Implicit Guidance

Matrix for up-dating adjoint variables p_i

$$
\begin{bmatrix} dp_1 \\ dp_3 \\ dp_4 \\ dp_5 \\ d\tau \end{bmatrix} =
\begin{bmatrix}
\frac{\partial p_1}{\partial\varrho} & \frac{\partial p_1}{\partial v_\vartheta} & \frac{\partial p_1}{\partial v_r} & \frac{\partial p_1}{\partial z} & \frac{\partial p_1}{\partial v_z}\\
\frac{\partial p_3}{\partial\varrho} & \frac{\partial p_3}{\partial v_\vartheta} & \frac{\partial p_3}{\partial v_r} & \frac{\partial p_3}{\partial z} & \frac{\partial p_3}{\partial v_z}\\
\frac{\partial p_4}{\partial\varrho} & \frac{\partial p_4}{\partial v_\vartheta} & \frac{\partial p_4}{\partial v_r} & \frac{\partial p_4}{\partial z} & \frac{\partial p_4}{\partial v_z}\\
\frac{\partial p_5}{\partial\varrho} & \frac{\partial p_5}{\partial v_\vartheta} & \frac{\partial p_5}{\partial v_r} & \frac{\partial p_5}{\partial z} & \frac{\partial p_5}{\partial v_z}\\
\frac{\partial\tau}{\partial\varrho} & \frac{\partial\tau}{\partial v_\vartheta} & \frac{\partial\tau}{\partial v_r} & \frac{\partial\tau}{\partial z} & \frac{\partial\tau}{\partial v_z}
\end{bmatrix} \cdot
\begin{bmatrix} d\varrho \\ dv_\vartheta \\ dv_r \\ dz \\ dv_z \end{bmatrix}
\tag{10}
$$

Explicit Guidance

Computation of partial derivatives

test run:

$$\vec{p} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{bmatrix} \quad \therefore \quad \overset{(0)}{\vec{X}}(\tau = 0) = \begin{bmatrix} \overset{0}{\varrho} \\ \overset{0}{v}_\vartheta \\ \overset{0}{v}_r \\ \overset{0}{Z} \\ \overset{0}{v}_z \end{bmatrix}$$

variation of p_1:

$$\vec{p} = \begin{bmatrix} p_1 + \Delta p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{bmatrix} \quad \therefore \quad \overset{(1)}{\vec{X}}(\tau = 0) = \begin{bmatrix} \overset{1}{\varrho} \\ \overset{1}{v}_\vartheta \\ \overset{1}{v}_r \\ \overset{1}{Z} \\ \overset{1}{v}_z \end{bmatrix}$$

variation of p_3:

$$\vec{p} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 + \Delta p_3 \\ p_4 \\ p_5 \end{bmatrix} \quad \therefore \quad \overset{2}{\vec{X}}(\tau = 0) = \begin{bmatrix} \overset{2}{\varrho} \\ \overset{2}{v}_\vartheta \\ \overset{2}{v}_r \\ \overset{2}{Z} \\ \overset{3}{v}_z \end{bmatrix}$$

(11)

variation of p_4:

$$\vec{p} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 + \Delta p_4 \\ p_5 \end{bmatrix} \quad \therefore \quad \overset{3}{\vec{X}}(\tau = 0) = \begin{bmatrix} \overset{3}{\varrho} \\ \overset{3}{v}_\vartheta \\ \overset{3}{v}_r \\ \overset{3}{Z} \\ \overset{3}{v}_z \end{bmatrix}$$

variation of p_5:

$$\vec{p} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 + \Delta p_5 \end{bmatrix} \quad \therefore \quad \overset{4}{\vec{X}}(\tau = 0) = \begin{bmatrix} \overset{4}{\varrho} \\ \overset{4}{v}_\vartheta \\ \overset{4}{v}_r \\ \overset{4}{Z} \\ \overset{4}{v}_z \end{bmatrix}$$

desired end condition (injection):

$$\begin{bmatrix} \overset{e}{\varrho} \\ \overset{e}{v}_\vartheta \\ \overset{e}{v}_r \\ \overset{e}{Z} \\ \overset{e}{v}_z \end{bmatrix} = \overset{e}{\vec{X}}(\tau = 0) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Explicit Guidance

Approximated up-dating matrix

$$\begin{bmatrix} {}^{e}\varrho - {}^{o}\varrho \\ {}^{e}v_\vartheta - {}^{o}v_\vartheta \\ {}^{e}v_r - {}^{o}v_r \\ {}^{e}z - {}^{o}z \\ {}^{e}v_z - {}^{o}v_z \end{bmatrix} = \begin{bmatrix} \frac{{}^{1}\varrho - {}^{o}\varrho}{\Delta p_1} & \frac{{}^{3}\varrho - {}^{o}\varrho}{\Delta p_3} & \frac{{}^{4}\varrho - {}^{o}\varrho}{\Delta p_4} & \frac{{}^{5}\varrho - {}^{o}\varrho}{\Delta p_5} & f_1(\tau=o) \\ \frac{{}^{1}v_\vartheta - {}^{o}v_\vartheta}{\Delta p_1} & \frac{{}^{3}v_\vartheta - {}^{o}v_\vartheta}{\Delta p_3} & \frac{{}^{4}v_\vartheta - {}^{o}v_\vartheta}{\Delta p_4} & \frac{{}^{5}v_\vartheta - {}^{o}v_\vartheta}{\Delta p_5} & f_2(\tau=o) \\ \frac{{}^{1}v_r - {}^{o}v_r}{\Delta p_1} & \frac{{}^{3}v_r - {}^{o}v_r}{\Delta p_3} & \frac{{}^{4}v_r - {}^{o}v_r}{\Delta p_4} & \frac{{}^{5}v_r - {}^{o}v_r}{\Delta p_5} & f_3(\tau=o) \\ \frac{{}^{1}z - {}^{o}z}{\Delta p_1} & \frac{{}^{3}z - {}^{o}z}{\Delta p_3} & \frac{{}^{4}z - {}^{o}z}{\Delta p_4} & \frac{{}^{5}z - {}^{o}z}{\Delta p_5} & f_4(\tau=o) \\ \frac{{}^{1}v_z - {}^{o}v_z}{\Delta p_1} & \frac{{}^{3}v_z - {}^{o}v_z}{\Delta p_3} & \frac{{}^{4}v_z - {}^{o}v_z}{\Delta p_4} & \frac{{}^{5}v_z - {}^{o}v_z}{\Delta p_5} & f_5(\tau=o) \end{bmatrix} \cdot \begin{bmatrix} \delta p_1 \\ \delta p_3 \\ \delta p_4 \\ \delta p_5 \\ \delta\tau \end{bmatrix} \tag{12}$$

Iterative Procedure Applied to Coefficients of Approximation Attitude Law

Approximation of optimal thrust programme

$$\frac{p_3}{p_2} \doteq \cos\Psi_1 \doteq K_{11} + K_{12}\tau$$

$$\frac{p_5}{p_2} \doteq \sin\Psi_2 \doteq K_{21} + K_{22}\tau \tag{13}$$

Equations of motion

$$\frac{d\varrho}{d\tau} = v_r$$

$$\frac{dv_\vartheta}{d\tau} = -v_r + b$$

$$\frac{dv_r}{d\tau} = \varrho + 2v_\vartheta + b(K_{11} + K_{12}\tau) \tag{14}$$

$$\frac{dz}{d\tau} = v_z$$

$$\frac{dv_z}{d\tau} = -z + b(K_{21} + K_{22}\tau)$$

Up-dating matrix

$$\begin{bmatrix} \Delta\varrho \\ \Delta v_\vartheta \\ \Delta v_r \\ \Delta z \\ \Delta v_z \end{bmatrix} = \begin{bmatrix} \frac{\partial\varrho}{\partial K_{11}} & \frac{\partial\varrho}{\partial K_{12}} & o & o & \frac{\partial\varrho}{\partial\tau} \\ \frac{\partial v_\vartheta}{\partial K_{11}} & \frac{\partial v_\vartheta}{\partial K_{12}} & o & o & \frac{\partial v_\vartheta}{\partial\tau} \\ \frac{\partial v_r}{\partial K_{11}} & \frac{\partial v_r}{\partial K_{12}} & o & o & \frac{\partial v_r}{\partial\tau} \\ o & o & \frac{\partial z}{\partial K_{21}} & \frac{\partial z}{\partial K_{22}} & \frac{\partial z}{\partial\tau} \\ o & o & \frac{\partial v_z}{\partial K_{21}} & \frac{\partial v_z}{\partial K_{22}} & \frac{\partial v_z}{\partial\tau} \end{bmatrix} \cdot \begin{bmatrix} \Delta K_{11} \\ \Delta K_{12} \\ \Delta K_{21} \\ \Delta K_{22} \\ \Delta\tau \end{bmatrix} \tag{15}$$

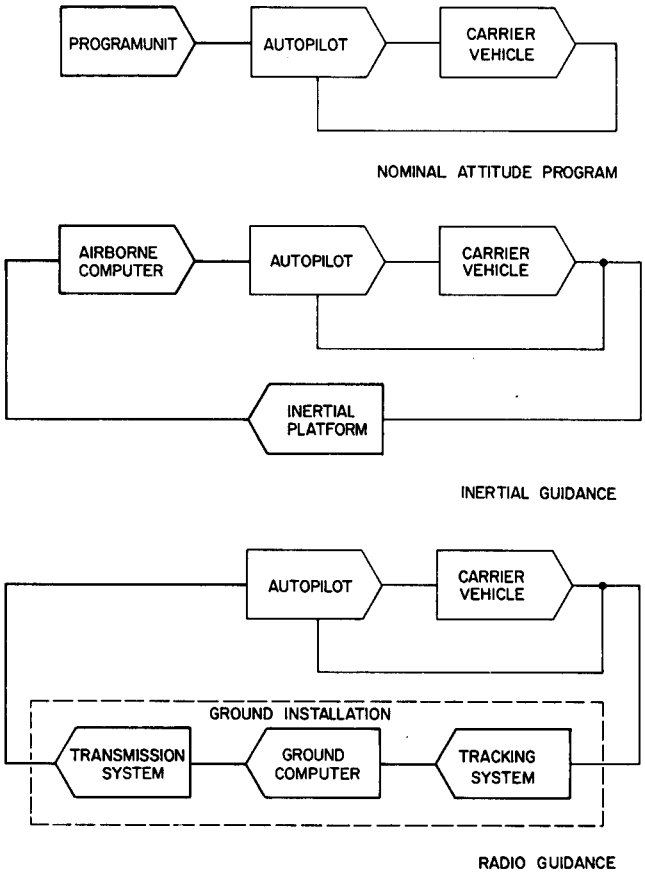


Fig.1 Methods of guiding a launch vehicle

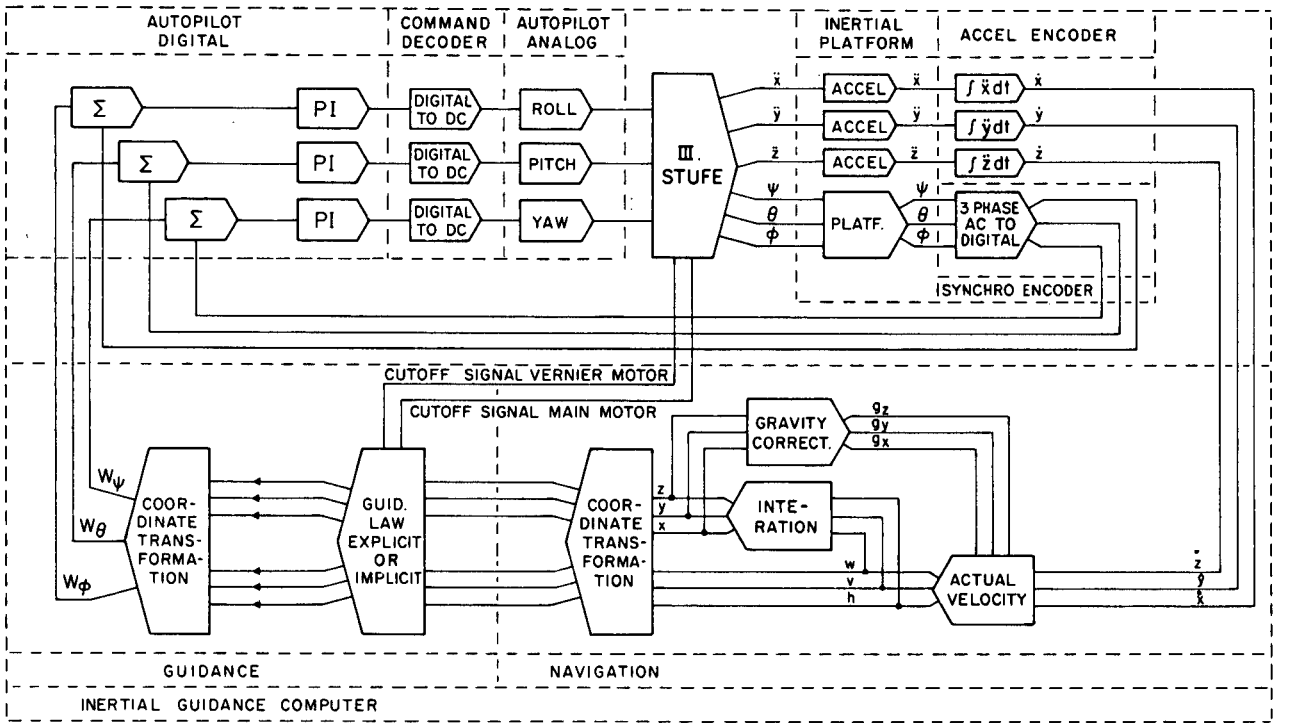


Fig.2 Block diagram of the attitude control and guidance system for inertial guidance

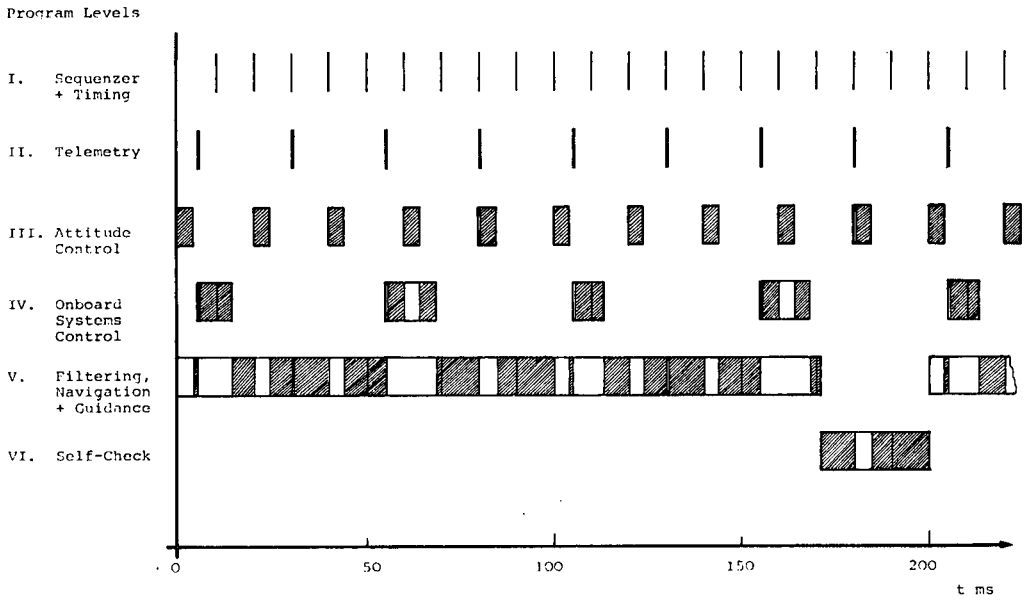


Fig.3 Typical example of program timing

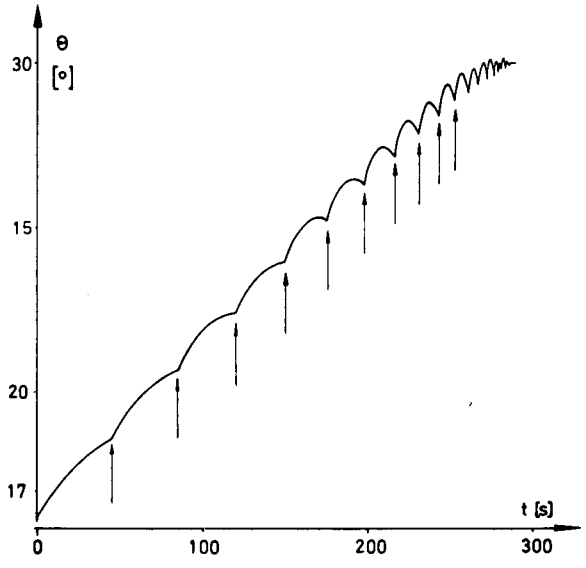


Fig.4 Attitude law for an implicit guidance procedure

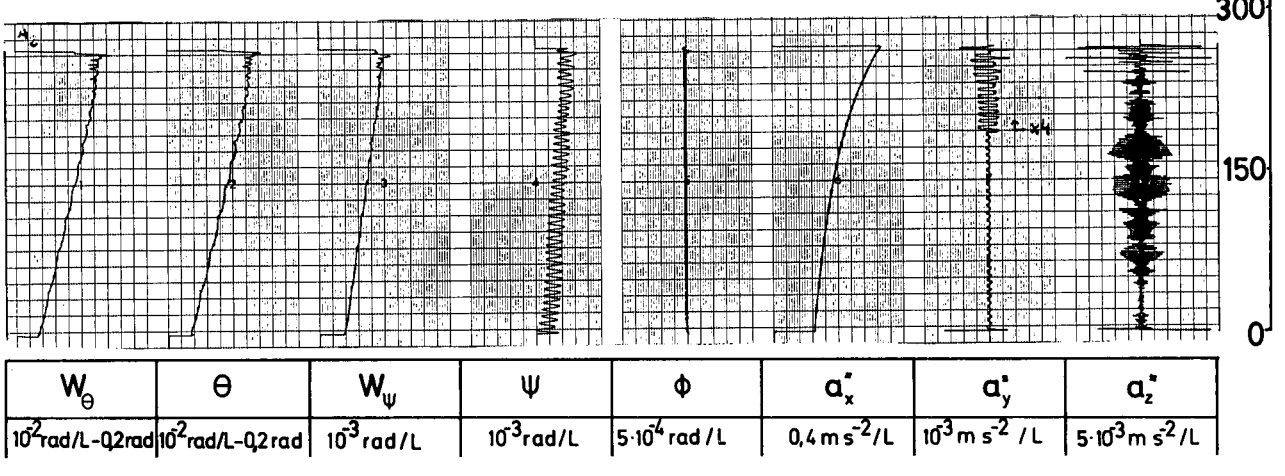


Fig.5 Vehicle behaviour under implicit guidance

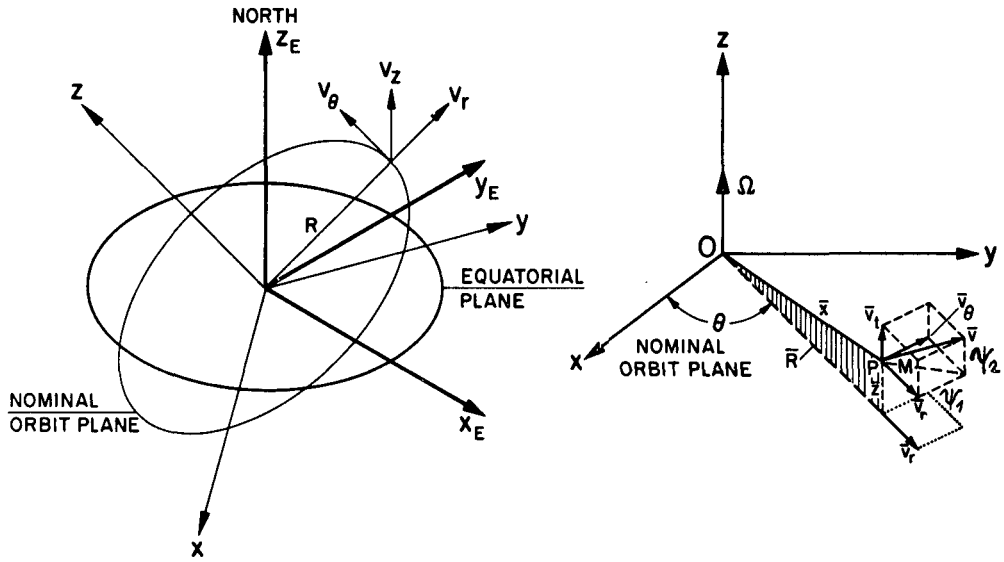


Fig.6 Inertial and nominal orbit plane coordinates

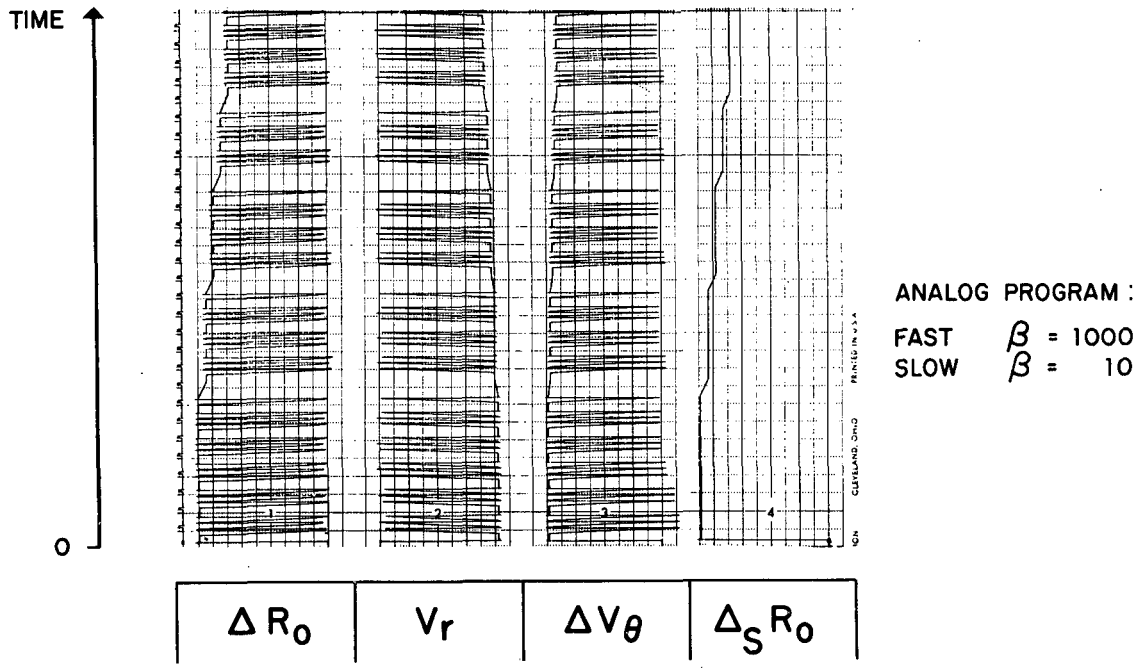


Fig.7 Iterative guidance procedures

No	z	ρ	$\Delta\psi_1$	$\Delta\psi_2$	p_2	integration step control	time to go ttg	perturbations	End Law	Fig	Case
1						no, $\Delta t = 5$ sec	279,31 sec	no	no		1a
2						yes, 3 steps	279,31 sec	no	no		1c
3						yes, 3 steps	279,66 sec	yes, $\Delta\psi_1 = +4^o$	no		2
4						yes, 3 steps	279,72 sec	yes, $\Delta\psi_1 = -4^o$	no		3
5						yes, 3 steps	274,67 sec	yes, $\Delta S = +3\,\%$	no		4
6						yes, 3 steps	284,36 sec	yes, $\Delta S = -3\,\%$	no		5
7	<1	<1				yes, 3 steps	279,37 sec	no	no		6a
8	<1	<1		<1		yes, 3 steps	279,37 sec	no	yes		7
9	<1	<1	<1	<1		yes, 3 steps	279,56 sec	no	no		8
10	<1	<1	<1	<1	const	yes, 3 steps	279,52 sec	no	no		9a
11	<1	<1	<1	<1	const	yes, 3 steps	279,52 sec	no	yes		9b
12	<1	<1	$\cos\psi_1 = K_{11} + K_{12}\,\tau$ $\sin\psi_2 = K_{21} + K_{22}\,\tau$			yes, 3 steps	279,49 sec	no	no		10c

Fig. 8 Approximations used in the iterative procedure

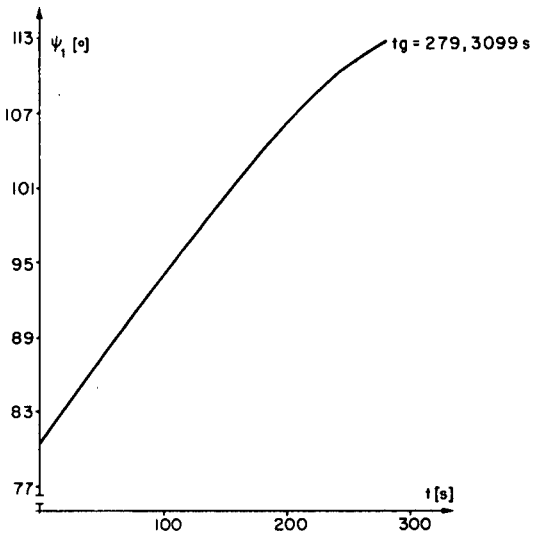


Fig. 9.1 Comparison between attitude laws and total "time to go" tg

case 1(a): fixed step size Δt = 5 sec

case 1(c): integration step control: 3 steps

no difference to be noticed

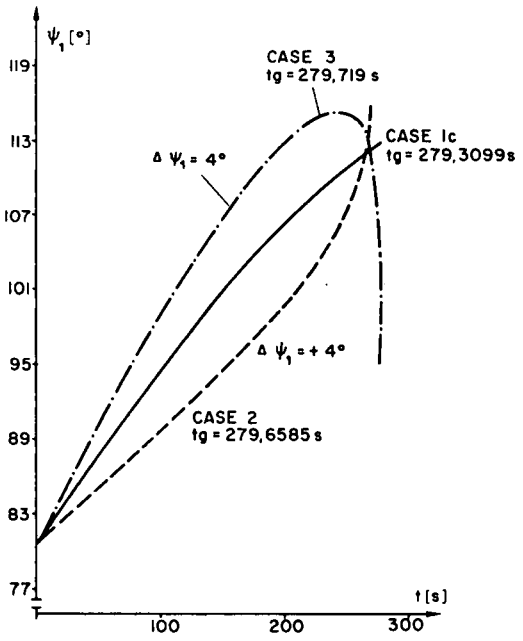


Fig. 9.2 Influence of angular perturbations

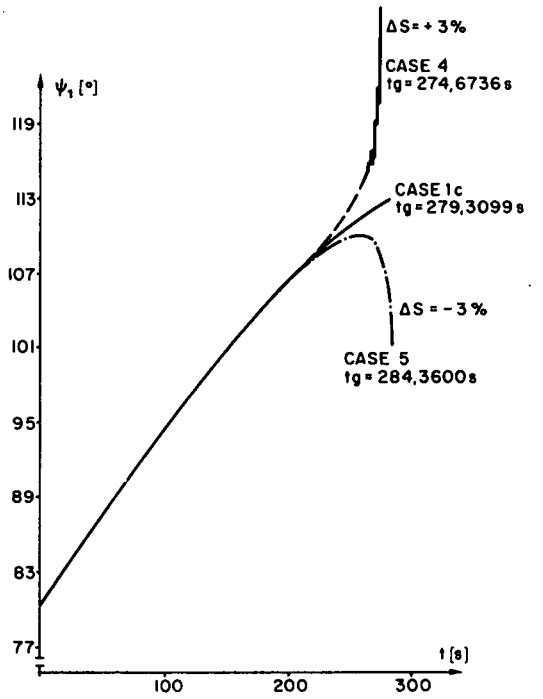


Fig. 9.3 Influence of perturbations in thrust magnitude Δs

Fig. 9 Attitude laws for an explicit, iterative guidance procedure, Euler-Lagrange equations nonlinear, integration step control: 3 steps

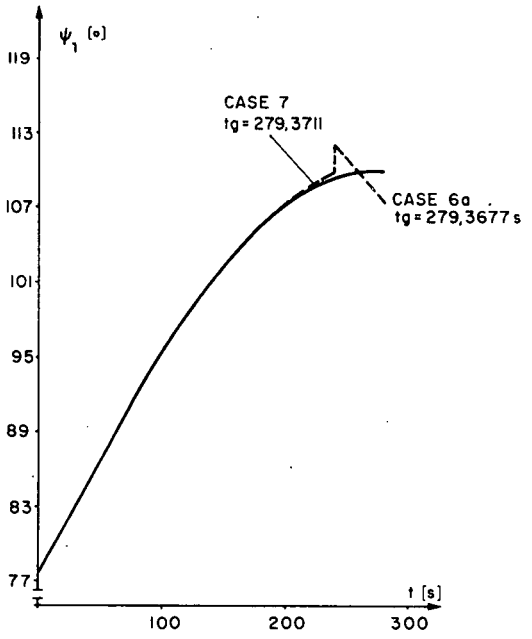


Fig. 10.1 Comparison of attitude laws and total "time to go" for

case 6(a): $z < 1$, $\rho < 1$

case 7: $z < 1$, $\rho < 1$, $\Delta\psi_2 \ll 1$

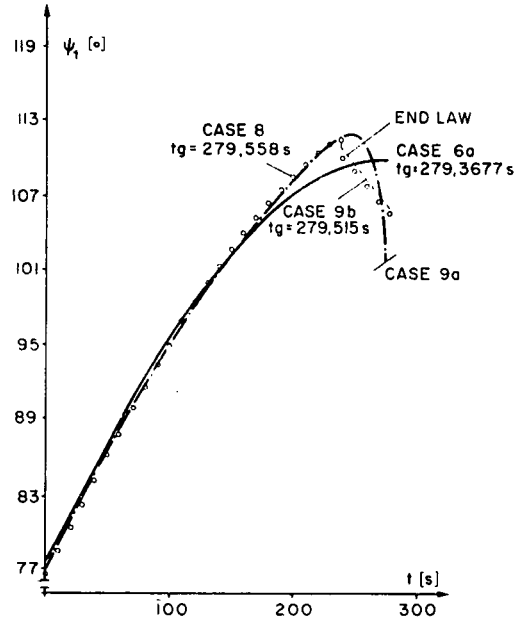


Fig. 10.2 Comparison of attitude laws and total "time to go" for

case 6(a): see Fig. 10.1

case 8: z, $\rho < 1$; $\Delta\psi_1$, $\Delta\psi_2 \ll 1$

case 9: z, $\rho < 1$; $\Delta\psi_1$, $\Delta\psi_2 \ll 1$, $p_2 = \text{const.}$

Fig. 10 Attitude laws for an approximated iterative guidance procedure, Euler-Lagrange equations linearised around the end conditions, integration step control: 3 steps

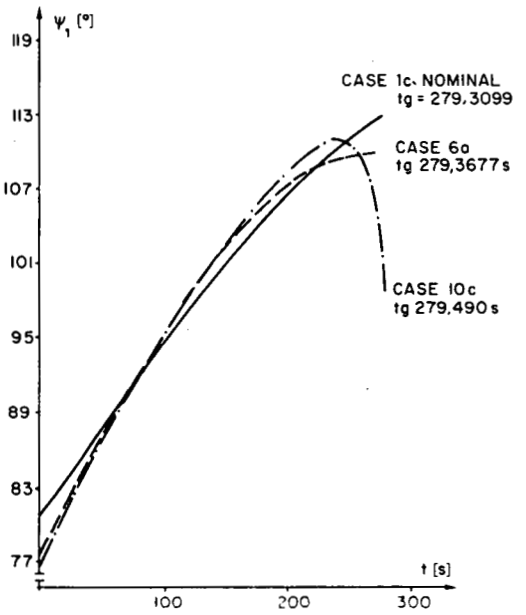


Fig. 11.1 Comparison of attitude laws and total "time to go" for

case 6(a): see Fig. 10.1
case 1(c): see Fig. 9.1
case 10(c): z , $\rho < 1$
$\cos\psi_1$: $K_{11} + K_{12}\tau$
$\sin\psi_2$: $K_{21} + K_{22}\tau$

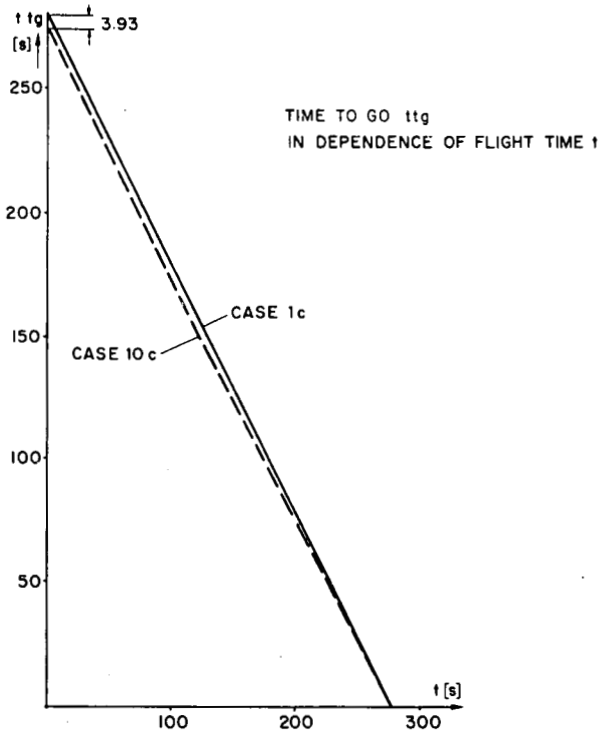


Fig. 11.2 Comparison of predicted "times to go ttg"

case 1(c): see Fig. 9.1
case 10(c): see Fig. 11.1

Fig. 11 Attitude laws for an approximated iterative guidance procedure, Euler-Lagrange equations linearised around the end conditions, integration step control: 3 steps

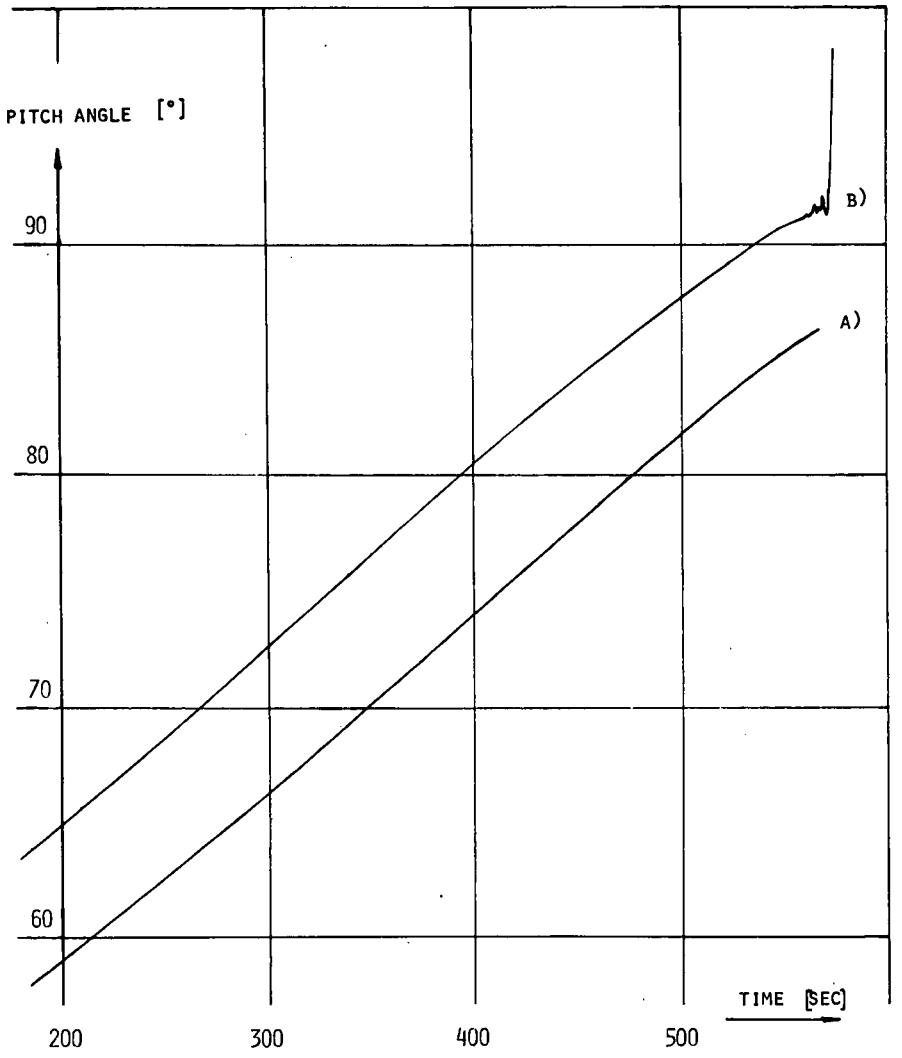


Fig. 12 Nominal attitude law for:

A) 200 km circular orbit

B) 200/36000 km elliptical orbit

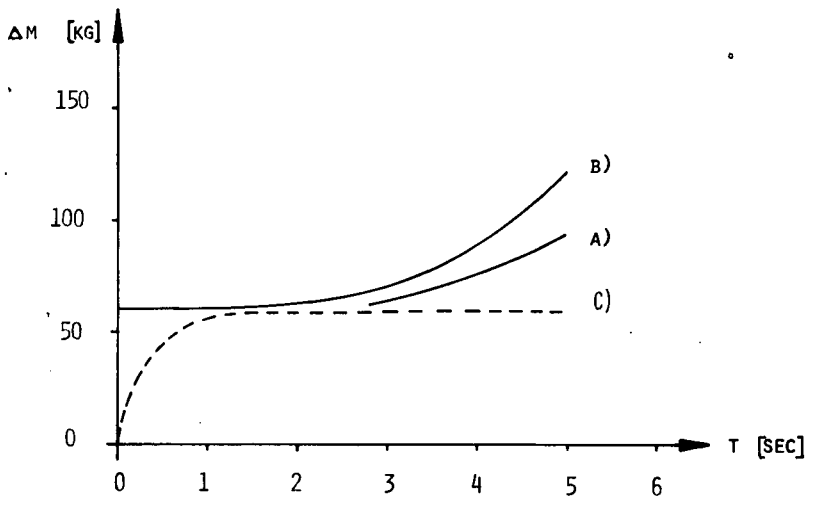


Fig. 13 Payload loss for different bandwidth of 2nd stage and for perturbations in thrust direction

A) $\Delta\alpha = +1^{o}$; $\Delta l = 30$ mm

B) $\Delta\alpha = -1^{o}$; $\Delta l = -30$ mm

C) Perturbations

2nd order vehicle model: T-variable; D = 0.7

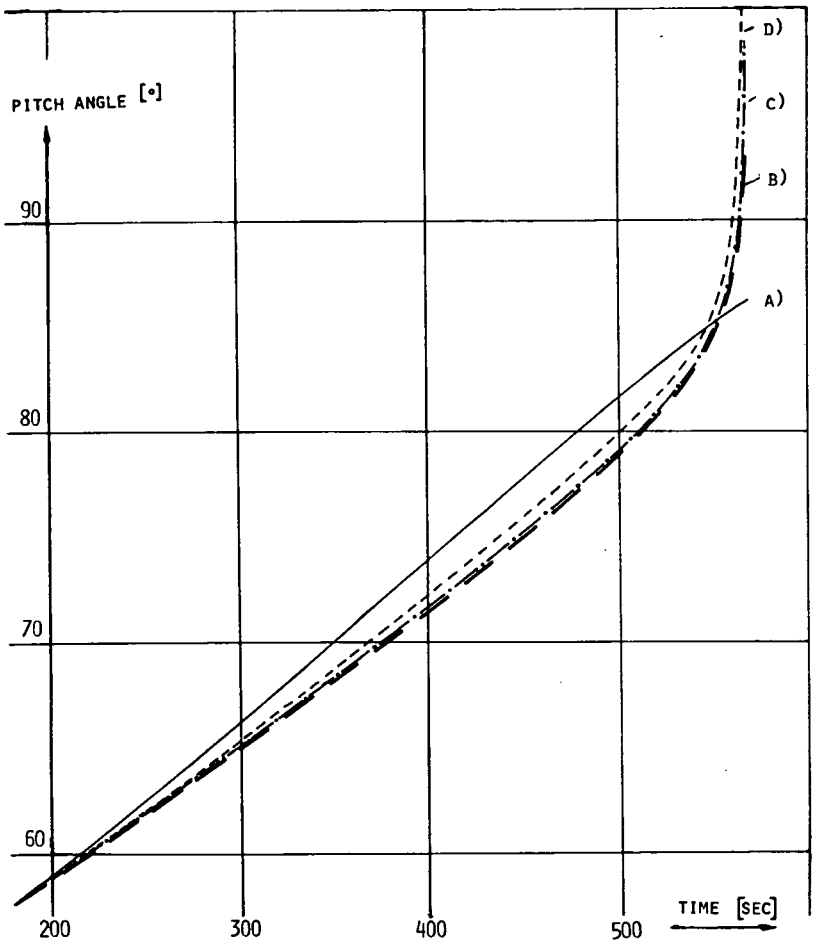


Fig. 14.1 A) Nominal
2nd order vehicle model:
B) $\Delta\alpha = +1^{o}$; $\Delta l = +30$ mm ; T = 0 ; D = 0
C) $\Delta\alpha = +1^{o}$; $\Delta l = +30$ mm ; T = 1 sec ; D = 0.7
D) $\Delta\alpha = +1^{o}$; $\Delta l = +30$ mm ; T = 5 sec ; D = 0.7

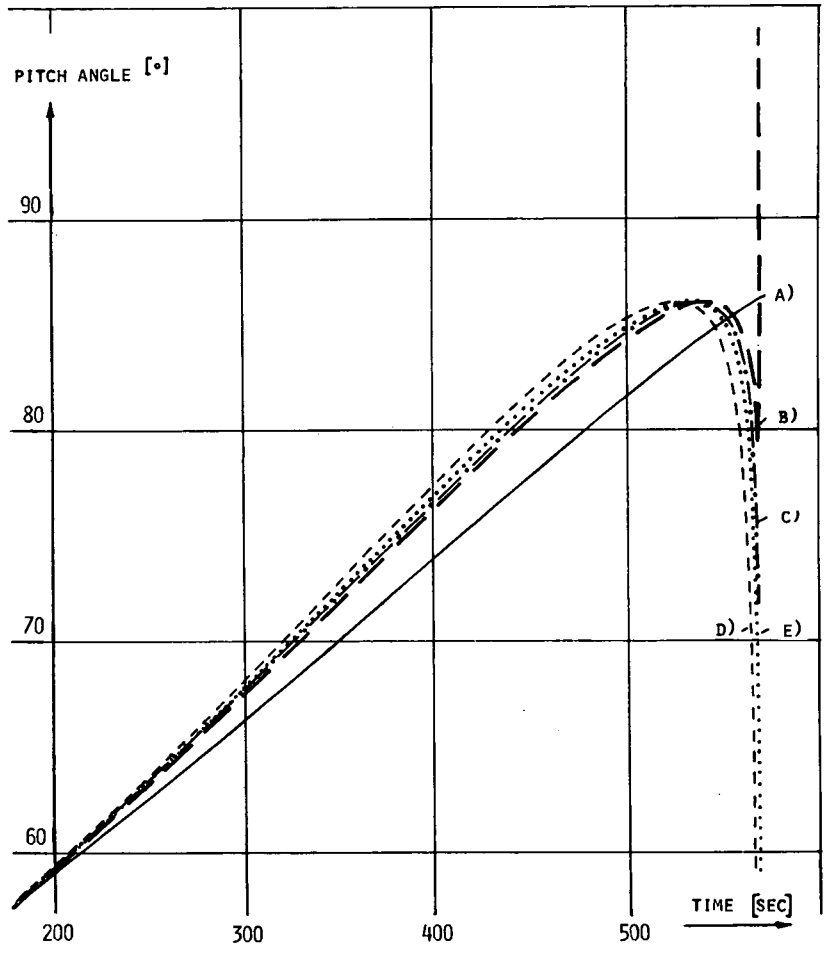


Fig. 14.2 A) Nominal
2nd order vehicle model:
B) $\Delta\alpha = -1^{o}$; $\Delta l = -30$ mm ; T = 0 ; D = 0
C) $\Delta\alpha = -1^{o}$; $\Delta l = -30$ mm ; T = 1 sec ; D = 0.7
D) $\Delta\alpha = -1^{o}$; $\Delta l = -30$ mm ; T = 5 sec ; D = 0.7
E) $\Delta\alpha = -1^{o}$; $\Delta l = -30$ mm ; T = 2.3 sec ; D = 0.75

Fig. 14 Changes in optimal attitude law due to perturbations and vehicle bandwidth for circular orbit

REAL-TIME DATA PROCESSING AND ORBIT DETERMINATION ON THE APOLLO TRACKING SHIPS

Dr. Frederick C. Johnson
Boeing Scientific Research Laboratories
P.O. Box 3981
Seattle, Washington 98124

SUMMARY

The Apollo Instrumentation Ships (AIS) are used during an Apollo lunar mission to provide radar and telemetry coverage during critical mission events that generally cannot be seen by the land-based portion of the Apollo tracking network. In this paper we will discuss the tracking and orbit determination functions of these ships and the instrumentation hardware and real-time software that support these functions. The general philosophy and design concepts of the real-time acquisition and tracking program in use on the AIS is discussed as well as the procedures that were used to minimize the problems associated with the development and test phases of the program.

The design and programming effort covered in this paper was performed by the author and his colleagues while the author was associated with DBA Systems, Inc., Lanham, Maryland. This effort was supported by the Goddard Space Flight Center, Greenbelt, Maryland, under contract NAS 5-9922.

I. INTRODUCTION

During an Apollo lunar mission the insertion of the spacecraft into earth orbit and subsequent injection into translunar orbit generally occur out of sight of the radar coverage provided by the land-based portion of the Apollo tracking network. The Apollo Instrumentation Ships (the USNS Vanguard, USNS Mercury, and USNS Redstone) were designed and implemented to provide a means for obtaining supplemental radar, and communication coverage during these critical events. A photograph of the USNS Redstone is given in Appendix I.

In order to best provide flexible and timely support for an Apollo lunar mission, the radar data is processed onboard the AIS and real-time orbit determinations obtained. The results are then transmitted to the Apollo Mission Control Center at the Manned Spacecraft Center in Houston, Texas, where the final GO/NO-GO decisions for the mission are made. The hardware systems on the AIS which support the radar tracking and orbit determination functions of the AIS are discussed in Section II. Section III covers the requirements that the AIS must satisfy during an Apollo mission. Section IV gives an outline of the real-time acquisition and tracking program used on the AIS. The development and verification of this program are covered in Section V, and concluding remarks are given in Section VI.

II. HARDWARE SYSTEMS ON THE AIS

The following systems are the major hardware components which pertain to the tracking and orbit determination functions of the AIS.

A. Tracking System. Two independent radar systems are carried on the AIS, a C-band system and a Unified S-band system. Both systems measure range, elevation and bearing of the target relative to the ship. The data rate of these systems is 40 points per second (pps).

B. Ships Position and Attitude Measuring System (SPAMS). The primary source of ship position, attitude and velocity data is an inertial navigational system. This system measures the latitude and longitude of the ship, the pitch, heading and roll of the ship, and the north and east components of the velocity of the ship. In case of failure of the inertial navigational system, the AIS also carry a gyro-compass as a backup system for attitude data and an electromagnetic log as a backup system for velocity data.

A flexure monitor system is also included in SPAMS. This system measures the amount of structural twisting that the ship experiences between the binnacle of the inertial navigational system and the radar pedestals. The output of this system consists of three flexure angles for each radar platform.

With the exception of the flexure monitor system, the data rate of SPAMS is 10 pps. The data rate of the flexure monitor system is 40 pps.

C. Communication System. Communication between the AIS and the Mission Control Center in Houston is accomplished via a high-speed link using a communications satellite and a low-speed teletype link. Data transmission to and from the communications satellite uses a separate steerable antenna, the Satcom antenna.

D. Central Data Processing System (CDPS). A simplified block diagram of the CDPS is given in Figure 1. The major component of the CDPS is a Univac 1230 general purpose computer on which the real-time

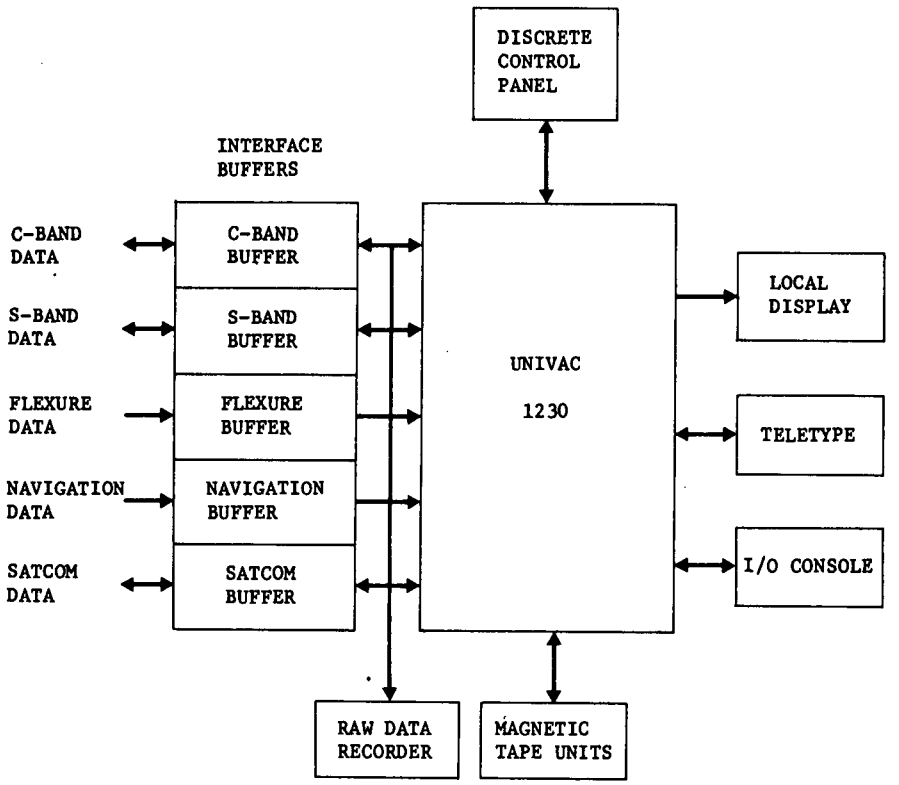


FIGURE 1

programs are run. The Univac 1230 has a main memory consisting of 32,608 30-bit words with a two microsecond cycle time and a control memory consisting of 120 30-bit words with a 400 nanosecond cycle time.

In addition to the Univac 1230, the CDPS contains the following devices:

Interface Buffers. The interface buffers provide the hardware interface between the Univac 1230 and the instrumentation systems of the AIS.

Raw Data Recorder. All data from the interface buffers is simultaneously recorded on one-inch magnetic tape by the raw data recorder as it is transmitted to the 1230. The raw data recorder also operates in a playback mode, a most important feature for system testing and check-out.

Discrete Control Panel. The discrete control panel consists of a set of 58 switches. The setting of these switches provides the primary means of operator control of the real-time program during a mission.

I/O Console. The I/O console provides operator control both during real-time and batch mode program execution. The I/O console consists of a keyboard and a paper tape reader.

Local Display. Several output devices are used for real-time display of parameters during a mission. These include a high-speed line printer, plot boards, brush recorders, and digital display windows.

Magnetic Tape Units. Six standard (1/2 inch) magnetic tape units are available for use by the 1230.

III. MISSION REQUIREMENTS

During an Apollo mission, the program which executes on the Univac 1230 is the real-time Acquisition and Tracking (ACQ/TRK) program. The following items are the major requirements that this program must meet.

A. Pointing Data. Prior to acquisition of the spacecraft by the shipboard radars, acquisition messages are transmitted to the AIS. These messages are processed in real-time and initial pointing data is computed for the radar systems. Radar pointing data is also continuously generated after the spacecraft has been acquired. This provides a means for rapid reacquisition of the spacecraft in the event that target lock-on is lost for some reason. Radar pointing data is generated at a 40 pps rate.

In addition to the radar pointing data, pointing data must also be generated for the Satcom antenna. Satcom pointing data is generated at a 10 pps rate.

B. Orbit Determination. Once the target has been acquired and valid tracking data is available, independent orbit determinations must be made based upon the data from each radar system. When the spacecraft is in powered flight a new pair of orbit determinations must be computed every second. During the free-flight phases of the mission a new pair of orbit determinations must be computed once every two seconds.

C. Parameter Computation. Flight dynamic and mission status parameters must be computed and updated for output to local display devices during all phases of the mission. The output rate requirements for these devices vary, with a maximum rate of 10 pps for the plot boards and brush recorders.

D. Data Transmission. Data is reformatted and transmitted to the Mission Control Center at Houston at a rate of two messages per second via the high-speed Satcom link and at a rate of one message every six seconds via the teletype link.

It is the responsibility of the ACQ/TRK program to insure that the above mission requirements are met. The logic of the ACQ/TRK program is covered in the following section.

IV. THE ACQ/TRK PROGRAM

The ACQ/TRK program consists of three major components: the Master Executive Module (MEM), the Real-Time Executive Monitor (RTEM), and the ACQ/TRK task agenda. MEM is the master control module for the ACQ/TRK program. When operating in real-time mode, the primary task of MEM is to process external interrupts and perform I/O operations on the standard 1230 peripheral equipment (the magnetic tape units, the line printer, and the I/O console). The major role of MEM, and the reason why these I/O functions are included at this level, is to serve as the overall system monitor, in batch mode as well as real-time mode. This aspect of MEM is covered in Section V.

The requirements that the ACQ/TRK program must satisfy, as outlined in Section III, are all cyclic in nature with a minimum cycle time of 25 milliseconds. That is to say, certain tasks, such as the outputting of radar pointing data, must be performed at a 40 pps rate, i.e., once every 25 milliseconds. A second set of tasks must be performed once every 100 milliseconds, another set once every 500 milliseconds, and so forth. The cyclic nature of the various tasks, together with the differences in cycle times (from 25 milliseconds to 6 seconds) led to the development of a multilevel task oriented processor which would permit the more frequently executed tasks to interrupt the execution of those tasks that are performed less frequently.

The allocation of processing time to the various tasks is controlled in RTEM by the task processor. This processor operates on a task agenda which consists of the coding for the tasks and a control block which specifies the frequency requirements for the various tasks in the agenda.

The basic philosophy of the task processor is illustrated in Figure 2. In this example the agenda consists of a 25 millisecond task (a task that must be executed once every 25 milliseconds), a 50 millisecond task and a 100 millisecond task. The basic cycle time of a task agenda is the cycle time of the most frequently executed task. All other tasks must have a cycle time that is an integer multiple of the basic cycle time.

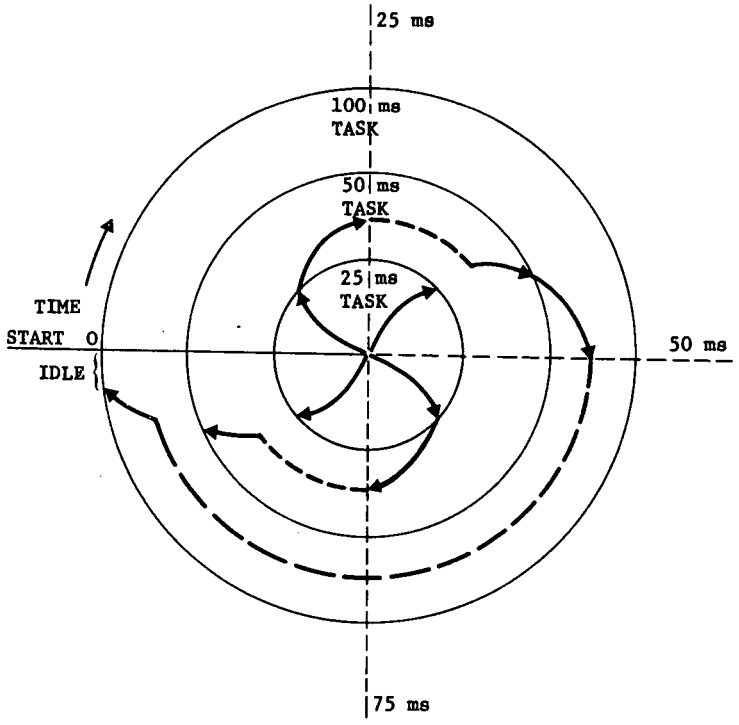


FIGURE 2

The operation of the task processor is as follows. Starting at time zero the 25 millisecond task is executed to completion. The remaining time of this 25 millisecond cycle is then allocated by the task processor to the processing of the 50 millisecond task. At the end of the first 25 millisecond cycle, the execution of the 50 millisecond task is interrupted, and the 25 millisecond task is again executed to completion. At this time the processing of the 50 millisecond task is continued from the point of interruption. The processing of the 100 millisecond task is not started until the 50 millisecond task has executed to completion.

The task processor continues to function in the above fashion until the 100 millisecond task has executed to completion. At the start of the next 25 millisecond cycle, the processing of the agenda is initiated again starting at the beginning. Thus, every 100 milliseconds the 25 millisecond task is executed 4 times, the 50 millisecond task is executed twice and the 100 millisecond task is executed once. It is the programmer's responsibility to insure that the total time required for processing this agenda takes less than 100 milliseconds.

In addition to task processing, RTEM also provides interrupt handling for the real-time data sources. All data from the real-time sources are treated as free running asynchronous inputs by RTEM. In other words, whenever an external data ready interrupt is received by the 1230 from the interface buffers RTEM will interrupt the task processor and initiate data transfer from the interface buffer to an internal core buffer in the 1230. Once this data transfer has been initiated, control is returned to the task processor. The data transfer itself then occurs in parallel with the agenda processing by the task processor. All input areas in core are double buffered in the sense that one set of input data is processed by the ACQ/TRK agenda while a new set of data is being brought into the 1230 by the interrupt processor of RTEM. In actual fact then, even though MEM is the master executive for the system, RTEM provides primary program control during real-time execution.

The ACQ/TRK task agenda consists of six task levels: 25 millisecond, 100 millisecond, 500 millisecond, 1 second, 2 second, and 6 second. A simplified flow chart of these tasks is given in Figure 3. The paragraph numbers in the flow chart commentary below correspond to the numbers in the figure.

25 Millisecond Task

1. The radar designate data is computed in the 100 millisecond task and is interpolated to a 40 pps rate in the 25 millisecond task. The interpolated data is then scaled and formatted, and output to the interface buffers is initiated.

2. A log tape of all data input is written on a standard tape unit by the ACQ/TRK program. This tape is completely independent of the raw data recorder tape. As discussed in the next section, the log tape also plays an important role in the check-out phase of the program.

100 Millisecond Task

1. Although the input data rate from the radar and flexure monitor systems is 40 pps, it is utilized at a 10 pps rate by the ACQ/TRK program. All input data is scaled, formatted and edited to eliminate bad data points at this time.

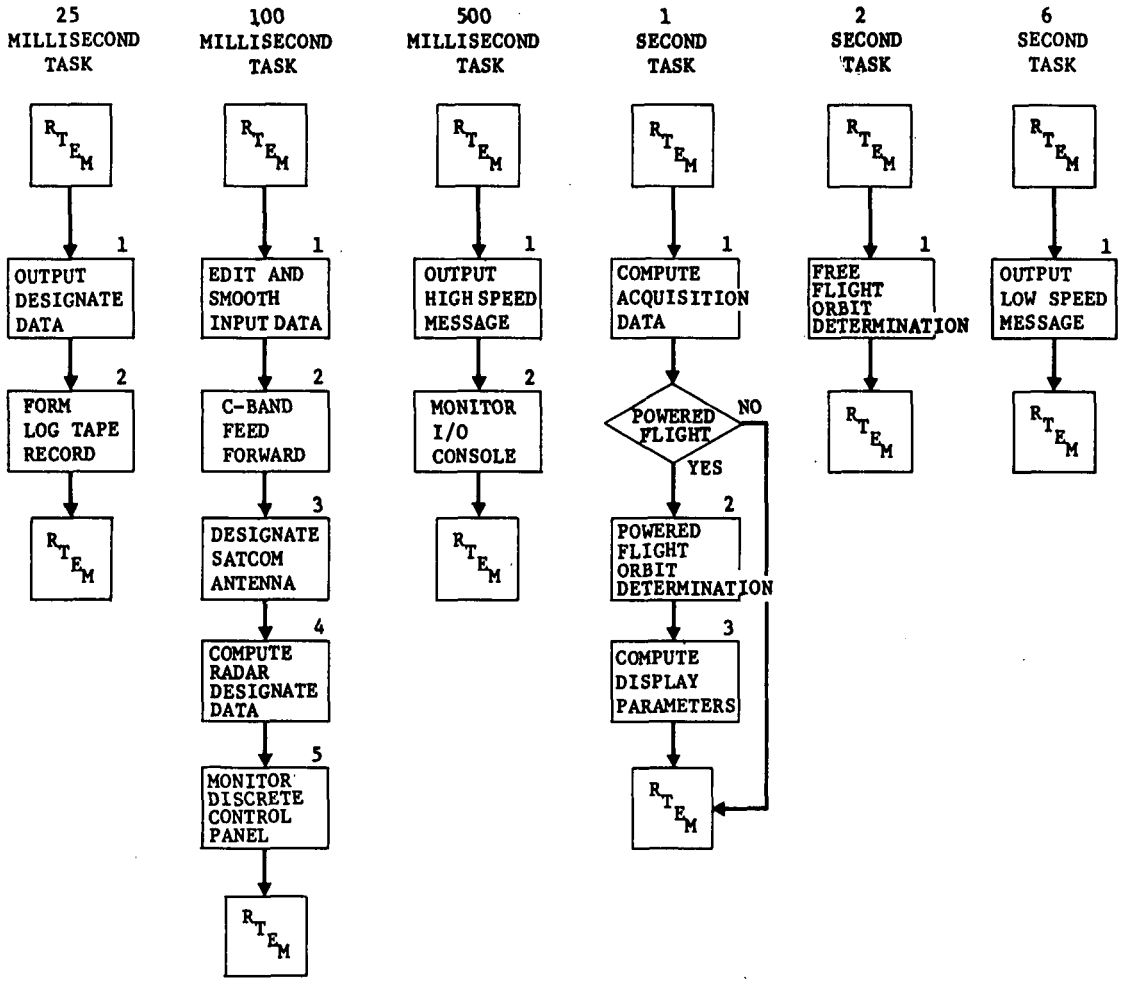


FIGURE 3

After the data from each source has been edited it is passed through a least squares polynomial smoothing filter. This filter takes one second of edited data (11 data points) and outputs a single smoothed result. Thus, the 10 pps input data rate has been compressed to an output data rate of 1 pps.

Finally, validity flags are set which indicate to the orbit determination routines whether the current input vector should be included in the orbit determination.

2. The C-band feed forward process consists of predicting the attitude of the ship ahead in time and computing a modified error signal for the C-band antenna servo system. The purpose of this is to prevent an undue amount of error in the C-band data due to servo lag. Because of different servo mechanisms, the S-band antenna does not require this procedure.

3. Designate data for the Satcom antenna is computed and outputted at a 10 pps rate.

4. The radar designate data is computed here. This data is based upon either the current best estimate of target position or an acquisition message.

5. The switch settings of the discrete control panel are sampled every 100 milliseconds in order to provide rapid program response to operator input.

500 Millisecond Task

1. The data for high-speed transmission is formatted and data transmission is initiated. Each message contains the most recent orbit determination vectors together with appropriate control information.

2. The I/O console is monitored twice a second for real-time input from the I/O console.

1 Second Task

1. Acquisition messages are processed at this time to obtain an estimate of when the spacecraft will appear on the local horizon and its coordinates at that time. This data is used by the 100 millisecond task to generate radar designate data. After the target has been acquired, reacquisition designate data is based on the most recent estimate of target position.

2. The powered flight orbit determinations are computed at this time. The algorithm for obtaining these determinations is given in Appendix II.

3. The mission parameters for local display are computed after the orbit determinations have been obtained.

2 Second Task

1. The algorithm for obtaining orbit determinations during free-flight is also given in Appendix II. After the orbit determinations have been obtained, the free-flight mission parameters for local display are computed.

6 Second Task

1. The low-speed message is formatted and transmission is initiated. This message contains only one position vector of the spacecraft (either C-band or S-band, under operator control) and no velocity data.

V. PROGRAM DEVELOPMENT AND CHECK-OUT

The development and test phases of the ACQ/TRK program were greatly facilitated by the use of a consolidated software system approach. The goal of this approach was to develop a unified operating system that would have: (1) a batch processing system for program development and initial testing; (2) an agenda preparation system for generating real-time agendas; and (3) a real-time operating system.

The components of the resulting system and their relationships are shown in Figure 4. The primary task of the Master Executive Module is to serve as a system bootstrap for loading in subexecutive systems. MEM also contains the I/O procedures for the standard 1230 peripheral equipment. Thus, these procedures are available to all subexecutives in the system.

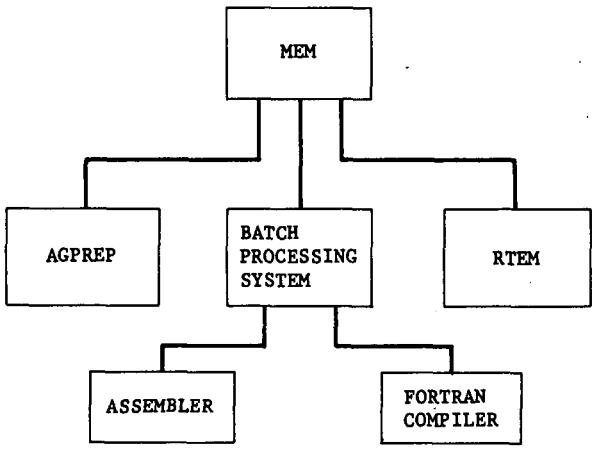


FIGURE 4

The batch operating system was government furnished. It contains a FORTRAN compiler and an assembly language processor together with an operating system for batch processing. The system was modified slightly so that it would run under the control of MEM. This system permits the 1230 to assemble, compile, and execute programs as in any typical batch operational environment.

AGPREP prepares real-time agendas that will be run by the task processor of RTEM. The input to AGPREP consists of a set of object programs. These programs are linked together by AGPREP into one module which also contains the necessary task control information. The resulting agenda is stored on a system agenda tape. Several other agendas may be stored on the same agenda tape.

As previously mentioned, RTEM is the executive program for real-time operation. After RTEM has been loaded into core by MEM, it selects the requested agenda from the system agenda tape and loads it into core. After the agenda has been loaded, control is transferred to the task processor section of RTEM and real-time execution of the agenda is begun.

In addition to the obvious advantages of working under a unified system rather than several disjoint systems, the above approach offers a great deal of operational flexibility. In particular, program modifications can be speedily implemented and a new agenda created with a minimum of delay. This proved extremely useful in modifying the ACQ/TRK agenda whenever changes to the real-time system operational requirements occurred.

A few words about the use of the FORTRAN language in the ACQ/TRK program perhaps are now in order. Since a 25 millisecond task in an agenda executes forty times every second, it is evident that execution time considerations are much more crucial for this task than for a 1 second task. The 1 second task, however, may very well be suitable for FORTRAN programming. In the ACQ/TRK program all of the mathematical algorithms associated with orbit determination, acquisition message processing and radar designation in the 1 and 2 second tasks are written in FORTRAN. In total, almost fifty percent of the final version of the program was written in FORTRAN. The results of timing tests indicated that .1 and .5 seconds, respectively,

are required to perform the orbit determination calculations in powered flight and free-flight. This easily satisfied the overall timing requirements for this portion of the agenda. The use of FORTRAN greatly enhanced both the development and test phases of the ACQ/TRK program, and its use imposed no severe penalties in execution time or storage requirements.

The development and preliminary testing of the ACQ/TRK program was carried out on a government furnished Univac 1230 test facility identical to the 1230 installations on the AIS. The testing and validation of the ACQ/TRK program was conducted in several phases. First, the FORTRAN analysis programs were developed and tested in batch mode. A simulation program was written which duplicated the inputs that these programs receive in real-time, and it was possible to completely test the logic of these programs in batch mode. While this test phase was in process, other tasks were tested by means of the log tape. As an example, a log tape was read by a special 100 millisecond test program and the logic of this task tested in a simulated environment.

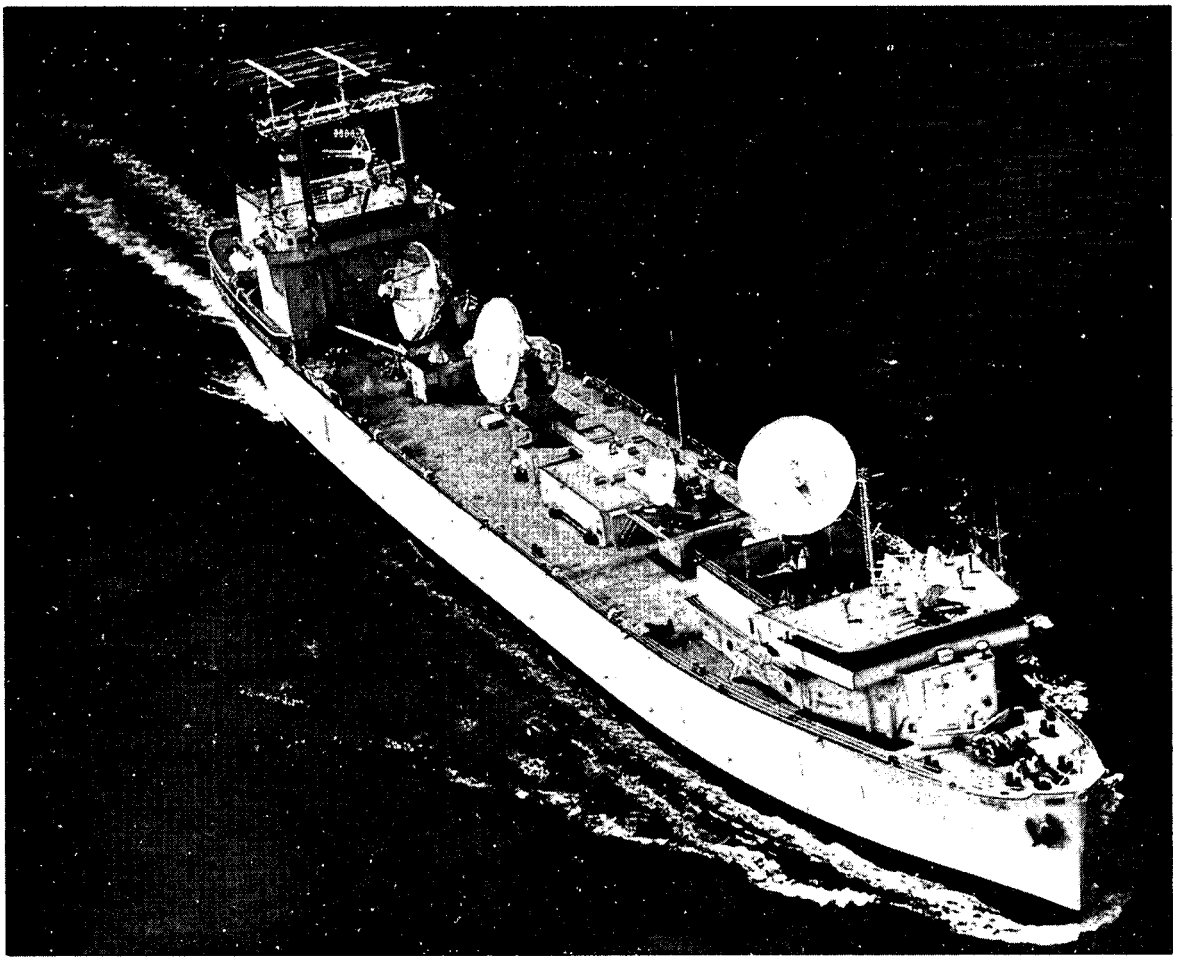
The most important phase of the validation process at the test facility was a complete real-time simulation using raw data tapes and a buffer simulator. The buffer simulator is a special hardware device which duplicates at the test facility the operation of the interface buffers. During a buffer simulator test a raw data tape is played back at the same speed at which it was recorded. The buffer simulator decodes the data on the tape and fills a complete set of interface buffers with this data. These buffers then generate interrupts in the 1230 in the same fashion as the actual interface buffers on the AIS. Thus, a very close reconstruction of the actual events of a mission can be obtained at the test facility. The buffer simulator also has a copy of the discrete control panel so that it is possible to test the reaction of the ACQ/TRK program to any operator intervention. The use of the buffer simulator permitted a very thorough testing of the ACQ/TRK program before the program was taken to the ships for final integration and testing.

The buffer simulator continues to play a very important role in the maintenance of the ACQ/TRK program and the AIS hardware. Whenever anomalies occur during a mission or a sea trial, the raw data tape can be used to recreate the problem conditions at the test facility, and an analysis performed to determine the cause of the anomaly. In addition, the raw data tape can be played back onboard the AIS so that testing and analysis can be done there. Finally, since the operating system contains all the system subexecutives, changes to the software program can also be made on the ships. This is particularly useful when a hardware anomaly occurs, and it is expedient to make a temporary program patch to accommodate the anomaly.

VI. CONCLUDING REMARKS

The system approach outlined above resulted in the on-time completion of the ACQ/TRK program. The final version of the ACQ/TRK program has been used on the ships beginning with the AS-205 Apollo mission. The program has performed without malfunction or anomaly during all real-time mission support. In particular, the Apollo lunar landing missions, Apollo 11 and 12, were successfully supported. All three Apollo Instrumentation Ships provided support for the Apollo 11 mission. Due to the reliability and accuracy of the navigational equipment onboard the Apollo spacecraft, and as an economy measure, the USNS Vanguard alone supported the Apollo 12 mission. The role of the Vanguard is to provide support during insertion of the spacecraft into near-earth orbit. The Vanguard will continue to provide this support for the succeeding Apollo missions.

APPENDIX I



The Apollo Instrumentation Ship USNS Redstone.

APPENDIX II

A. Powered Flight Orbit Determination

The input to the powered flight orbit determination program consists of the smoothed data points from the 100 millisecond task. The following input data is received each second:

i) range, elevation and bearing of the target from the C-band antenna,

ii) range, elevation and bearing of the target from the S-band antenna,

iii) the three flexure angles for the C-band antenna,

iv) the three flexure angles for the S-band antenna,

v) roll, pitch and heading of the ship,

vi) latitude and longitude of the ship,

vii) the time of the measurement set.

The orbit determination procedures for C-band data and S-band data are identical. The following is an outline of the orbit determination algorithm using C-band data. The position vector of the spacecraft in inertial coordinates is obtained by first transforming the C-band REB vector to cartesian coordinates in a coordinate system centered at the radar pedestal. Next, this xyz vector is rotated through the C-band flexure angles and translated to a deck cartesian system centered at the inertial binnacle. This vector is then rotated through the pitch, heading and roll angles into a coordinate system that is also centered at the inertial binnacle but has its xy plane tangent to the earth. Finally, this vector is rotated through the latitude and longitude of the ship and the rotation of the earth is taken into account. The resulting vector gives the position of the spacecraft in an earth-centered inertial cartesian system.

The most recent 11 inertial position vectors are passed through a second order least squares polynomial filter. This filter is then evaluated and differentiated at the mid-point of the data span. The resulting position and velocity vector is the estimate of the orbit of the spacecraft during powered flight.

B. Free-Flight Orbit Determination

The input data set to the free-flight orbit determination algorithm is identical to the input given in A above. However, a substantially more complex algorithm is used to estimate the orbit of the spacecraft in free-flight than in powered flight. The reason for this is that under normal conditions GO/NO-GO decisions for the near-earth portions of an Apollo mission are based on free-flight rather than powered flight data.

During free-flight, a modified form of a Kalman filter is used to obtain orbit determinations of the spacecraft. As in powered flight, an identical procedure is used for both C-band data and S-band data. The modification to the Kalman filter consists of the addition of an age weighting parameter to the filter. The purpose of this parameter is to have the filter place more weight on recent data and less weight on older data. The classical Kalman filter places equal weight on all data. However, since the orbit determination problem is a nonlinear problem, the linearizations involved in the Kalman filter make it desirable to deweight older data. A complete discussion of the actual equations of the modified Kalman filter may be found in the paper "A Real-Time Kalman Filter for the Apollo Tracking Ships", by F. C. Johnson, presented at The Seventh Semi-Annual Astrodynamics Conference, Goddard Space Flight Center, April, 1968.

DISPLAY AND INTERFACE IMPLICATIONS IN THE USE OF DIGITAL COMPUTERS

by

J. G. Carr

Royal Aircraft Establishment, Farnborough, Hampshire

SUMMARY

In introducing the management type of digital computer into aircraft, there are two important interfaces to consider; that between the man and the computer and that between the aircraft system and the computer.

The most versatile means of communication between the man and the computer available at present is the Cathode Ray Tube or crt in association with a Programmed Function Keyboard or pfk.

The crt display may use cursive, alpha-numeric, pictorial and TV types of presentation and this enables all the conventional display formats, together with interesting new ones, to be presented as required on the crt by selection on the pfk.

The pfk is a multi-function keyboard which is linked to the computer in such a way that the function of each key and the label associated with it, changes according to the mode selected. These keys enable, for example; the appropriate format to be selected, or the answers to a number of standard questions to be displayed, or data to be inserted into the computer using the keyboard as a typewriter and the crt to verify the data before insertion.

The information being processed in the management computer originates in sensors in various parts of the aircraft and may be in one of many analogue or digital forms. This paper briefly discusses the conversion of signals into a common digital form and their transmission by time division multiplexing along either a wire cable or a 'fibre optic cable'.

The final section of the paper briefly describes how a Comet 4 aircraft has been fitted out as a flying laboratory at the RAE, and describes the part this aircraft will play in the evaluation and further development of the techniques discussed.

1. INTRODUCTION

Airborne general purpose digital computers have reached a stage of development where they surpass analogue devices on four major counts - accuracy, flexibility, versatility and reliability. Consequently, they are being used to a greater extent in modern aircraft, both civil and military. However, most of the existing sensors and users of sensor data are essentially analogue in function and therefore present interface problems when working with a digital computer. The aircrew themselves are major users of data and the interface between the computer and the crew is the subject of a great deal of research and development at the present time.

2. AIRCRAFT SYSTEMS/COMPUTER INTERFACE

Computers suitable for the management role have binary words of at least 16 bit and usually 18 or 24 bit lengths and therefore, in principle, quantities can be represented to at least 0.002 per cent. This is much more accuracy than is needed for a representation of analogue quantities where 0.1 per cent is ample for most purposes, and there are a number of digital to analogue and analogue to digital conversion methods which are capable of this accuracy. For economic and logistical reasons, computers tend to be built in modular form. For example, the processor, which is common to all versions of the computer, and the input/output unit which itself may be modular but is special to a given application. This latter unit will contain the requisite number of A to D, D to A synchro to digital converters, etc, and the logic circuitry necessary to pass the digital data, suitably labelled and time sequenced, along a common digital highway to and from this processor. Even when the original data is in digital form an interface is required to convert the data to the correct logic levels and to time sequence it into the computer processor. In this case, the interface is much simpler and where only digital information is to be handled may be incorporated into the same unit as the processor. This is therefore one incentive to produce the original data in digital form, another one being the inherent advantages of digitally processing the raw sensor data, mentioned earlier. On the output side, ac or dc voltages or synchro-type signals can be provided to operate conventional instruments, though if a numeric instrument is satisfactory it is more economic to provide a binary coded decimal or bcd output to drive such a display.

A management computer, receives data from and transmits data to a number of sensors, other computers, displays, etc in various parts of the aircraft with all the attendant problems of cable weight, integrity of circuits, etc. Techniques are being investigated which should reduce these problems. A digital computer is essentially a time sharing device; the parts of the program are performed in sequence using the same bits of hardware over and over again. This is only possible because of the very high speed at which calculations are performed. Thus, although the output representing a given parameter is only up-dated once per program cycle this occurs so frequently that for most practical purposes the output is continuous. This same principle of time sharing which is used for handling data within the computer can also be extended to the digital transmission of data from point to point within the aircraft. The data transmission cables can be time shared with a consequent saving in cable weight. The number of signals that can be transmitted down one cable is only limited by the bit rate and the frequency of up-dating required. There are many aspects which will have to be thoroughly investigated before such multiplexing is adopted on any wide scale. For example, (a) the necessary electronics will have to be carefully miniaturised otherwise the weight saving, due to the fewer cables required, could easily be swallowed up by the weight of the boxes required at each end for coding and decoding the signals; (b) the data highway could be either in the form of a ring main, linking all the data sources and sinks to the computer or a network connecting each source or sink individually to the computer. The ring main has the advantage of being readily extended to include more sensors, but may introduce a risk of common mode failure. Radial connection is less flexible but the integrity problem is not so great.

An interesting development on which a number of UK companies are working is the use of a fibre optic data transmission link in place of conventional wires. This idea has several attractive features; the information is transmitted along the glass fibre cables as pulses of light and is therefore unaffected by electrical interference radiated by other equipment, and at the same time gives good electrical isolation between the equipment and each end of the link. The bandwidth and hence the bit rate and the number of data channels available can be increased considerably without danger of radiating interference which is what limits the pulse rate on conventional cables. There are technical problems to be solved in areas such as the mechanical construction of the fibre optic link, the coding and decoding units and the light transducers at each end, but the advantages mentioned make this development well worth pursuing.

If digital data transmission within the aircraft to a standard code were adopted this should mean that sensors and receptors at each end of a digital link could be changed without rewiring the aircraft. This could be particularly valuable for military aircraft where the role of a given airframe and therefore its avionics fit may change considerably during its life, but it also affects all aircraft during the development phase when it is so difficult to define the wiring. It is much easier to change a few address cards than to modify the cable loom.

3. AIRCREW/COMPUTER INTERFACE

The widespread use of digital computers in aircraft can, if properly engineered, reduce the work-load of the crew considerably. In civil aircraft, this could mean that fewer crew members are required or, at least, that the crew that are required need less expensive training. In military aircraft, the crew should be able to undertake more complex tasks more efficiently. However, man still has a number of distinct advantages over the best and most carefully programmed computer. Although the machine is

more accurate and handles data more quickly, man has the ability to respond to a wide variety of stimuli, recall large amounts of data and cope with unexpected situations better than the machine and this must be taken full account of as systems become progressively more automated.

The computer is capable of performing quickly and accurately all the necessary routine calculations and processing the data into the most convenient form for use by the crew, but it must then be displayed in some clear, unambiguous manner. Some quantities are better displayed pictorially or diagrammatically and others numerically. A topographical display, which is fitted to a number of military aircraft, is an example of a pictorial presentation. In the type which was developed at the RAE, a projected image of a map or chart is driven from navigation information, such that the position of an aircraft symbol on the map image represents the aircraft position relative to the ground. The crew are thus aware of their position relative to the surrounding terrain and, if a computer driven track line is also superimposed, can examine terrain features that lie ahead. If the topographical display is driven by position information supplied by a digital computer and also the map co-ordinates are transmitted back to the computer from the display, the display can be operated in "closed loop" and this brings other advantages. For example, the co-ordinates of the way points and fixed points stored as part of the flight plan can be checked by allowing the display to set itself to each one in turn. Similarly, the co-ordinates of additional points can be inserted into the computer by driving the map to the selected point. This is a quick way of inserting the co-ordinates of a target of opportunity or of a visual fix.

However, there are many other pieces of information to display, which are not appropriate to the topographical display and for these the most versatile instrument, at the present time, is the cathode ray tube or crt display (1). When a crt is driven from a digital computer it is capable of showing, by software changes only, an almost infinite variety of formats both alpha-numeric and diagrammatic, and some tubes can also show TV-type pictures either separately or superimposed on the alpha-numeric information. Thus, it is possible to replace most, if not all, of the clutter of instruments in present day cockpits by a few time shared crt displays and to present the crew with suitably processed data for immediate use. However, little would be gained by exchanging a multitude of separate instruments for one crt and a multitude of selector switches and so, again, there is the opportunity for time sharing the switches in some form of multi-function keyboard. At the RAE we are experimenting with various types of crt display and multi-function keyboard and Figs 1-10 illustrate some of the arrangements being tried. Essentially, the multi-function keyboard of mfk (sometimes called a programmed function keyboard or pfk) is linked to the computer in such a way that the function of each key and the label associated with it changes according to the mode selected. In Fig 1, one experimental model which has been flight tested is shown. Twelve of the buttons can each have one of eleven different functions, making available 132 individually identifiable switch functions in a panel about 5 inches x 6 inches. The remaining switches are of more frequent use and have a fixed function. In the model illustrated, the illuminated switch labels above each button are of the digital in-line type, which use miniature lamps and projected film images. We hope that these will eventually be replaced by small addressable 6 to 10 letter matrix labels using either electroluminescent, plasma, or liquid crystal techniques.

One other method, which is already widespread, uses part of the crt face itself for displaying the labels and the "switches" take the form of pieces of wire on the face of the tube below the label - a touch-wire system. When the wire is touched by the finger, the action indicated by the label above it is initiated. This method is not favoured for aircraft use, where gloves are normally worn. However, RAE are experimenting with labels written on the face of the tube in association with an adjacent row of miniature push-buttons - Fig 2. In this figure there are twelve such buttons shown below an $8\frac{1}{2}$ inch diagonal crt, on which en-route waypoints are displayed: 'Set Numbers' has been selected and the twelve buttons are associated with numbers 0 to 9; 'clear' and 'minus'. If more space were available it would be preferable to have 14 or 15 multi-label buttons so that 'NSEW' could be included with 0 to 9 and in addition the complete alphabet would need only two sets of labels. The 15 fixed or dual function keys could then be reduced to 12.

Most, if not all, instrument scanning is to determine either whether a parameter is between certain limits or whether its rate of change is between certain limits. Both these decisions are ideally suited to the logic of a digital computer. Only if values are outside these limits do the crew need to know their actual values. They could then be displayed automatically and the crew alerted. However, there is danger in this philosophy if carried to extremes. One problem is to be sure that the right information is presented to the man in a form to which he can react quickly in an emergency. A second problem is to retain the man's confidence that all is well when it is. It is no good saying nothing until something goes wrong - the man would probably be asleep by then anyway - his attention must be retained and this is difficult if he normally has nothing to do. This brings us to the idea of display modes. These are display formats which keep the crew appraised of the total flight situation at each phase of the flight and their use can be illustrated with reference to the display of navigation information. We divide a typical flight into a number of phases; some of which are illustrated in Figs 2-10. In Figure 2, the mode selected is 'en-route waypoints' and in this mode the keyboard has been arranged so that it can be used as a numerical typewriter for the insertion of revised information into the table of co-ordinates. If now the 'Modes' button is pressed, a selection of functions become available as shown in Figure 3 along the bottom edge of the crt. The abbreviations taken from left to right stand for: Before take off and Climb; Navigate; Descent; Howgozit (Fuel, Range); En-route Waypoints; Tactical Waypoints; Weather Report; Weather Radar. In Figure 3 "Tactical waypoints" has been selected and the crt shows a list of the co-ordinates of the waypoints that have been fed into the computer as part of the flight plan; this display also shows estimated fuel and estimated time of arrival. Figures 2 and 3 both show information which would be stored in the computer as part of the flight plan. A further piece of information which would be particularly appropriate to a supersonic transport is an estimate of the optimum climb profile determined by the

computer from a knowledge of weather conditions, weight of aircraft, etc. Figure 4 suggests how this might be displayed diagrammatically. The full curve is the optimum height versus distance profile for the predicted temperature and the dotted curves, on either side, indicate the variation appropriate to a departure of plus or minus 5 degrees from this. If now for some reason this profile is not acceptable to Air Traffic Control, the restrictions of height and time which they would like to impose can be typed into the computer via the keyboard into the box below the graph. In the example the restrictions are, "Pass through Way Point 1 at flight level 320, (ie at 32,000 ft), three minutes later than the original flight plan". When the 'Restrict' button is pressed the computer then comes back with a display (Figure 5) which shows that the pilot should level out at flight level 220 until he is within 62 miles of Way Point 1 and then resume normal climb in order to meet these restrictions. If now, this new climb profile is acceptable to both Air Traffic Control and the pilot it can be entered into the computer by pressing the 'Insert' button. This can now be used as a dynamic display, showing the progress of the flight along this profile. The small aircraft symbol on the left of the display can be made to follow the actual height profile as time progresses so that the pilot can see whether or not he is succeeding in keeping to the planned profile.

It is possible to produce a display for use during the cruise phase of flying which resembles very closely the conventional horizontal situation indicator but Figure 6 shows an alternative that may be used. This display gives, both numerically and diagrammatically, an indication of the aircraft across track error and track angle error; ground speed and drift; co-ordinates of the next stage point - No 3; the expected time of arrival at that stage point and the distance to go in nautical miles. This is of course just one of many possible ways of conveying the same information. On this figure the switch labels have changed to indicate a further range of facilities that are available during the cruise phase of the flight. They are, Navigate (which is Figure 6); Fix (illustrated in Figure 7); Attack; Vertical Navigation; Write (for typing up messages for storage or transmission); Print (for making a hard copy of messages transmitted or received); Divert (illustrated in Figure 8). The next figure - Figure 7 - is the 'Fix' display and shows in both numerical form and diagrammatic form the present position, as determined by the various navigation sensors on board this aircraft. This is a particularly well equipped aircraft with an inertial navigator, an alternative dead reckoning system and short range and long range radio navigation aids. There is also the facility for inserting a manual or map fix as indicated by 'MF' and by the lower set of lat long figures. The pilot has chosen to use the information from the inertial navigator to feed to the autopilot and the position as determined by this sensor is represented by the aircraft symbol in the centre of the circle. The differences between the positions as determined by the other sensors and the inertial navigator are shown in tabular form on the left and in diagrammatic form in the circle of 10 nautical miles diameter. This diagram enables the pilot to see at a glance whether there are any serious discrepancies between the four or five versions of present position. Figure 8 is the 'Diversion' format and shows the fuel margin that the pilot has for reaching either his destination or one of the three alternative diversions. It also indicates the change of track that will be necessary to reach each of these alternatives. Figures 9 and 10 illustrate the 'Howgozit' mode and show an airline pilot's suggestion of an alternative way of showing much the same information as Figure 8. This is a dynamic display continuously plotting fuel remaining against distance and predicting range on the basis of present fuel consumption rate. The letters along the horizontal axis represent the destination (D) and en-route (E1 E2) and arrival A1, A2, A3) alternatives etc. Figure 9 shows the predicted performance if the flight continues as planned at supersonic speed, whereas the dotted line in Figure 10 predicts what would happen if the flight had to be continued at subsonic speed. The display shows that in this case there would not be enough fuel to reach alternative No 3.

As well as showing how the details of individual mode formats might be arranged, Figures 1-10 illustrate two more general points:

a. The versatility and flexibility of the crt display when used in conjunction with a digital computer.

b. The way that the multi-function keyboard and crt display facilitate two-way communication between the crew and the computer.

These illustrations have all been of the cursive writing type of crt displaying navigation type information. These can also be used to replace many other types of instruments such as engine instruments and fuel gauges and if the ability to operate in the TV raster mode is included the scope is even wider. A TV picture of the outside world might assist in landing and taxying and Boeing have carried out some experiments in this area. At RAE we are experimenting with the TV presentation of maps or charts; a picture being relayed to the cockpit display by closed circuit TV from a modified topographical display installed in the equipment bay. This idea again, economises on valuable instrument panel space in the cockpit.

4. EXPERIMENTAL PROGRAMME

In a current RAE Navigation Systems Exercise, examples of all the types of navigation aid currently in use are installed in a Comet aircraft and made available for combination into any hybrid system (2). The outputs from all the sensors are fed into a central computer (using in some cases experimental multiplexing techniques) so that new systems may be configured by software changes only.

In the preliminary experiments in the Comet 2 aircraft, outputs from Doppler and compass and inertial navigator, Decca navigator, Omega navigator, VOR and TACAN have been processed in an Argus 400

computer, recorded on punched paper tape, magnetic tape or teleprinter and outputs provided to drive a topographical display and a crt display. This work will be continued and extended in the Comet 4 aircraft which has been specially refitted for the purpose. In the first place the displays' experiments will be confined to the cabin of the aircraft but provision has also been made for the fitting of two head-down and two head-up crt displays in the cockpit in the second pilot's position. Laboratory work continues on displays formats and techniques, multifunction keyboards, high speed crt printers, and computer software for integrated systems.

REFERENCES

1. Treadgold M G and Walters D J
The Presentation of Navigational Information.
Journal of Institute of Navigation Vol 22 No 2 April 1969 page 184.

2. Carr J G and Stringer F S
A Commentory upon Aircraft Navigation Systems
Proceedings of the UK Symposium on Electronics for Civil Aviation. Paper No 35. London September 1969.

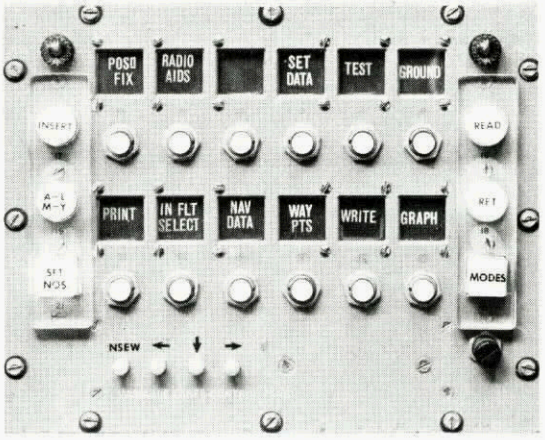


Fig.1. Multi-function keyboard with optically projected switch labels

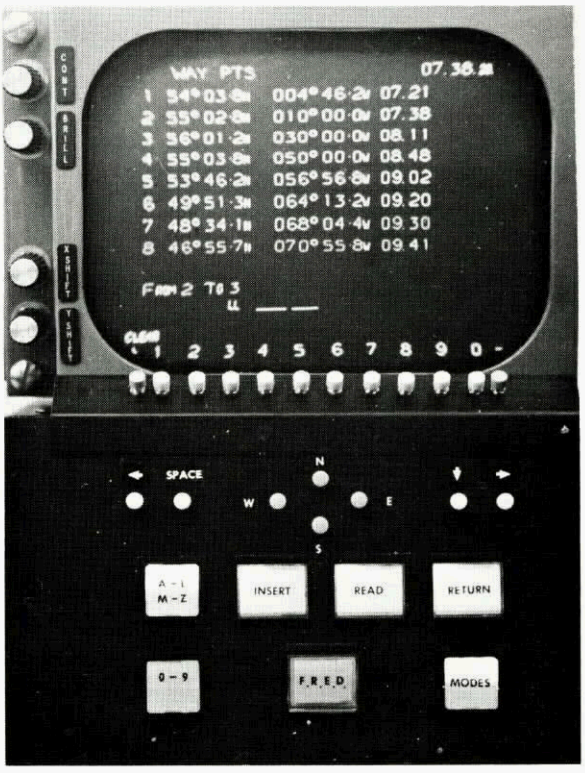


Fig.2. En-route waypoints

Fig.3. Tactical waypoints

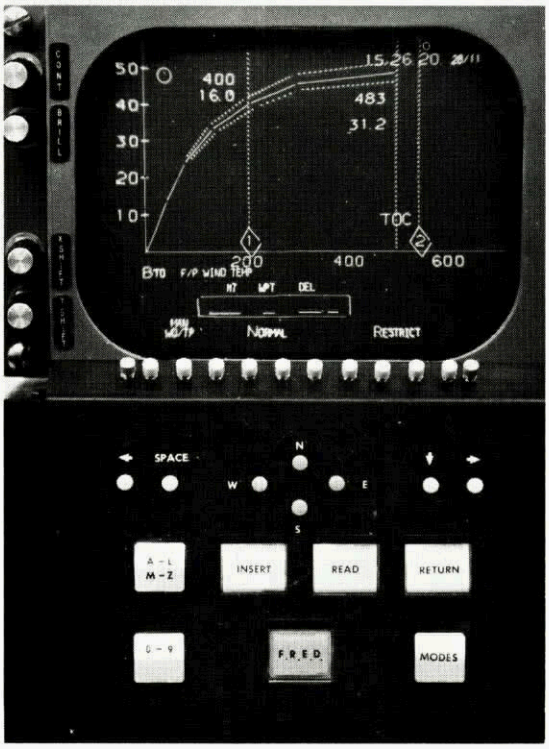


Fig.4. Planned climb profile

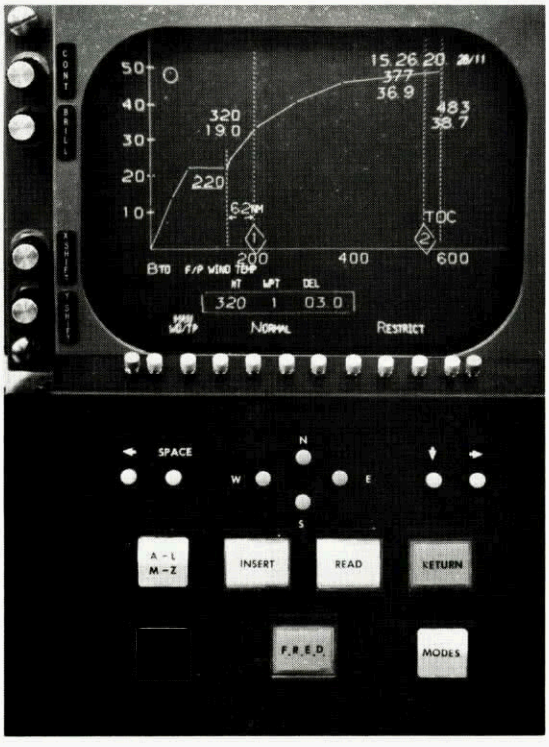


Fig.5. Revised climb profile

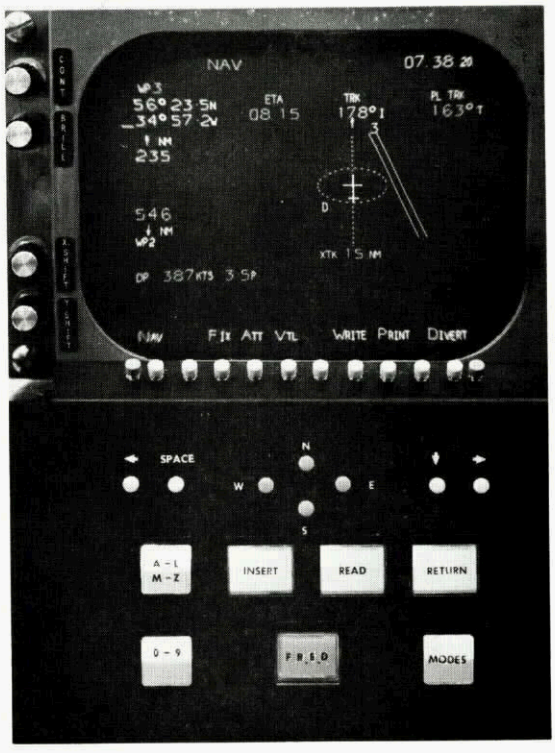


Fig.6. Navigation display during cruise

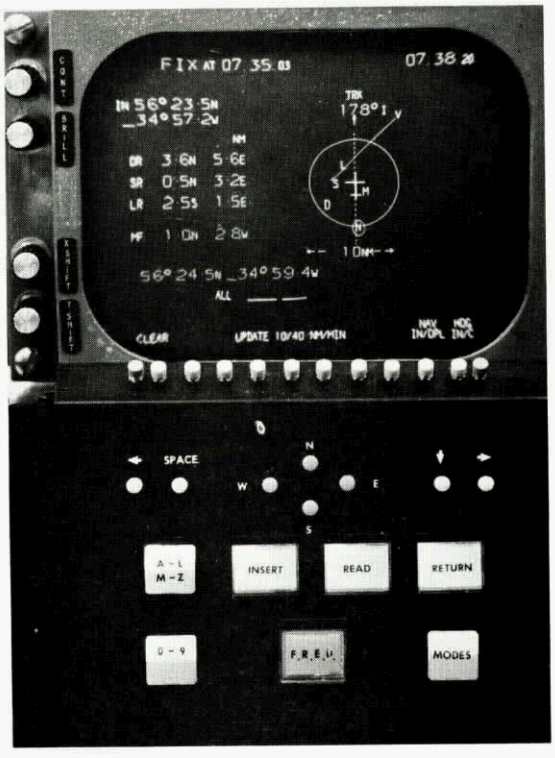


Fig.7. Position fixing

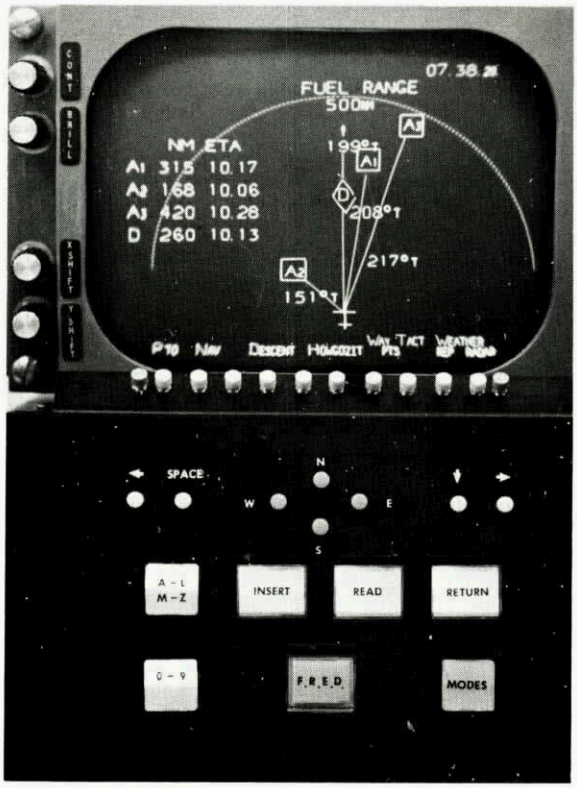


Fig.8. Fuel range and diversion alternatives

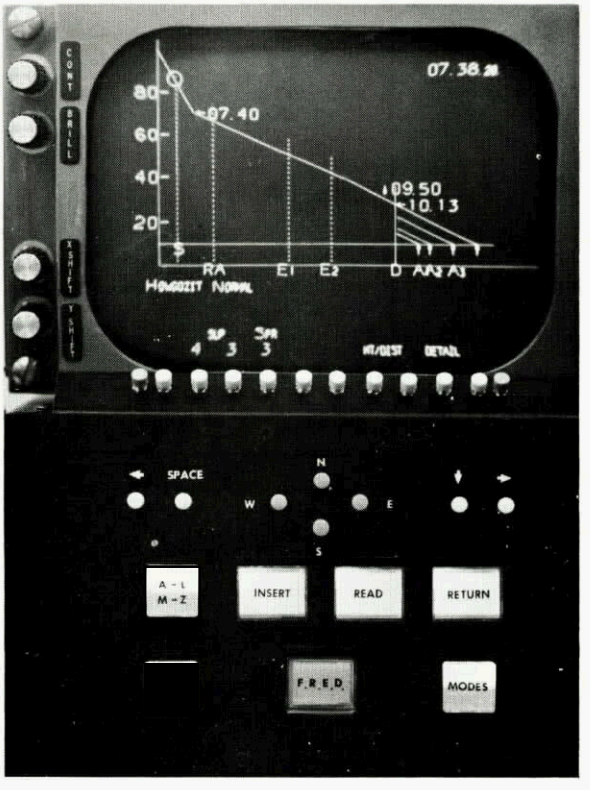


Fig.9. 'Howgozit' - a dynamic plot of fuel against distance

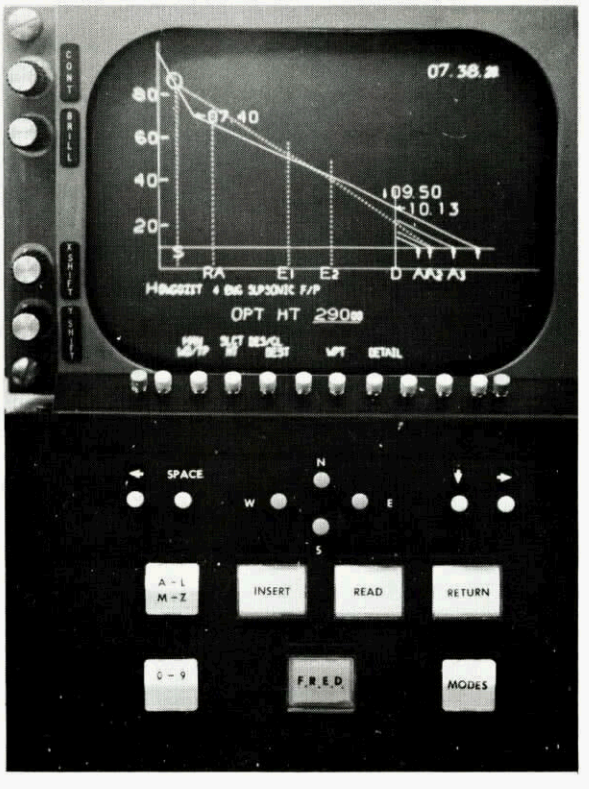


Fig.10. 'Howgozit' - revised for different flight conditions

THE TESTING OF COMPUTER-EQUIPPED GUIDANCE AND NAVIGATION SYSTEMS

by

Martin G. Jaenke

Central Inertial Guidance Test Facility (CIGTF)
Holloman Air Force Base, New Mexico, USA

SUMMARY

The use of computers in aided navigation systems to make optimum estimates of the systems state or to perform the coordinate transformation function in strapdown systems, involves the evaluation of complex computer software in the test process: Two approaches are being used for this purpose. In the first, the evaluation is based on the analysis of the recorded test outputs of the systems computer. In the second, recorded or reconstructed subsystems test outputs are input to a ground-based computer and used to "re-run" the test as frequently as desired with a variety of computer algorithms. In this way, optimization of the algorithm under actual test conditions can be achieved efficiently. Examples are given for both approaches and the digital test data collection process is described.

THE TESTING OF COMPUTER-EQUIPPED GUIDANCE AND NAVIGATION SYSTEMS

Martin G. Jaenke

I. INTRODUCTION

Computers have always been an integral part of inertial navigation and guidance systems. In addition to computing velocity and position from the indicated acceleration, they determine commands for platform stabilization, and signals to compensate for deterministic errors. Besides these quantities, which are a prerequisite to the functioning of the inertial unit, they compute the information necessary to implement the guidance concept of the vehicle and the resulting control commands.

Originally, these basic functions were performed by analog computers. However, digital computers have been increasingly employed and their inherent accuracy has contributed significantly to the remarkable performance improvement of inertial systems during the last two decades. Now, digital technology has developed explosively during the last few years, leading to computer performance characteristics in terms of size, speed, reliability and cost which go far beyond what could have been reasonably expected even 10 years ago. The availability of computers of small size, high speed and capacity, and high reliability makes it possible not only to perform a large variety of avionics and astrionics functions, checkout, self-test and other general vehicle tasks, but also provides basically new features which lead to significant improvements of the inertial system. These features and their implications with respect to inertial systems testing are the topic of this paper.

The features of primary concern which will be discussed are two-fold:

The first is the capability, provided by the computer, to make in real time optimum estimates of the true values of the quantities of interest, namely the position, velocity and heading of the guided vehicle. This estimation process makes optimum use of all information sources which are available. Besides using the outputs of the inertial unit itself, it incorporates measurements derived from Doppler radar and position fix devices such as Loran or Navigation Satellites. In combining these information sources, the process continually accounts and compensates for all known deterministic and correlated random errors and it weighs the contribution of each information source in accordance with the best knowledge of its statistical uncertainty. Thus, the process achieves results which are better than those obtainable from a mere superposition of the various measurements. In this way, the computation process reduces the performance requirements on actual physical hardware, a factor which is of economical significance. The process is well known as "Kalman Filtering" and will be referred to as such in this paper.

The other improvement made possible by increased computer capacity, reliability and speed is strapdown technology. Again, the computational process of transforming the accelerometer outputs into another defined coordinate system in accordance with observed gyro outputs replaces mechanical hardware, in this case the gimballed platform. Thus, a definite advantage in systems size, initial cost and maintainability is realized. While the shortcomings of strapdown systems are fully recognized, this advantage is significant enough to reserve an important place in future inertial technology for strapdown systems.

In both the Kalman Filter and strapdown system algorithm areas, the computer software is immensely more complex than in the conventional computer application for inertial gimballed systems. In both cases, the analyst-programmer must have an intimate knowledge of the environment in which the inertial system will have to operate and needs a thorough understanding of the influence of this environment on the errors of the system. However, this knowledge and under-

standing can never be perfect and it must be backed up by test results to avoid erroneous conclusions. Thus evolves a new aspect for the testing of such systems. That is the necessity to evaluate computer software. Here, besides "debugging" the program, the purpose is to determine whether the assumptions made with respect to the structure of the operational environment and its effects on the systems errors were correct and comprehensive. Furthermore, it must be determined whether these effects are considered in the program in an optimum way and, if not, corresponding improvements must be made. Thus, the test process becomes a part of the development process. Since software can be changed much easier than hardware, it can be seen that the transfer of systems tasks from the mechanical hardware to the computer software contributes to the flexibility and adaptability of the system.

To meet these test goals, the test environment must be identical to, or if this is not possible for practical reasons, as similar as possible to the operational environment for which the system under test is being developed. This requirement lends new and increased emphasis to the "operational" test procedures which are available at Holloman AFB. These are the tests performed in aircraft such as the C-130 or similar test beds on aircraft navigation systems, and the sled tests on the Holloman High-Speed Track, performed on missile guidance systems. Both techniques are well established and have been used successfully for many years.

Aircraft testing provides a wide variety of test trajectories which are flown over the White Sands Missile Range and its vicinity and use its precision radars as reference instrumentation. They are designed to exercise specific error terms. A prominent example is a SE-NW trajectory with turn-arounds every 42 minutes, designed to excite the Schuler cycle and coupling terms. In addition, long-range transcontinental trajectories are used which are specifically designed for the testing of aided navigation systems. Such trajectories which are presently in use and in preparation are shown in Fig. 1. They use existing range instrumentation on Government ranges to the widest extent possible and provide a large number of precisely surveyed check points for reference and position updating purposes. Besides providing long-range E-W, N-S and 45° flight paths, they include instrumented over-water flights to determine Doppler radar error due to ocean surface motions. And they extend into areas of the eastern United States where Loran coverage is available to evaluate aided inertial navigation systems which use Loran for position updating.

To reduce the cost of an operational test program for aircraft navigation systems, a van test capability has been established. Such a test van is essentially a moving laboratory which provides the means to check out and "debug" a system under moving conditions before it is submitted to the relatively expensive tests in aircraft. The van provides a substitute for Doppler radar in the form of a fifth wheel which very accurately measures the velocity of the vehicle and provides information which can be used for velocity updating of the system under test. Position updating is achieved by accurately surveyed check points along the route the van is taking. Thus, by a flexible combination of pre-flight bench tests, van and aircraft tests, highly meaningful and cost effective operational test programs for aircraft navigation systems can be performed. (1)

Another method of operational testing, the Holloman high-speed, precision test track, is used extensively for the testing of the boost phase guidance systems of ballistic missiles and space vehicles. (2) It is now being used increasingly for the testing of inertial systems for the mid-course flight phase of strategic missiles and of the non-inertial seeker systems of their terminal phase. (3) A high-g capability is in preparation, suitable for testing systems under re-entry conditions. Representative sled trajectory characteristics for these various purposes are shown in Figs. 2a and 2b.

The availability of these operational test methods provides a unique capability for testing, evaluating and optimizing the computer software of guidance and navigation systems. The following sections will be concerned with describing the methods used and experiences collected with the software of Kalman Filters of aided aircraft navigation systems and of the strapdown algorithms of missile guidance systems.

II. APPROACHES TO SOFTWARE EVALUATION

The task of testing, evaluating and optimizing complex software is a relatively new item in a guidance test program. A systematic and comprehensive body of theory and experience for this purpose is not yet in existence. However, the analysis efforts conducted in the past on various test programs show trends which are promising for developing such a unified body of theory and practice. Essentially, these trends are two-fold:

The first trend is based on the recorded output data of the systems computer, i.e. on the information which the system as a whole derives from the test. The analyst who compares this information with the reference instrumentation data must use the resulting error functions to interpret observed deficiencies with respect to their cause and to propose corresponding improvements. This is not easy in view of the complexity of the computer programs involved and in general requires the availability of a simulation program. Simulation provides the capability to test the validity of the hypotheses made concerning the cause of observed deficiencies and to test the

efficiency of proposed improvements. Generally, such a simulation program is available, mostly at the plant of the contractor who developed the guidance system under test. If the knowledge about actual test conditions is judiciously used to update the simulation program, a realistic and efficient tool for system evaluation and optimization can be obtained. Consequently, availability of such a simulation program at the test agency is desirable to allow an unbiased and critical evaluation of the system under test.

The other evaluation trend is based on the recorded output data of the subsystems under test, primarily of the inertial unit and of aiding devices such as Doppler radar. Having these subsystem data available, it is then possible to run them through the systems computer algorithm, mechanized on a ground computer, in the post-flight analysis. If this algorithm is not fully satisfactory, it is relatively easy to change the computer program and to "re-run" the test on the ground until a fully optimized algorithm is achieved, without actually conducting a new test. The number of actual test flights necessary to confirm the validity of the proposed optimization in the algorithm is then relatively small.

A version of this approach is to reconstruct subsystem output data if they are not available in an unperturbed form. This is the case if a closed loop Kalman filter is employed, i.e. if the filter performs physical corrections on the subsystem itself. The reconstruction is possible if these corrections are recorded and properly considered in the process.

None of the approaches described above is inherently superior. Their usefulness depends on the characteristics of the individual test program. For example, in some cases strapdown systems had to be tested without a systems computer being made available for the test. In this case, the evaluation approach based on the recorded subsystems output was for all practical purposes the only possible one. On the other hand, in the tests of aided aircraft navigation systems the approach based on the recorded systems computer outputs is normally preferred. The reason for this is that these data are digital in nature and thus lend themselves to post-flight analysis in a ground-based digital computer with a minimum of instrumentation and data processing time requirements. The reconstruction approach mentioned above was also successfully employed in the evaluation of an aided navigation system.

Thus, the most appropriate evaluation approach must be found in each individual test case. For the evaluation of aided aircraft navigation systems in general it is felt that the most promising and efficient approach is one based on systems computer output data, and which reconstructs as many as possible of the subsystems data and uses all this information to update a ground-based, post-flight computer simulation process. This in turn can then be used to determine sources of error and to optimize the systems computer algorithm by re-running the test on the ground computer as frequently as desired. In this way, a highly cost-effective test program can be achieved.

III. EVALUATION OF AIDED AIRCRAFT NAVIGATION SYSTEMS

The majority of aircraft navigation systems employ Kalman filters which serve the purpose of establishing optimum estimates of the true value of such quantities as aircraft position, velocity and heading. These optimum estimates are based on additional information derived from "aiding" sources such as Doppler radar for velocity updating and Loran, Navigation Satellites and check points for position updating. The filter must know the deterministic and statistical errors of these various information sources and it must properly apply this knowledge to establish truly optimum estimates. The filter designer uses the best available a priori information about these error sources when he programs the filter algorithm and he normally achieves remarkable success when he uses his filter in a computer simulation run. However, reality may look quite different from what was assumed in the simulation process and this is why the testing and evaluation of such filter-equipped systems in a test environment which is as closely identical as possible to the actual future operational environment is of such significant importance.

To fully appreciate the difficulty of the task of filter evaluation, it is necessary to obtain an understanding of the status and trends, and of the complexity of filter design.

Filter designers must strive to minimize computer capacity requirements in view of the many other tasks which the central computer has to meet. A typical compromise is to make the computer available to the filtering process for about 20% of the total time. To obtain an effective filter algorithm under these restrictions, the following factors have to be considered.

The quantity which primarily influences the complexity of the filter is the state vector. It is now general practice to use as the state quantities which are to be estimated by the filter the error of the outputs of the inertial system and not the outputs themselves. (4) This leads to the estimation approach which is shown in Fig. 3. This approach reduces computer requirements because the model of the systems errors, which has to be mechanized in the algorithm, is simpler than the model of the system itself. Furthermore, the cycling rate of the filter, i.e. the rate at which measurements have to be taken and the estimates recomputed and updated is lower because the errors of the systems outputs fluctuate at a slower rate than the outputs themselves.

Of significant importance is the dimension, n, of the state vector because the matrices which have to be manipulated are of order n. The memory location requirements are roughly proportional to n^2 and computer time requirements to complete an estimation cycle are proportional to n^3. A minimum figure for n is about 5. In this case the filter would only estimate the quantities of direct interest (two horizontal positions and velocities, and heading). Since in this case the filter would not make use of the predictability of deterministic error sources and of random sources with long correlation times, its value would be highly questionable. A maximum for n is about 20 and typical values used in practical systems are about 15. For such a state vector the array storage requirement would be about 1000 word locations and the cycle time about 8 sec., a figure which is valid for a not overly fast computer with floating point hardware. Since the updating time requirements are typically within 1 to 3 minutes, sufficient computer capacity is left for other purposes.

To make fullest possible use of all available information sources, aircraft navigation systems are usually required to operate in various modes. Typical examples of such modes are: Ground align, Normal Navigation (using the inertial system and all available aiding sources), Inertial only, Doppler only, or other "degraded" modes in which only a part of the navigation subsystem is being used. In practical cases, up to 25 modes and sub-modes have been observed. Mode switching can be automatic or manual, depending on certain criteria. In either case, the computer routine must provide the appropriate initial conditions at each mode transition. This is important because the Kalman filter routine usually is different for each mode. In particular, in modes where the inertial unit is not available, the definition of the state vector in terms of error states is no longer applicable and a new definition in terms of the navigation states themselves must be formulated.

As mentioned before, in most of the tests performed at Holloman on aided aircraft navigation systems, the Kalman filter was evaluated by using the recorded systems computer output data to determine deficiencies by inspection and with the help of a simulation routine. Table I summarizes and classifies the observed deficiencies.

TABLE I

Observed Deficiencies of Kalman Filter Algorithms

Type of Deficiency	Comments
Inadequate modeling	See example of Fig. 4. Note wide discrepancy between predicted and actual performance in early flight, 4a, and improvement in later flight due to better modeling, 4b.
Insufficient word length	See example of Fig. 5. The estimate of gyro drift was significantly improved by applying double-precision arithmetic. This in turn improved estimate of position errors.
Programming assumptions	A reduction in position error by a factor of 5 was achieved by making the updating of transition and measurement matrix coincide in time. Before that, the measurement matrix was updated throughout the Kalman cycle and the transition matrix only at the start of the cycle.
Timing and scheduling problems	In one case, improvements were achieved by: (a) Changing the time of entry of Magnetic Variation information, (b) Changing the time spot of Radar Antenna stabilization to the inertial unit.
Reasonableness criteria	The programming of this criteria for position fixes had to be changed during test series to avoid rejection of an accurate position update.

The "reconstruction" approach to filter evaluation was also investigated and the results were published. (5) As will be recalled from section II, the approach consists of reconstructing the outputs of the subsystems as they would have been without the influence of the Kalman filter. Then these unfiltered outputs are fed into a ground-based simulation of the system Kalman filter during the post-flight analysis and the test re-run on the computer as frequently as necessary to debug and optimize the Kalman routine. Fig. 6 shows the system error observed during a test flight as published in Ref. (5). Figures 7 and 8 show the reconstructed outputs of the inertial and Doppler unit, respectively. As can be seen, the large longitude error is attributable to the inertial unit and the large latitude error to the Doppler.

These results demonstrate how valuable this approach is to localize error sources. However, the computational effort involved in such reconstructions is quite considerable and consequently this approach has not been pursued further. But, as stated before, a reconstruction such as shown in Fig. 9, would be the most efficient solution for the evaluation of Kalman filter equipped navigation systems. It is based on the recorded output data of the systems computer and uses a comprehensive and flexible simulation program for determination of deficiencies and system optimization.

IV. EVALUATION OF STRAPDOWN SYSTEMS

The CIGTF has performed a variety of strapdown tests in the past. Its laboratory facilities were used for the testing of components and systems. In these laboratory tests, particular emphasis was placed on the development of methods involving angular motion inputs. (6) Furthermore, strapdown systems were tested on the sled (7) and in aircraft. In many cases, the systems submitted for test were delivered without a systems computer. To obtain meaningful answers from operational tests in such a case, it was necessary to record the outputs of the accelerometer and gyros and process this information in a ground-based computer algorithm. In this way it was possible to obtain results which were equivalent to those of a computer equipped system. Besides substituting for the missing systems computer, this method makes it possible to use different algorithms in re-runs of the actual tests on the ground computer. Various proposed algorithms can thus be compared and an optimization can be performed.

A particularly interesting example for such an evaluation approach is the sled test program performed on the Lunar Excursion Module Abort Sensor Assembly (LEM/ASA). This system and the sled test results obtained have already been described in considerable detail to the AGARD Meeting on Inertial Navigation in (8). However, the evaluation of the sled tests was not complete at the time when that report was given and the results obtained since then are of specific interest in the context of this paper. (9) The motion profile of the sled used in these tests was similar to the one shown in Fig. 2. While the translational vibration environment fairly well simulated the true operational environment, it was not possible to suppress the rotation of vibrational accelerations to the desired level. Thus, the system was overtested in this respect. But considering the facts that the sled is the only test device which provides trajectory acceleration and translational and rotational vibrations simultaneously, this is not necessarily detrimental. Damage to the gyros was avoided by keeping the maximum angular rate below the specified limit.

One explicit objective of the tests was to compare two proposed computer algorithms and to evaluate their influence on systems accuracy. Both algorithms used direction cosines, second order integration and provided compensation for gyro and accelerometer scale factor and bias errors. However, only one provided compensation for the mass unbalance term of the gyros. Fig. 10 shows the comparison of the system velocity errors obtained with the two algorithms and indicates that the mass unbalance term is quite significant for the gyro which is exposed to the high trajectory acceleration.

Another interesting result of the tests was that the analysis of the systems velocity error showed no evidence of an influence of the vibration environment. This was confirmed by the following: In a number of computer re-runs, the rate of updating the direction cosines was changed from 20 to 200 updates per second. Since the peak of the vibration spectrum is in the vicinity of 20 Hz it could be expected that a change in the velocity error would occur at the low updating rates if the vibration influence were significant. Such a change would be due to the aliasing effect which is generated if the sampling rate is not high with respect to the frequencies involved. However, such an effect could not be observed since the system velocity errors were not affected by the changes in updating rate. Thus, it was concluded that the influence of vibrations was not significant and that the system was capable of operating in the sled environment or a corresponding operational environment.

V. TEST DATA COLLECTION

The output information of aircraft navigation systems under test is available in the systems computer. Since these data are digital, they are inherently well suited for processing in a ground-based digital computer for post-flight evaluation and analysis. All evaluation schemes should therefore be based on these digital data and avoid the necessity of collecting analog systems data with the ensuing analog to digital conversion requirement.

Although the format of each systems computer output is different there are various possible ways of making it compatible with the input requirements of the ground-based processing computer. The approach illustrated in Fig. 11 has proven the most efficient one, making correctly-formatted data available immediately on landing of the test aircraft and in this way leading to a minimum time requirement for obtaining processed data for the analyst. This is achieved by performing the necessary format conversion process on-board the test aircraft before the data are recorded on an incremental magnetic tape recorder for further ground processing. In essence, this process is a serial to parallel conversion. The serial output data of the systems computer, consisting of words of different word length for different systems, is fed into a 48-bit shift register which is

read into a memory in 6-bit bites. This is the basic conversion process. The memory content is then transferred to the magnetic tape when a full "frame" of words is assembled. The sequence is repeated for the next frame which is available after the systems Kalman filter completes its next computation cycle. Such a frame comprises all the words which are required for the evaluation of the systems test. The test analyst points them out to the system designer who makes them available for readout, together with a frame sync word, by appropriate programming of the systems computer.

Thus, with the adaptability of the on-board conversion equipment to any word length up to 48 bits, and with the freedom of defining frame contents which are most suitable for the evaluation of a particular system, a considerable flexibility of the data collection process is achieved. This contributes significantly to its efficiency and to the speed by which processed data are made available for analysis.

VI. CONCLUSIONS

The examples which were presented demonstrate the significance of the interactions between test system performance and the systems computer software. The design of test conditions and of analytical procedures which help to expose these relationships is therefore of increasing importance as the role which computers are playing expands. The experience collected in the tests and evaluation of computer equipped guidance and navigation systems will be a valuable asset for the evaluation of computer controlled integrated avionics systems.

REFERENCES

(1) Aircraft Inertial Navigation System Test Program Information, Air Force Missile Development Center, MDC-TR-66-109, Sept 1966.

(2) Test Program Information, Sled Testing of Inertial Guidance Systems (In preparation).

(3) J. D. Taylor, Sled Testing of Terminal Guidance Seeker Systems for Short-Range Missiles, Central Inertial Guidance Test Facility, Working Paper WP-MDSSS-69-1, Nov 1969.

(4) R. G. Brown, Analysis of an Integrated Inertial/Doppler-Satellite Navigation System, Engineering Research Institute, Iowa State University, Technical Report ERI-62600, Nov 1969.

(5) L. D. Brock, "The Analysis of Navigation Systems that use Kalman Filtering", Proceedings of the Fourth Inertial Guidance Test Symposium, Air Force Missile Development Center, MDC TR 68-76, Nov 1968.

(6) T. D. Magnuson, "Equipment and Instrumentation for Gyro Dynamic Tests", Proceedings of the Fourth Inertial Guidance Test Symposium, Air Force Missile Development Center, MDC TR 68-76; Nov 1968.

(7) R. E. Holdeman, F. F. Kuhn, G. H. Raroha, "Analysis Philosophy for Sled Testing Strapped-Down Guidance Systems", Proceedings of the Third Inertial Guidance Test Symposium, Air Force Missile Development Center, MDC TR 66-106, Oct 1966.

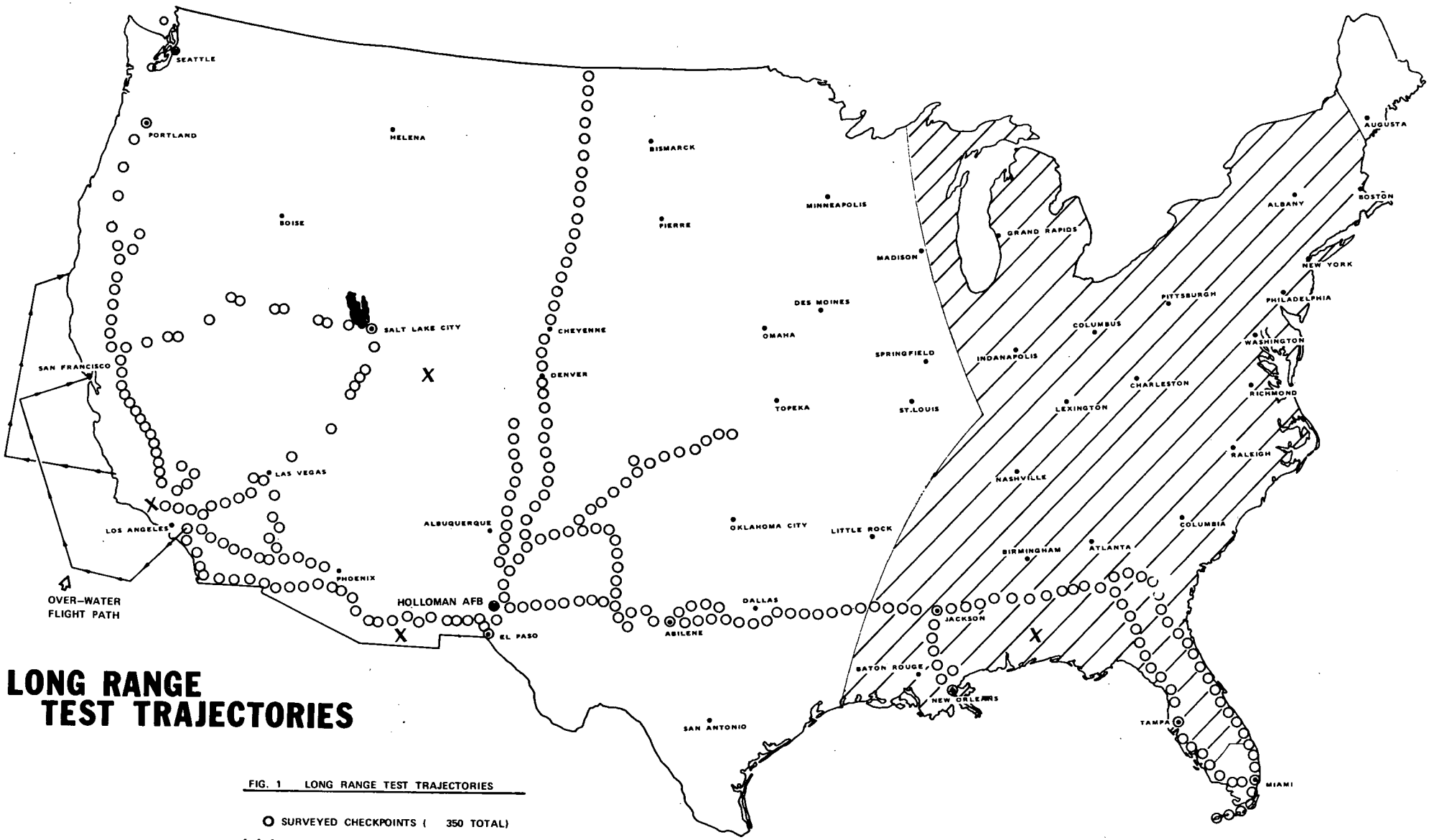
(8) J. Yamron, "Description of a Strapdown Inertial Measurement Unit", AGARD Conference Proceedings No. 43, May 1968, p. 71.

(9) L. D. Beggs, D. A. Lorenzini, N. L. Ingold, E. Simaitis, N. C. Belmonte, Final Report, Laboratory and Sled Tests of the Lunar Excursion Module Abort Sensor Assembly, Air Force Missile Development Center, MDC TR 68-11, Feb 1968.

ACKNOWLEDGEMENTS

The assistance of the following engineers and analysts of the CIGTF, who made information available and participated in discussions, is gratefully acknowledged:

Captain F. Henderson and Lieutenant R. E. Witters (Aircraft Navigation System Flight Tests), N. L. Ingold (Sled tests of strapdown systems), and Lieutenant C. A. Boldt (Test instrumentation).



LONG RANGE TEST TRAJECTORIES

FIG. 1 LONG RANGE TEST TRAJECTORIES

O SURVEYED CHECKPOINTS (350 TOTAL)

/ / / AREA WITH LORAN COVERAGE

X GOVERNMENT RANGES WITH TRACKING RADAR

■ HOLLOMAN AIR FORCE BASE

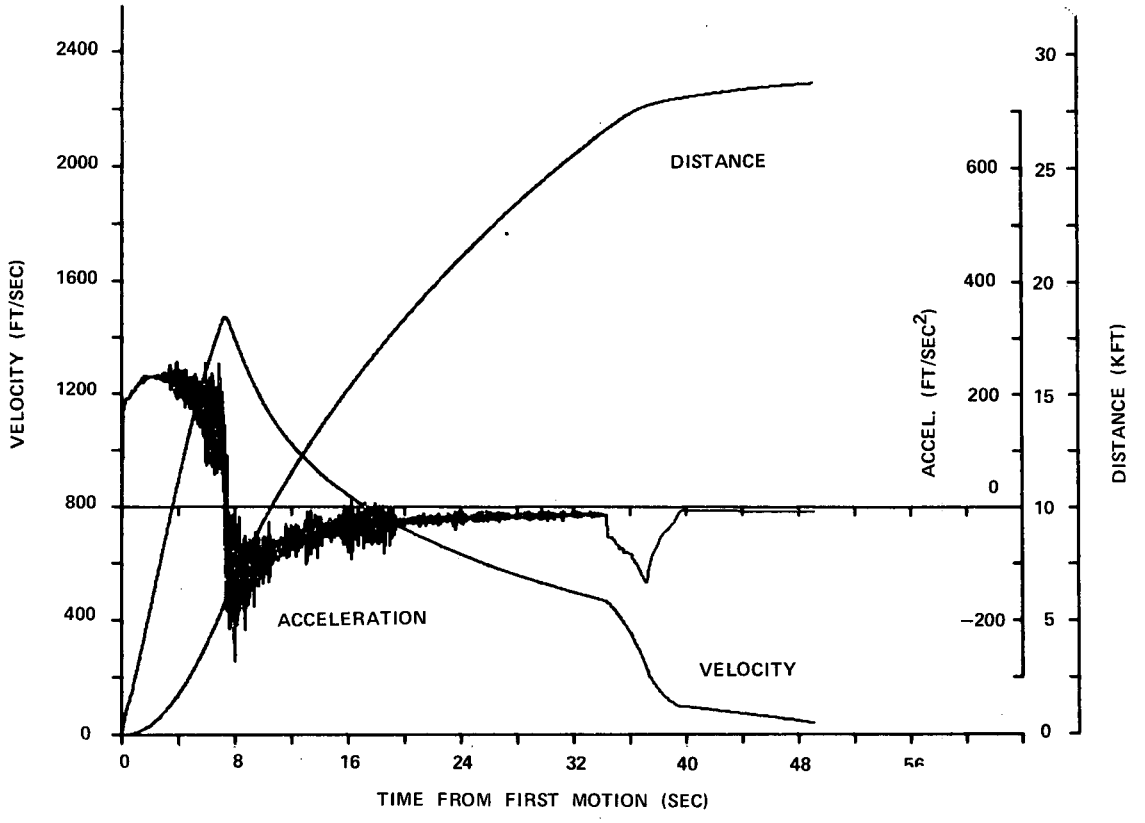


FIG. 2a SLED TRAJECTORY
BOOST PHASE SYSTEM TESTING

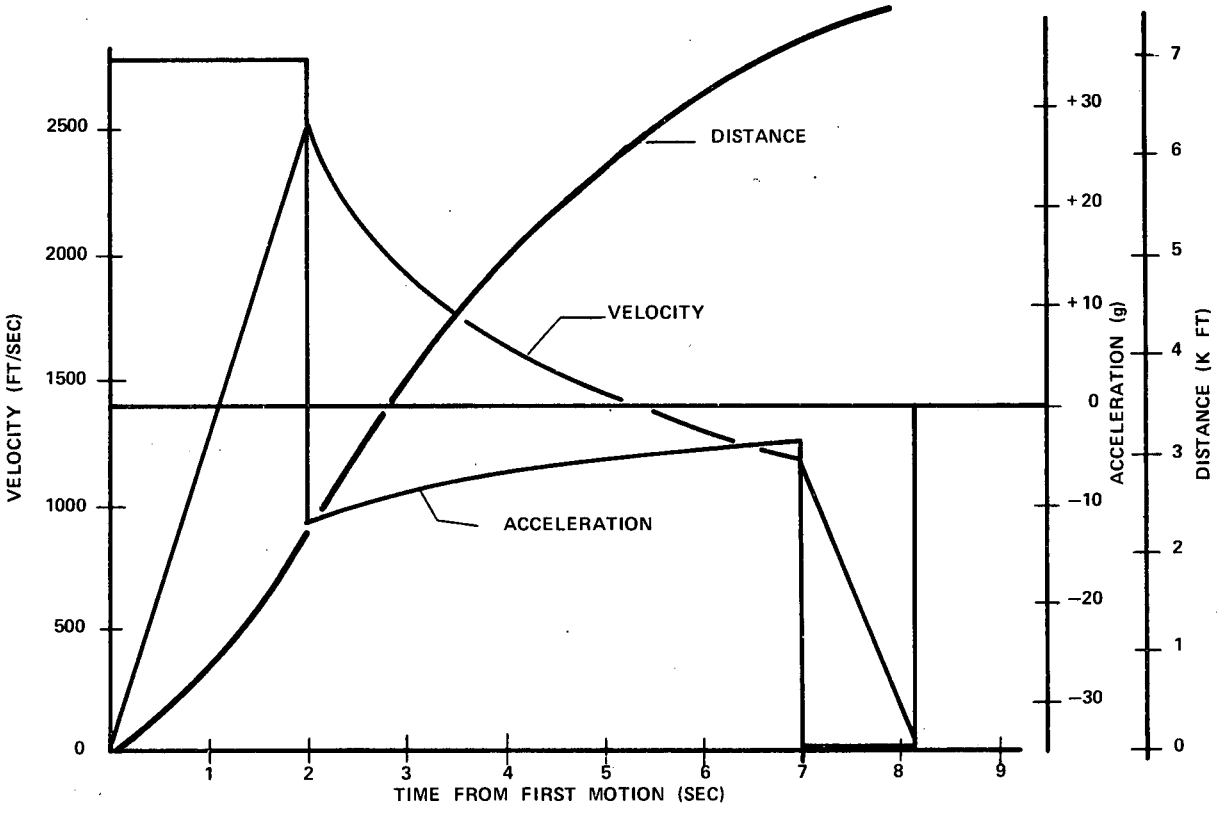


FIG. 2b PROPOSED SLED TRAJECTORY
HIGH – g TESTING

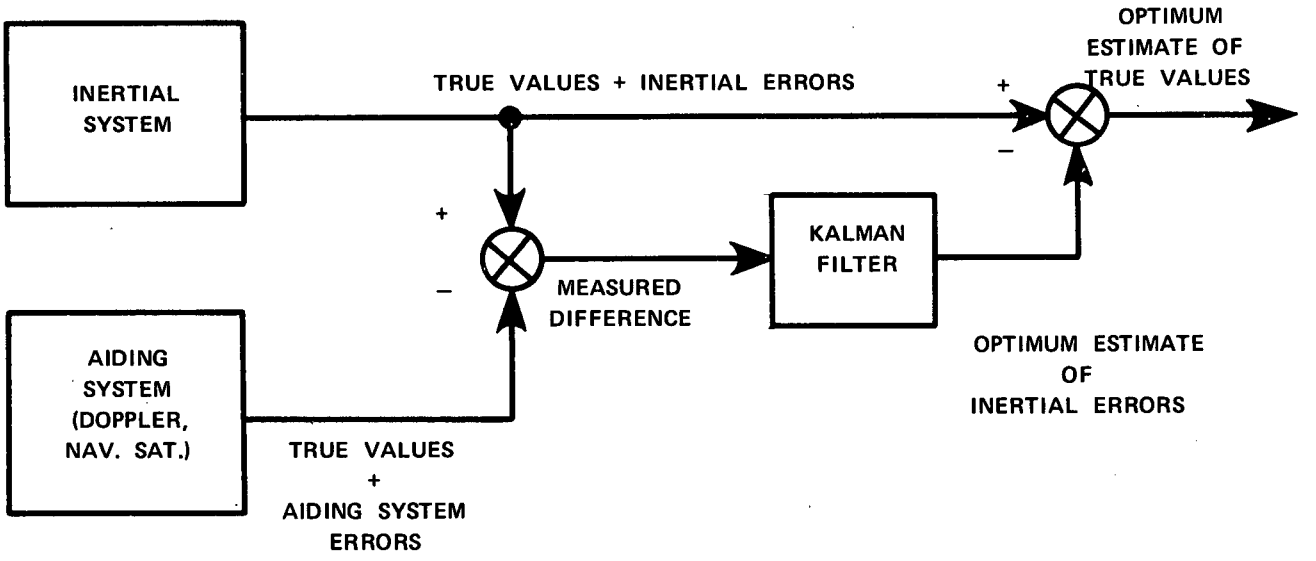
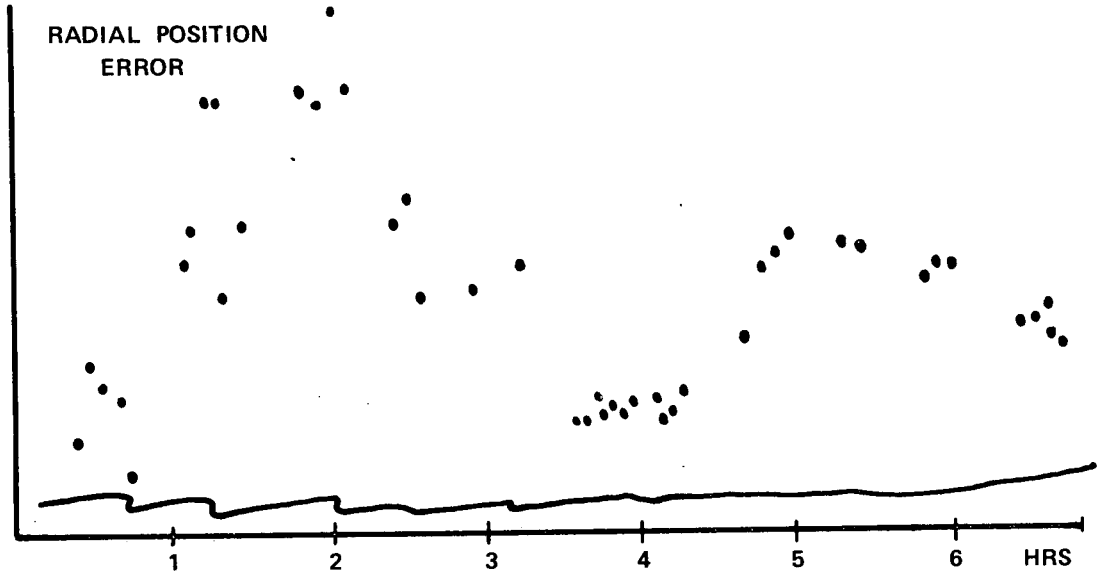


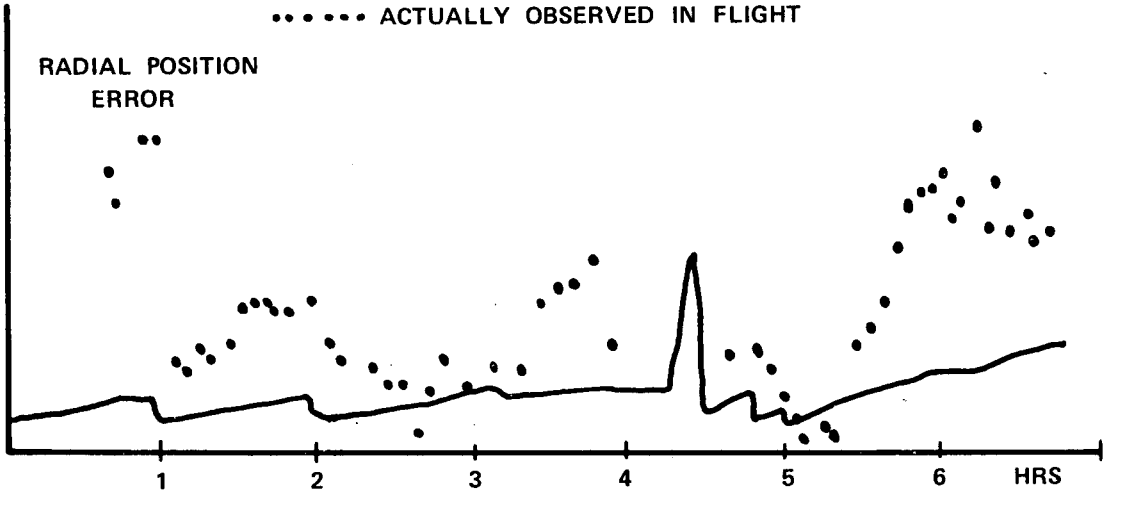
FIG. 3 ESTIMATION OF TRUE SYSTEM STATES



(a) AT FIRST FLIGHT IN TEST SERIES

——— PREDICTED BY FILTER

• • • • • ACTUALLY OBSERVED IN FLIGHT



(b) AT LAST FLIGHT IN TEST SERIES

FIG. 4 EFFECTS OF INADEQUATE MODELING

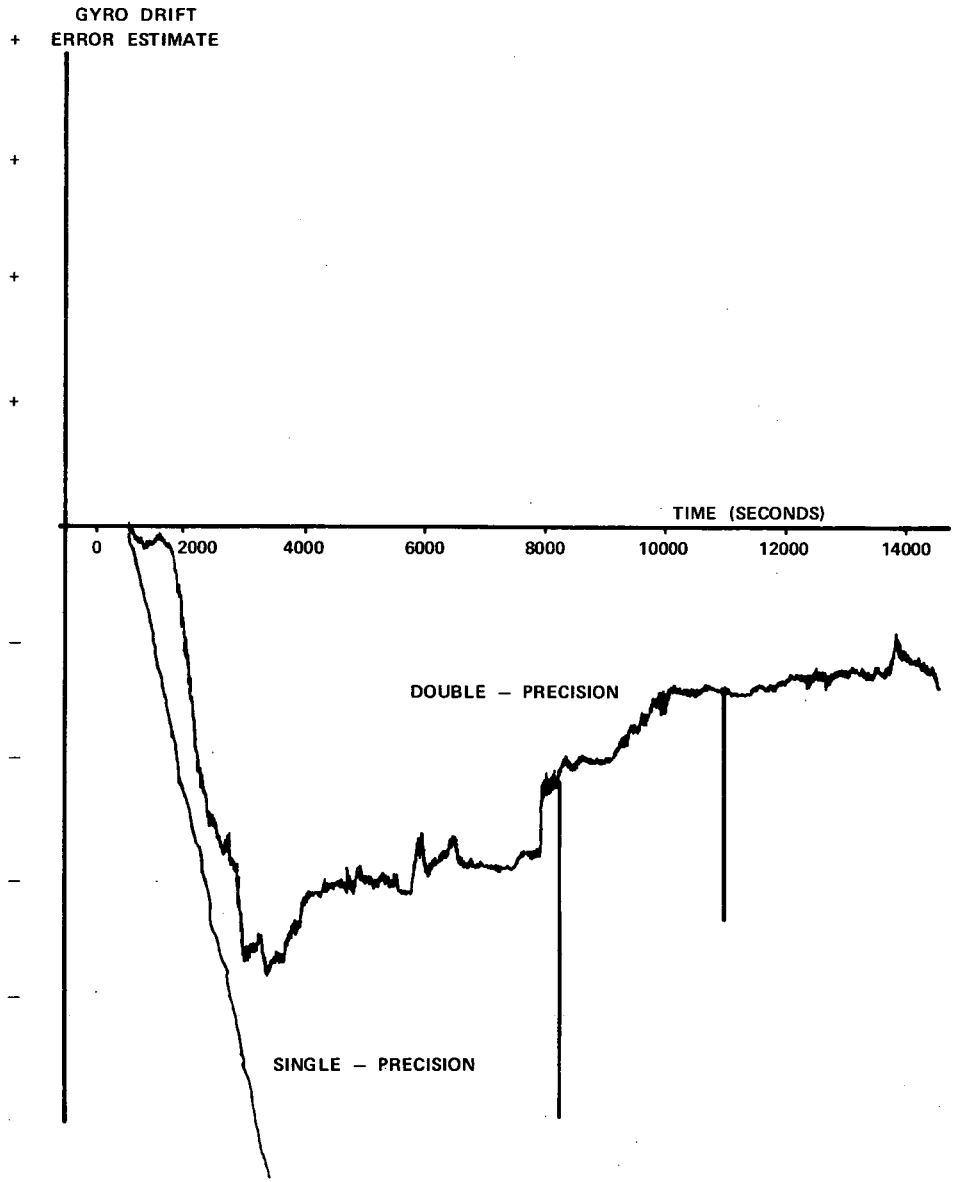
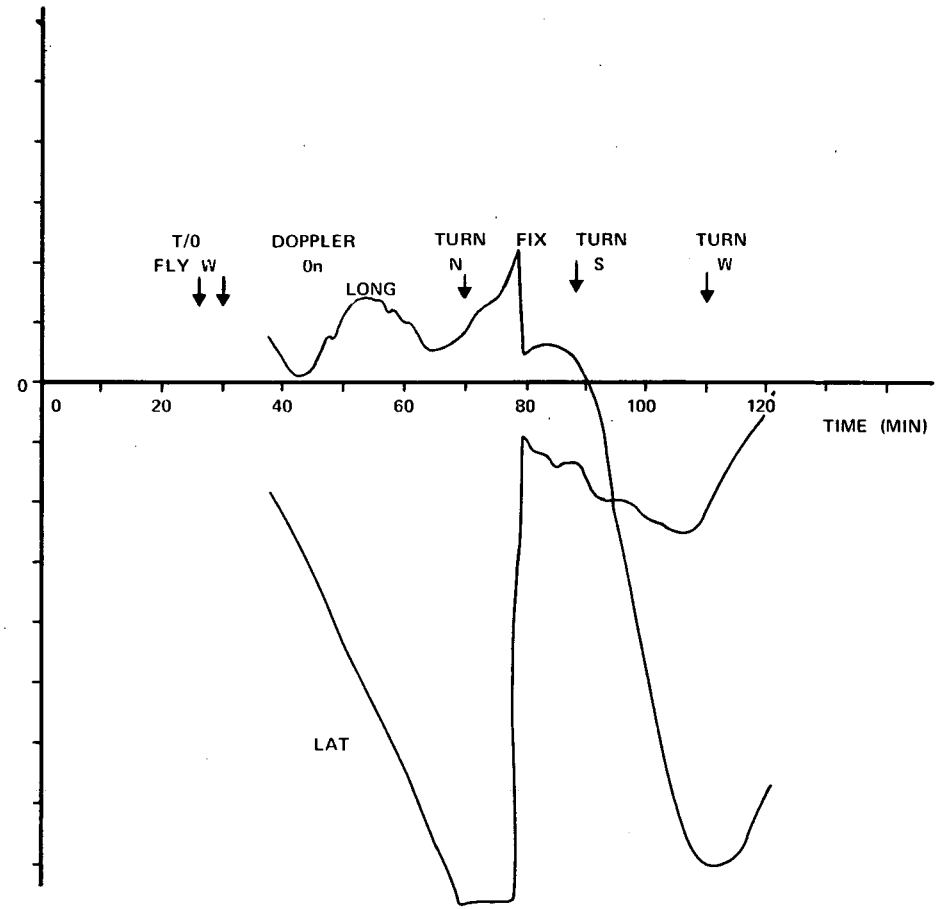
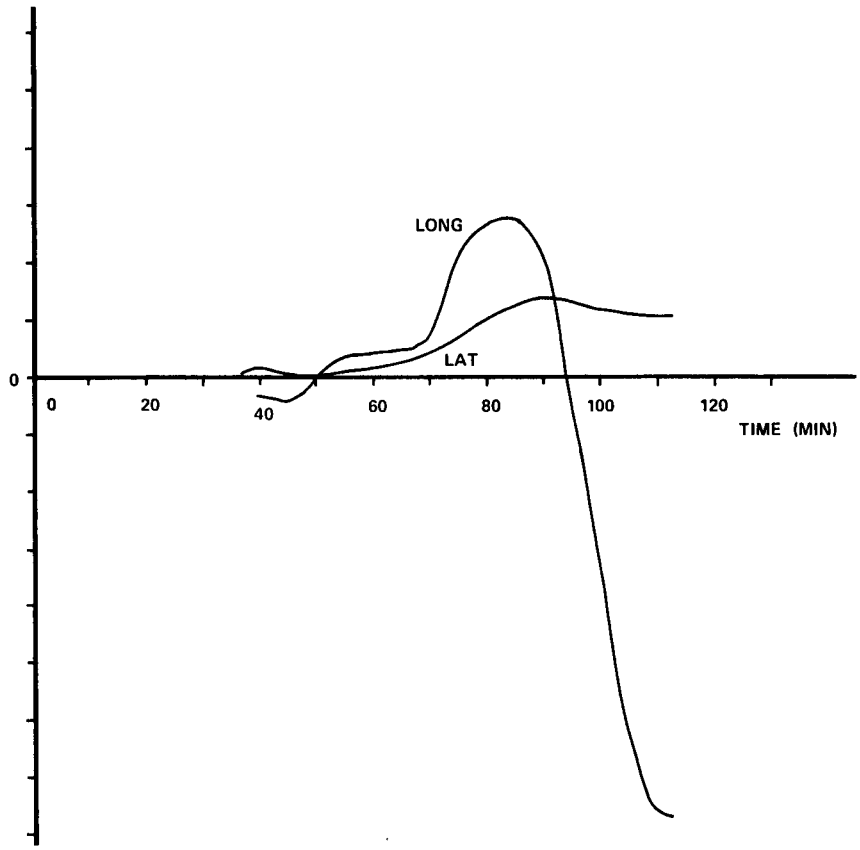


FIG. 5 EFFECT OF INSUFFICIENT WORD LENGTH



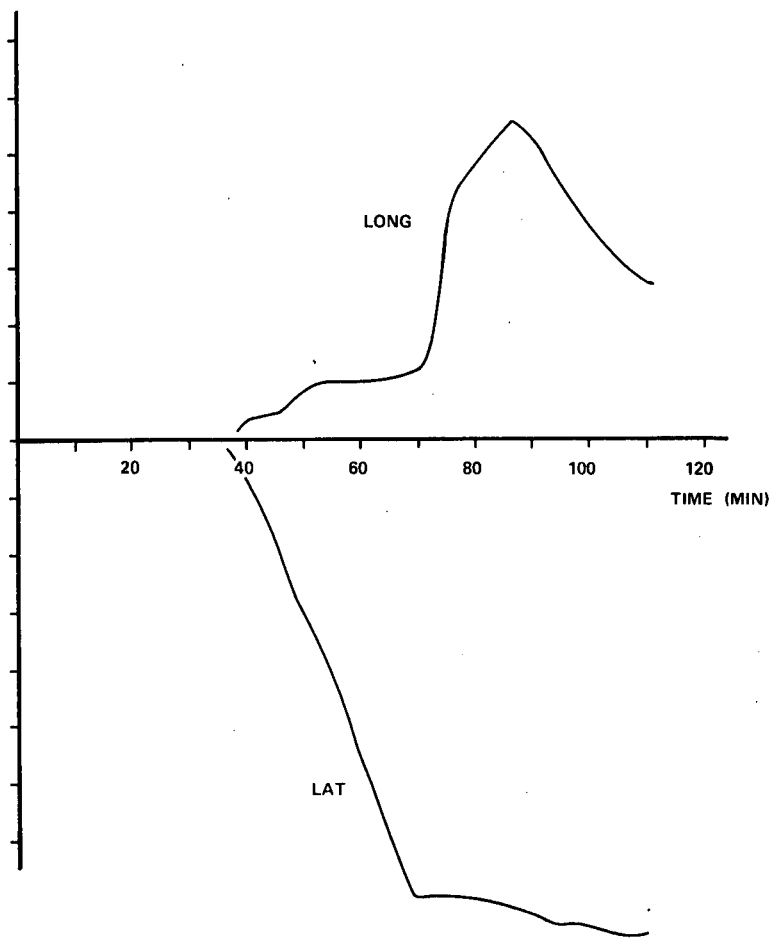
POSITION ERROR OF NAVIGATION SYSTEM WITH FILTER

FIG. 6



RECONSTRUCTION OF FREE INERTIAL POSITION ERROR

FIG. 7



RECONSTRUCTION OF DOPPLER HEADING REFERENCE POSITION ERROR

FIG. 8

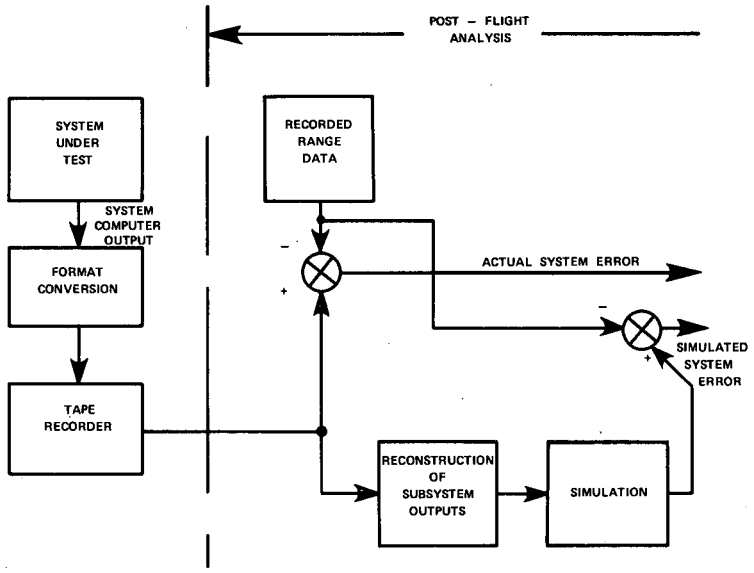


FIG. 9 RECOMMENDED EVALUATION APPROACH

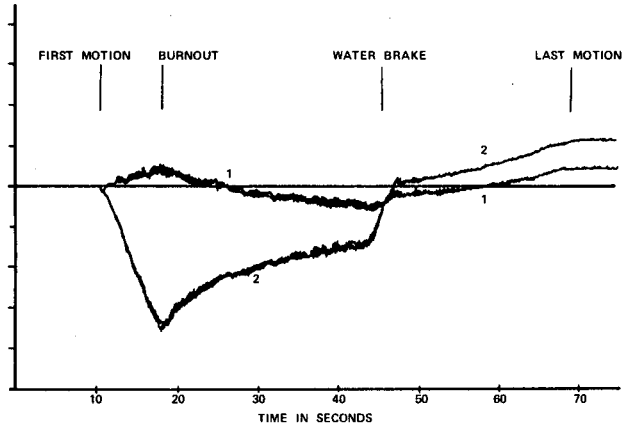


FIG. 10 CROSS TRACK VELOCITY ERROR

1 WITH MASS UNBALANCE COMPENSATION

2 WITHOUT

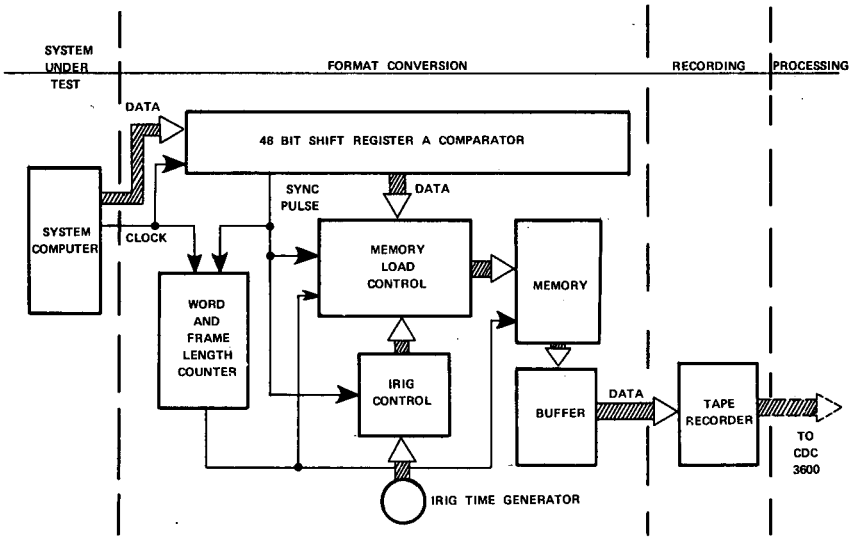


FIG. 11 DATA COLLECTION FLOW CHART

FAULT ISOLATION

IN A

DIGITAL GUIDANCE & CONTROL COMPUTER

BY

DR. DAVID H. BLAUVELT
CHIEF ENGINEER

THE BENDIX CORPORATION
NAVIGATION & CONTROL DIVISION
TETERBORO, NEW JERSEY
U. S. A.

SUMMARY

The capabilities of a general purpose digital computer used in a Guidance and Control application can be used to realize an optimum fault isolation capability for the system. If proper attention is given to the functional partitioning of the computer, self test and self diagnostic programs can be written which will determine that faults have occurred and will isolate them to the replaceable card level. This can be accomplished with virtually no additional flight hardware and a relatively simple test console which allows maintenance personnel to communicate with the computer in question. This can be accomplished with relatively unskilled personnel.

INTRODUCTION

The utilization of a general purpose digital computer to replace an analog computer or a special purpose digital computer in implementing guidance and control computations introduces the possibility of a new approach to the problem of maintaining the system in the field. The nature of a general purpose digital computer is such that the arithmetic and decision making capability necessary to implement the guidance and control function can also be used to perform a maintenance and fault isolation function without any significant increase in computer hardware.

The maintainability philosophy should be designed so that the full advantage of the capabilities of the general purpose digital computer can be achieved. One such approach to the maintainability requirements is as follows. The digital computer operational program includes a small self test program which is designed to exercise most of the hardware in the computer and to provide a specific set of results. If these results are not achieved then a fault indication is given and the computer is removed from the vehicle. The computer is then taken to a test station where it can be tied into a computer operated test console. This console is designed such that an operator can control the operation of the computer under test manually or automatically. At this point a more exhaustive self test program is fed into the computer. This test is designed to completely exercise the computer to confirm the limited self test results and to print out those tests that the computer is failing. The next step in the maintenance procedure is to isolate the failure to a replaceable card assembly. This is first attempted by means of a self diagnostic program. The self diagnostic program is designed such that it sequentially exercises sections of the computer starting with steps which utilize only a very limited amount of hardware. Each succeeding step of the program is then designed to exercise some slightly expanded amount of hardware until all of the computer hardware is exercised. The results of each step in the program are compared to precomputed results and any discrepancy in the comparison would indicate that the increment of hardware first exercised in that program step is not properly operating.

If the self-diagnostic program can not isolate the failure to a single card, then a computer aided test system can be utilized to attempt to isolate the fault to a single card. The approach here is to utilize a test system which can apply selected logic level stimuli to specified test points and measure logic level responses at corresponding test points. Again when the failed card has been isolated it will be replaced and the self test program rerun to insure proper operation of the computer.

Test Console

The test console consists of a control and display panel which enables the operator to communicate manually with the computer under test and a digital computer which is used to automatically control portions of the test and fault isolation procedures.

The control and display panel consists of a multiplexed readout device which is capable of displaying the contents of a 16 bit register in octal format; a selector switch which determines whether the information on the display is the contents of any selected memory location or the contents of any selected computer register; an input keyboard; a set of control buttons which can cause the computer to run, halt, execute a single instruction, execute a single instruction phase, or to run to some preselected location in program memory; and control logic which gives the console access to the I/O data bus, to the memory address bus, and to critical timing and control lines within the computer. This control and display panel enables the operator to hand load short programs directly into memory or into the computer registers, to execute these programs one instruction at a time, to run through selected portions of the programs, to observe the state of any computer register or memory location and in general get a detail status of the computer operation for software or hardware debugging.

The test console also contains its own operating general purpose digital computer. This computer can be used to generate special test stimuli which can be sent to the computer under test through the test connector and in return receive response signals back through the test connector. This can be utilized to test those component failures which cannot be isolated by means of self diagnostic programs. The console computer can also be used to assist in the self diagnostics if desired as well as performing external diagnostics. This computer can also be utilized to analyze the results of the various tests performed and to print out diagnostic reports through a teletype system.

Fault Isolation Approach and Procedure

The fault isolation procedure to be followed in order to isolate failures on replaceable subassemblies is depicted in the flow chart of Figure 1. This procedure is a balanced combination of external and self diagnostics which tests all of the hardware. It provides total external control while allowing testing to be done in dynamic operating modes without dependence on flight operational software. It also makes maximum use of the general-purpose nature of the central processor for self-test and for exercising various parts of the computer.

Fault isolation is based on a 'signal testing' concept, whether external or self diagnostic procedures are employed. This concept involves choosing key signals within the computer for examination (either directly or indirectly), and equating their failure with a printed circuit card. Tests are performed on a pyramidal basis, using tested and verified hardware to test other hardware; this pyramid testing is of primary importance in the deduction of which card is to be replaced.

Ground Rules

In preparation of the diagnostics of fault isolation, the following ground rules were observed.

1. A single failure - It is assumed that only one failure exists in the computer. While the pyramid testing concept attempts to check out all hardware before it is used for testing other hardware, certain areas (particularly in CPU self diagnostics) require that once an error is found, no other errors exist.

2. The computer under test has been operating prior to the failure. Wiring errors would cause improper call outs of failed printed circuit cards.

Computer Self Test Procedures

The limited self test program is a short program, which is interlaced with the guidance and control operational program, and is continuously exercised when the computer is being used. It generally involves solving a fixed known problem and comparing the results with precomputed stored answers. Although a program of this length cannot exercise all possible conditions of the hardware within the computer it can be utilized to detect 95% of the computer failures. Examples of the types of programs which may be in a self test of this type are:

1. A memory sum check on the fixed program to insure that program memory has not been altered.

2. A memory operability test which exercises all memory locations verifies their proper operation.

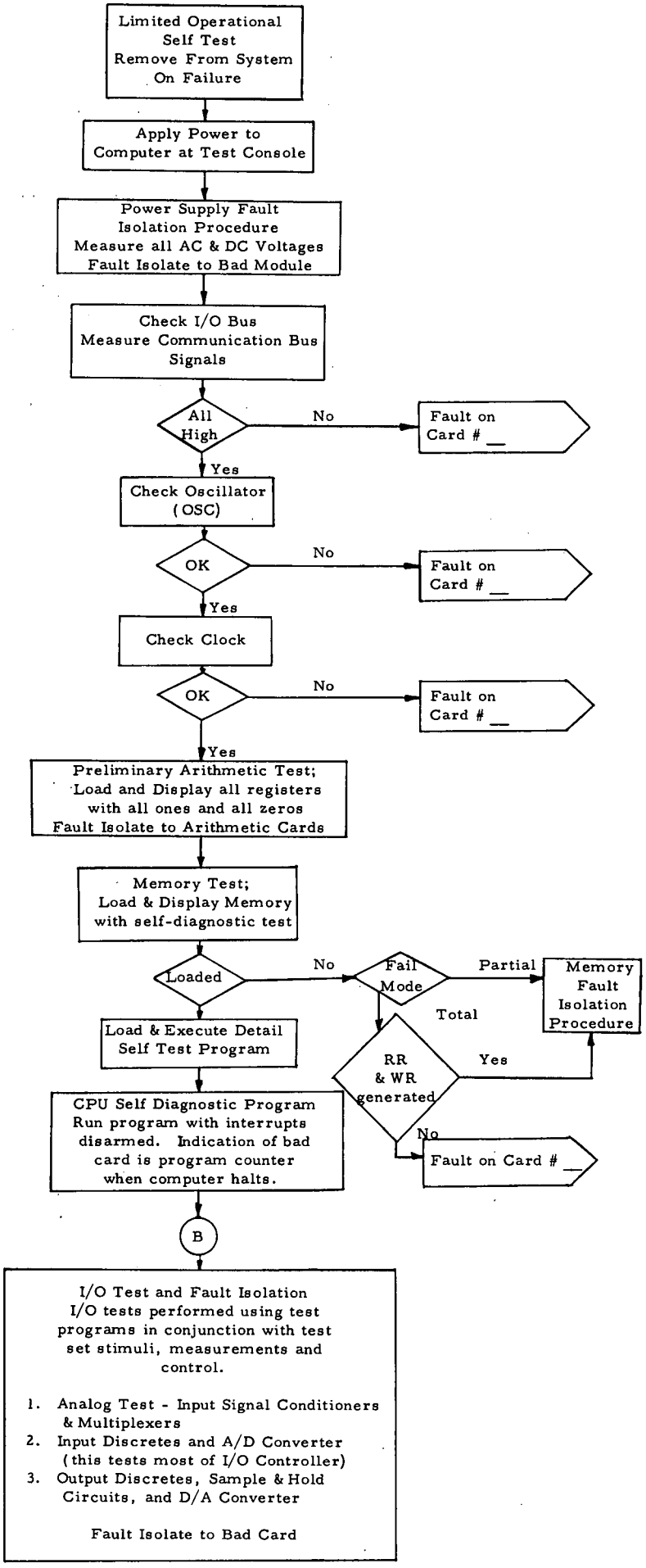


Figure 1
Fault Isolation Overall Flow Chart

3. An arithmetic and control test which exercises all computer instructions in a fixed solution problem.

4. A tail biting D-A, A-D converter test to verify operation of the major part of the I/O section.

After the limited self test program indicates a computer failure the computer is removed from the system and taken to the test area. The computer is tied into the test console at this point. The function of the test console is to enable the operator to take control over the operation of the computer. Before attempting to operate the computer with the console, however, certain tests are made to insure that there have been no catastrophic failures. All power supply voltages, the oscillator and the computer clock are checked. The computer data input lines are also checked to insure they are operational.

After these preliminary steps are taken the next step is to determine if the test console can successfully communicate with the computer under test. The procedure is to load each of the operating registers of the computer with an all zero pattern and then an all ones pattern. After loading, each register should be displayed on the test console. Any discrepancy between the data loaded into the registers and the data displayed on the test console indicates a failure in the arithmetic section or the control section of the computer. Knowledge of the partitioning of the computer onto printed circuit cards and the combination of specific discrepancies in the display can be logically analyzed to determine which card has failed. It should be noted that care in partitioning the computer along functional lines in the design phase can greatly simplify the fault isolation process at this point. If no discrepancies are noted then proper operation of the I/O bus, each of the computer registers, and the adder can be assumed, and this fact can be used to form the basis for further tests. The next test is to set the memory address register to some preselected value and to load a data word in that memory location by means of the console. Display the memory location to insure that the memory was properly loaded. Manually set a load accumulator instruction into a second memory location and manually execute that instruction. Change the load instruction to a store instruction and execute this instruction. Change the store instruction to an add instruction and execute this instruction. If all these procedures fail the read and write commands from the central processor should be checked. If they are properly generated the fault is in the memory. If only certain of these steps fail the fault can be on one of the central processor cards, or in the memory and the exact status can be deduced from the specific failures.

The next step is to insure that the computer memory is completely operational. This is accomplished by loading a memory test program from a tape reader utilizing a hard wired bootstrap loader. The loader can also be utilized to verify that the proper information got into memory by reading the tape a second time and comparing the inputs from the tape with the information stored in memory on the first pass. If this validation fails one of the instructions used in the bootstrap loader may be inoperative. The bootstrap program should then be stepped through one step at a time and all affected registers should be examined to determine if that instruction was properly executed. This is continued until the faulty instruction is located. The nature of the fault can then be used to isolate the faulty card.

If the memory test program is properly loaded it is then run. This program sequentially goes to each memory location, reads out the data and transfers it to a temporary location in memory. Next an all zero pattern is stored at the test location which is then read out and compared to the accumulator. This is followed by an all ones pattern, an alternate one zero pattern, an alternate zero one pattern and the location address. If any of the comparisons fail the computer is halted indicating a fault at that location. If the comparisons all pass,the data originally located at that address is restored and the program moves on to the next memory address where the process is repeated. The program will continue to cycle through all memory locations until the program is halted by the operator. At this point proper memory operation is assured.

After memory operation is verified a detailed self-test is loaded into memory and executed. This test exhaustively exercises all of the hardware and all the capabilities built into the machine. This test is used to confirm the original failure indication.

The next step is to load and run the central processor self-diagnostic program. This program is designed to exercise all the functions in the central processor in a sequential manner such that with each step of the program some additional section of hardware is verified to be operative. The self-diagnostic programs assume that the following functions have been previously verified to be operating properly: memory; memory buffer register; adder; and the load accumulator, store accumulator, add, and halt instructions. The general philosophy utilized in writing the self-diagnostic is as follows.

The preliminary manual exercising of the computer through the console established that all of the registers were operational and that data could be loaded into them. In addition the add, load, and store instructions were established as being operational. This means that those control

signals and gates needed to execute these three instructions are properly operating. Using these known facts an instruction utilizing that hardware already verified and one untested control signal can be exercised. If it operates properly then the new control signal and the associated gating are verified. If the instruction does not operate properly then the failure must be isolated to either the control signal or to the gating. This can be accomplished by choosing another instruction which utilizes the questioned control signal. If it operates properly the fault is in the gating and is isolated. If the second instruction does not operate properly then the control signal is not operating and the fault is isolated. This process can be repeated until the fault is isolated or until all control signals and all instructions have been determined to be operational. If no fault is isolated at this point the self test failure print out should either be given to a troubleshooter and he should attempt to isolate the fault manually, or the test console computer should be used to card test one or more selected cards as suggested by the nature of the self test failure print out.

If the self test or the self-diagnostic program determines that there is a fault but is not able to isolate the fault to a single card, the test console and its computer can then be used to apply more specific tests to each of the potentially failed cards. This is accomplished by utilizing the test console computer to supply selected stimuli in a sequential fashion to the suspected cards by means of the test connector and measuring responses at corresponding points also brought out to the test connector. The test console computer is then used to evaluate the responses to the stimuli and to determine if the suspected card is operating properly.

This same test console computer can also readily be used as a card tester if a suitable adapter is provided. After the fault isolation procedure has located the failed card that card can then be removed from the computer and tested by itself, its failure confirmed, and the actual failed integrated circuit located exactly or narrowed down to a few possible integrated circuits which can then be replaced in order to restore the card and the computer to its operational status.

The detailed self-test program can then be rerun in order to confirm the computer has been restored to proper operational status.

The fault isolation of the I/O section of the computer varies because the I/O requirements for each specific requirement will differ. However, most systems will contain a multiplexed analog to digital converter, a multiplexed digital to analog converter, a digital data link and some means for sending and receiving discretes.

The proper design of the A-D and D-A converter and multiplexer can greatly simplify the fault isolation of this hardware. If this hardware is all packaged on a single card and if one of the multiplexed output channels is connected to one of the multiplexed input channels, then a diagnostic program can be written which will generate a digital output from the central processor which is converted to an analog voltage by the D-A converter. This analog voltage is then multiplexed after passive scaling to the input multiplexor of the A-D converter, where it is converted back into a digital number. This digital number is then inputted into the central processor where the result is compared to the known correct value. This process can be repeated for a number of different voltages to determine the proper operation of the converters or to isolate the failure to this card.

The fault isolation of the digital input output link can also be simplified by having this entire function on a single card. The test can be implemented by using the test console computer to send a simulated message to the computer under test and then having this same message retransmitted back to the test console computer. By properly choosing the test message all the associated hardware can be exercised and either validated or fault isolated.

This same general approach can also be utilized to fault isolate the discrete input output section of the computer provided that all the associated hardware is packaged on the same card.

Any other I/O functions which may be in a specific computer application can generally be fault isolated in a manner similar to that described above as long as a functional approach is taken in the partitioning of the cards in the I/O section of the computer.

SPACEBORNE COMPUTER FOR ATTITUDE CONTROL AND DATA HANDLING

by

P.van Otterloo
Philips Research Laboratories
Eindhoven (Netherlands)

SUMMARY

In the ANS (Astronomical Netherlands Satellite) which is planned to be launched in 1974, an on board computer is foreseen. This computer will consist of two parts, the computer proper and the data-memory.

The computer, which will be reprogrammable by ground command, will serve as a direct digital controller for the attitude control of the satellite and as a data processor for reduction of the experiment data.

The data memory serves as a buffer for storage of separate experiment and housekeeping data for a period of 12 hours. Both the computer and the data memory have to be designed to serve weight and power limitation.

INTRODUCTION

The computer to be discussed here is designed as a subsystem of the ANS (astronomical netherlands satellite). As long as no acronym has been found it shall be referred to as OBC (on board computer) which is the subsystem name. The ANS is a semi polar orbit (500 km) satellite with high pointing accuracy ($\pm$ 1') intedded to perform astronomical UV and X ray measurements. It is a joint U.S./Netherlands effort to be launched in 1974 by a SCOUT launcher. Due to the limited performance of this launcher low weight and power consumption are stringent requirements. Because the satellites orbit is polar, ground contact with a station at the latitude of the Netherlands is possible only once every 12 hours during 7 minutes. This means that experiment data have to be stored on board as well as commands stating the measurements to be performed in at least one 12 hour period. So in the 7 minutes contact time not only all experiment data have to be extracted from the satellite, but also new commands have to be transmitted for the next 12 hour period. A digital memory for which a core memory has been chosen is unavoidable because of the inherent unreliability of taperecorders as well as the unrealistic record/play back ratio required.
Telemetry subsystems are working digitally too, this favouring digital ionboard information storage. Experiment data are all of a digital nature and need some reworking in order to limit storage space and transmission time. This requires a digital computer.
The attitude control subsystem required calculations which for various reasons (integration, logical decisions) can only be performed by a digital computer. An analog system built to calculate these signals would not only be very complicated, heavy and highly power consumptive but would also frequently need adjustment. The above mentioned requirements still do not express a clear preference for fixed program or stored program machines. The initially performed work went in the direction of a fixed program machine.
Soon however, it became apparent that a lot of changes can be expected in the program up to the time that the flightmodel is ready for launch. Also different formats are required for on board storage of data while different modes of operation made the data handling of the experiments more and more complicated. This led to the conclusion that the computer should have a stored program. This automatically brings up the question whether program changes were to be done only on the ground or also in orbit. Since the operation of the satellite depends on the possibility of transmitting 5000 bits in one or two minutes to the satellite every 12 hours (star measurements program) and reading about 400 000 bits out of it, it was decided that sending up and verifying a part of the program was not more difficult and therefore quite well realizable.
The limited space on the satellite made impossible to build the complete subsystem in one box. In order to put the interface at a convenient spot, it was decided to split the memory in two parts. The biggest one will be used only for data storage awaiting later transmission to the ground. The capacity will be about 400 000 bits. The rest, 64 k bits, will share its box with the central processor and the 1/0 unit and serve as computer program space, working space and command storage. Both boxes shall weigh no more than 4 kg and consume no more than 4 Watt of power each.

COMPUTER

In fig. -1 a schematic diagram of the computer is presented. The following parts can be found
-arithmetical unit
-control unit
-memory
-input/output unit

These units will be detailed further below. This will however not lead to a complete description as the design is not yet complete. Attention however is asked for the program given in the next section on computer software, which should give an impression of the capabilities of the subsystem.
Speed can be relatively low since the program repetition time is 1 sec. dictated by the sampling time of the attitude control subsystem for which the computer performs the calculations.

ARITHMETICAL UNIT

This is the unit where the actual calculations take place. The only possible calculations are adding, multiplication by powers of 2 (:shifting) and inverting. All other calculations have to be programmed.
It has not been decided yet but the hardware calculations will probably be performed in series. This decision will depend on the length of the program and its execution time.

CONTROL UNIT

In the control unit the instructions are decoded and transformed in machine control pulses while the addresses are, depending on the instruction, routed to the corresponding registers. As will be discussed later, one word of the program has 16 bits.
Of these 6 are used to indicate the instruction, allowing for 64 different instructions,and 10 which can have different meaning, depending on the instruction to which they are coupled. They can be:
- a number to be added to or to replace another number (address modification etc.)
- an address in the memory
- a register

The 10 bits available are not sufficient to address the whole memory, which will have 4k words. The additional bits are again generated in this control unit.
Finally this is also the unit where the clock is built in. This has to be the most reliable part of the OBC since it also delivers clock pulses to the rest of the satellite and since it makes sense to keep supplying these signals once the complete OBC failed.

INPUT/OUTPUT UNIT

This unit consists of an address register with decoding used to open the gates to the requested input or output as well as some logic to feed shift and transfer pulses to these inputs or outputs. A standard computer output (SCO) interface and a standard computer input interface (SCI) has been specified (fig.2) All units wishing to communicate with the OBC can do so via these interfaces. In order to limit the number of interface units and wiring, subcommutation of the experiments is performed. The OBC supplies them via SCO with an adress, allows for transfer from the register indicated by the adress to the SCI and then reads the SCI.

MEMORY

A 4k 16 bits memory module indentical to the blocks of the data memory has been chosen, as a first order approximation, at the same time hoping to reduce the number of spare parts and easeing for suppliers the qualification of items. It is thought to be divided into 4 parts of 1k each of which 2 are not used and saved for redundancy. Of the other two parts 1000 words are used to store the program and the remaining 1000 words will serve as a working space for the computer and at the same time as storage space for the commands.

The two redundant parts normally are not used. The redundant program part will be filled with an emergency program for a start and later on, when the program in use appears to be reliable, can be used to put new programs in . The advantage is that during transmission of the new program, which takes several minutes, the old one is still available for attitude control. The redundant working space cannot be used to store more program as long as there is no emergency requiring its use, because upon occurrence of this failure in the normal working immediate switchover must be possible. This means that defined memory is required to be available.

COMPUTER SOFTWARE

Tasks of the program

The tasks of the computer program can be divided into several parts:

a) The attitude sensor data are gathered, processed according to the observation program, and the resulting set-points are given to the attitude actuators, i.c. the reaction wheels: also attitude data are stored in the data memory.

b) The experiment data are gathered, processed and stored in the data memory.

c) Commands are transmitted to the experiments according to the observation program and the status of the experiments.

d) A selection is made from the information available at the housekeeping encoder and this selection is transmitted to the data memory.

e) A selection is made from the experiment and attitude data and transmitted to the housekeeping encoder.

f) A series of clock pulses is generated and distributed.

In order to perform this task a very limited number of standard instructions will be made available.
These instructions serve as building blocks for the program.

Operations

The program consists of a number of instructions. Each instruction consists of a 16 bit word, stored at a certain location of the computer memory.
Of these 16 bits 6 are the instruction part: these refer to a certain operation to be performed by the computer. The other 10 bits are the address part and indicate a certain location in the memory. In the first part of each computer operation the content of the operation counter is increased by 1. The content then is transferred to the memory address register. (see fig.-1) The contents of the addressed is transferred to the M register and from these to the instruction register. The second half consists of the execution of the instruction called from the memory in the first half of the operation.

Instructions

The instructions foreseen in this stage of the project are:

Read X into A (IA/X)
The word at address X is read from the memory into M, from where it is transferred to A and at the same time rewritten in the memory at the location X.

Read X into A, negative (IA-/X)
The same, but the sign is reversed before transfer to A.

Read X into A, destructive (IAD/X)
The same as IA/X but without the rewrite operation.

Read X into M, (IM/X)
The word at address X is read into M, and rewritten into location X. After that it is added to the word in A. The result appears in D. Storage in M cannot be used as such in the program, because register M is used in the next operation for transfer of the instruction word.

Read X into M, negative (IM-/X)

Read X into M, destructive (IMD/X)

Read X into M, negative, destructiee (IMD-/X)

Write D into X (OD/X)
The word in the D-register is transferrred to M and from these written into the memory at location X.

Write D into X, negative (OD-/X)
The same, but with sign reversed.

Jump to L(TO/L)
In this case the address part of the instruction contains the number L-1. This is transferred to the operation counter from the instruction register. In the next operation the program proceeds at instruction number L.

Set flip-flop if D positive (SIFD)
This instruction, which has no address part, sets a flip-flop if the content of the D register is positive.

Jump to L, if the flip-flop is set (IFTO/L)
If the flip-flop is set, the operation is the same as in TO/L, after which the flip-flop is reset. Otherwise nothing is performed, so that the program proceeds with the next step.

Shift N steps from register P to register Q (SHN/PQ)
The instruction part here consists of 2 bits indicating that the instruction is of the SHN type, and 4 bits giving the numbers from 1 to 16. The address part consists of two 5 bit words, each of which indicates one of up to 32 possible registers, either in the computer arithmetic and control unit or in the input-output unit. The effect of this instruction is to shift the indicated number of bits from P to Q. P and Q may be identical: Q may be zero, in this case the information is shifted only out of P.

Statements

In order to ease the task of programming the computer a simplified FORTRAN language will be designed, where each FORTRAN-statement consists of a small number of instructions. This makes it possible to translate the programs used for simulation in a straight forward way into machine language. The following statements have been considered.

X = Y transfer to X of Y		IA	O
		IM	Y
		OD	X
X = Y + Z		IA	Z
		IM	Y
		OD	X
X = Y - Z		IA-	Z
		IM	Y
		OD	X
GOTO L		TO	L
IF (X) L	if X>0 Goto L	IA	O
		IM	X
	if X<0 proceed with next statement	SI FD	
		IF TO	L
IF (X-Y)L		IA-	Y
		IM	X
		SI FD	
		IF TO	L
$Y = X * 2^N$		IA	X
		S HN	A, O
		IM	O
		OD	Y
READ (P)X		SH16	P, D
		OD	X
WRITE (P)X		IA	X
		SH16	A, P
	subroutine calling sequence :		
CALL K(A, B)		IA	O
		IM	A
		OD	P
		IM	B
		OD	Q
		SH16	OC , D
		OD	MAIN
		TO	K

```
SUBROUTINE K (P,Q)    K:        --  |  --
                                --  |  --

END                             IA  |  2
                                IM  |  MAIN
                              SH16  |  D, OC

proceed with calling program    IA  |  O
                                IM  |  P
                                OD  |  A
                                IM  |  Q
                                OD  |  B
```

In the location MAIN the content of the operation counter in the calling program is stored. After execution of the subroutine this instruction number is retrieved and increased by two for compensation of the other instruction between the storage and retrieval of MAIN.
The shifting of information fromthe main program to the subroutine and back can be diminished in some cases by declaring a COMMON storage location.

As a small example we give a program which calculates X + Y if X - Y > 0 and X + Z if X - Y < 0 and stores the result in Z.

```
          IF(X- Y)L                     IA  |  X
                                        IM- |  Y
                                      SIFD  |
                                      IFTO  |  L

          Z = X + Z                    IMD  |  Z
                                        OD  |  Z

          GOTO N                        TO  |  N

L         Z = X + Y        L:          IM  |  Y

N                          N:          OD  |  Z
```

Sample program

To give an indication of the way in which the computer is programmed for one of its tasks a sample program of the attitude control part is given below: A schematic diagram of it is presented in fig. 3.

```
AC        IF(SLEWo) SL
          DPL=PLH1 - PLH2          guide star distance
          DPL=DPL - SGS            SGS is desired distance
          IF(DPL - 5) ERR
          IF(-DPL-5)ERR            distance should agree between
                                   ± 5 elements
          IF(PLUM)PL
          PLUM = 1
          GOTO HOR                 Plumbicon mode is initiated only
                                   after 2 recognitions
ERR       PLUM=O
HOR       FSY=FSY - 100            zero correction for 4 sensors
          FY=FSY - SFSY            subtraction of setpoint gives error
          HS=HS1 +HS2
          HS=HS - SHS
          FZ=5 * HS                E_z = 10 * [(19 + 92)/2] - 10 * [(2* 90)/2]
          GOTO LOOP
PL        FY=PLV - 4
          FZ=PLH1 - SPLH1
LOOP      FX=FSX - 100
          CALL FCO(FX, FXV, INTX)  fine control subroutine
          WRITE RW1, V             transmission to X-reaction wheel
          CALL FCO (FY, FYV, INTY)
          WRITE RW2, V
          CALL FCO (FZ, FZV, INTZ)
          WRITE RW3, V
          GOTO EXP                 EXP is start of experiment data handling
```

```
SLS     SLEWo=1
        AX=0                          A indicates the accelerating phase of the
                                      slew procedure
        AY=0
        AZ=0
        STARTSL=0
SL      FX=FXS-100
        FY=FSY-100
        FY=FSY-SFSY
        FZ=HS1 + HS2
        FZ=FS - SHS                   because control in on-off, scale is irrelevant
        IF(STARTSL)SLL
        FXo, FX
        FYo=FY
        FZo=FZ
        TXo=T
        TYo=T
        TZo=T
        STARTSL=1
SLL     CALL SLEW (AX,FX,FXo, TXo, TSX)
        WRITE RW1, V
        CALL SLEW (AY, FY, FYo, TYo, TSY)
        WRITE RW2, V
        CALL SLEW (AZ, FZ, FZo, TZo, TSZ)
        WRITE RW3, V
        IF(FX - 30)      EXP
        IF(-FX - 30)     EXP
        IF(FY - 30)      EXP
        IF(-FY - 30)     EXP
        IF(FZ - 6)       EXP
        IF(-FZ - 6)      EXP
        SLEWo=0                       If all errors are in abs. value  <0.3°
                                      fine pointing
        INTX=0                        is initiated.
        INTY=0
        INTZ=0
EXP
        SUBROUTINE FCO (F, FV, INT)
        COMMON V
        IF(F-50) SLS
        IF(-F-50)SLS                  If any error is in abs. value  >0.5°
                                      slew is initiated.
        INT=INT + F
        CALLSAT (INT)
        DIF=F-FV
        FV=F
        DIF-DIF * 16
        P=F * 8
        V=P + DIF
        V=V + INT
        CALLISAT (V)
        END
        SUBROUTINE SAT (X)
        IF (X-1000)SAT +
        IF (-X-1000)SAT -
        GOTO ENDSAT
SAT+    X=1000
        GOTO ENDSAT
SAT-    X= - 1000
ENDSAT END

        SUBROUTINE SLEW (A, F, Fo, To, TS)
        COMMON V, T
        IF(A) DEC                     decelerating phase
        IF(Fo) POS
        IF(F-Fo) INC
        Fo=F
        To=T
        GOTO MIN
```

```
INC     F=F + F
        IF(F - Fo) MIN
        TS=T - To
        TS=TS+TS
        A=1
        GOTO PLUS
POS     IF(F-Fo)INC2
        Fo=F
        To=T
        GOTO PLUS
INC2    F=F + F
        IF(F-Fo) PLUS
        TS=T - To
        TS= TS + TS
        GOTO MIN
DEC     IF(Fo) POS 1
        TS1=TS + To
        IF(TS1 - T) PLUS
ENDSL   Fo=F
        To=T
        A=0
        IF(Fo)MIN
        GOTO PLUS
POS1    TSt=TS + To
        IF(TS1 - T) MIN
        GOTO ENDSL
PLUS    V = 1000
        GOTO LAST
MIN     V =-1000
LAST    END
```

The length of this part of the program is about 120 FORTRAN - statements so that in the real program about 350 - 500 lines will be required.

Overall program

The overall program is not yet completed but it will consist of the following blocks:

First the time is updated by the statement T = T + 1.
Next a test is performed whether new measurement setpoints are required by IF(TM-T)AC. As long as T TM the program proceeds with attitude control; otherwise new setpoints are transferred from the observation program part of the memory to their working addresses (SHS, SPLH, TM etc.). The stored commands are transmitted to the command control unit. The following part is attitude control, described above. At statement EXP the processing of experiment data starts, consisting e.g. of fixed-to-floating conversion, integration etc. The last part of the program controls the transfer of experiment and attitude data to the data memory, as well as transfer of these data to the encoder and of housekeeping data from the encoder to the computer and from there to the data memory. This transfer is controlled by the statements:

WRITE DM(N), X N= 1 to 6.
The word at address X is transferred to the N-th block of the data memory.

WRITE ENCADD, N N= 1 to 16.
This statement causes the N-th 8 bit word, available each second at the encoder to be stored in the encoder output buffer register.

READ ENC X.
The computer reads the word at the output buffer of the encoder and stores it at address X.

WRITE ENC X.
The last 8 bits of the world X are transferred to the encoder input register.

DATA MEMORY

In the first part of this chapter attention will be paid mainly to the type of memory selected while the second half will be used to describe the proposed design.

Choice of memory type

In the original concept it was envisaged to have a tape recorder on board to store the information flow from experiments to housekeeping. The expected number of bits was 2 or 3.10^6. Due to a reduction in the number of experiments as well as in the number of bits per experiment this was at start of the feasibility study reduced by a factor of 10. During this study an experiment was added and the housekeeping as well as experiment data increasesd again.

Instead of giving every experiment its maximum storage space and adding these together to get an overall storage space requirement, another policy was followed. The computer is being used now not only to handle the data but also to select data for transmission to the memory. This selection is made by a computer program that easily can be changed to adopt the system to various environments, requirements and situations by reading in another one. This works of course only under the assumption that maximum storage space is asked by only user at a time.
For example during dynamic attitude control measurements nearly all storage space is used for storing the sensor data so no experiments and very little housekeeping data can be written in the memory, while during pulse and measurements the x-ray experiments will fill all available memory space in 90 minutes.
In this way the memory size needed is in the order of magnitude of 400 K bits.

Because of this reduction in the number of bits the tape recorder was not the only solution anymore.
It was actually immediately rejected because the required record to play-back ratio was very unfavourable. The record time being 12 hours and the play-back time of a few minutes brought this ratio in the order of 1:200. A recorder with this feature is not commercially available and would have to be developed. This, added to the fact that tape recorders for space application are still rather unreliable, made that such a recorder as a data memory was rejected.

The next choice to be made was that between dynamic and static memories. With dynamic is meant that the information stored should be moving continuously in order not to loose it, for example dynamic shift registers and delay lines; while static shift registers, core stores etc. do not need to pump information around in order to store it.
Therefore they are called static memories. This immediately reveals a weak point of the dynamic memories. Information being pumped around is more susceptable to noise etc. than static information. Since furthermore the power consumption as well as the reliability of dynamic shift registers is unfavourable, while little was known about delay-lines, of which the weight could have been critical, it is decided not to use dynamic memories.
Another disadvantage, which these memories share with some static registers, is that they will loose all information due to a temporay power shut down. This power shut down may be voluntarily (to save power during acquisition and eclipse) or accidentally (wrong commands or noise) but will certainly occur, without a possibility for immediate reprogramming. And a loss of program means loss of attitude control and of data handling which is a near to catastrophic failure and cannot be tolerated. So no active elements are allowed for storage.

Consequently a choice had to be made out of the various types of magnetic memories which are the only non active memories considered. Exotic circuitry cannot be used due to unknown reliability and lifetime, even plated wire memories are not yet fully space qualified, so a preference exists for magnetic core memories. Non destructive read out is not used because it would add too much weight to the memory.

Description of proposed design

The most stringent requirements to be fulfilled are:
low power and weight, high reliability.

Since the average write and read repetition is low, the only noticeable dissipation is the stand-by power of the logic. For this purpose low power logic with an acceptabel speed has been used. (The speed is determined by the required rise time of the read pulse). To decrease the power consumption of the read amplifiers, they are fed by a supply pulse just overlapping the read pulse. In this way a device has been made with a low power consumption.

Since the power consumption in the stack can be neglected, no precautions have to be taken for cooling. This means that the heat conducting strips can be cancelled and the weight of the stack is decreased. Since the stack determines for the most part the weight, the weight of the memory can be made very low by using low weight material for the remaining part of the stack.

Yet another requirement is a restriction in size in order to have everything fit in a box with outside dimensions of 13 x 16 cm. This can probably only be accomplished with the core size of 20 mils. The configuration chosen is shown in fig. 4-. Three printed circuit boards carry one block. The first two boards carry on both sides 4 matrices of 4 K bits, while on the third one the selection diodes are mounted.

As the whole information can be written in series, the address register is a simple counter. A high realiability has been achieved by making a good and simple design and using reliable components as well as by allowing for gentle degradation. The whole memory consists of 6 separate memory blocks.
In the case of a failure in one of the blocks that particular block can be rejected and skipped.
In this way not only high reliability but also high flexibility can be obtained. At the same time the number of spare parts can be redueed. A similar 4 K 16 bits memory unit will be used in the computer to serve as working, program and command memory.
The 6 blocks required to get the 384 K storage space are fully independent but in order to keep the interfaces and cabling to an absolute minimum, they are not connected to the computer. In stead a separate switching mechanism is used to route the information to the proper block. This device can be used to skip a damaged block but may also be programmed from the computer to send for instance housekeeping and experiment data to different blocks to speed up ground handling of the data. However, it will be a fairly simple and straight forward piece of hardware and is therefore not described here.

In fig.5 a block diagram of one memory block is shown. The information that must be stored in the memory is applied to shiftregister I. The register can store 16 bits. The number of bits of one set of information can be 8 or 16. In the case of 8 bits a write pulse is given after two times 8 bits. The applied information is accompanied by an equal number of trigger pulses applied to the trigger input of the register. At the same time a pulse with a duration time of 8 bits is applied to the timing. In the case of a read pulse the tele-command delivers 16 trigger pulses to shift the shiftregister. After the 16 trigger pulses all the flip-flops of the shiftregister are reset. This means that only the set-inputs are connected with the outputs of the read amplifiers.

Timing

At the end of the pulse accompaning the first 8 information pulses a flip-flop is set and at the end of a second pulse the flip-flop is reset again.

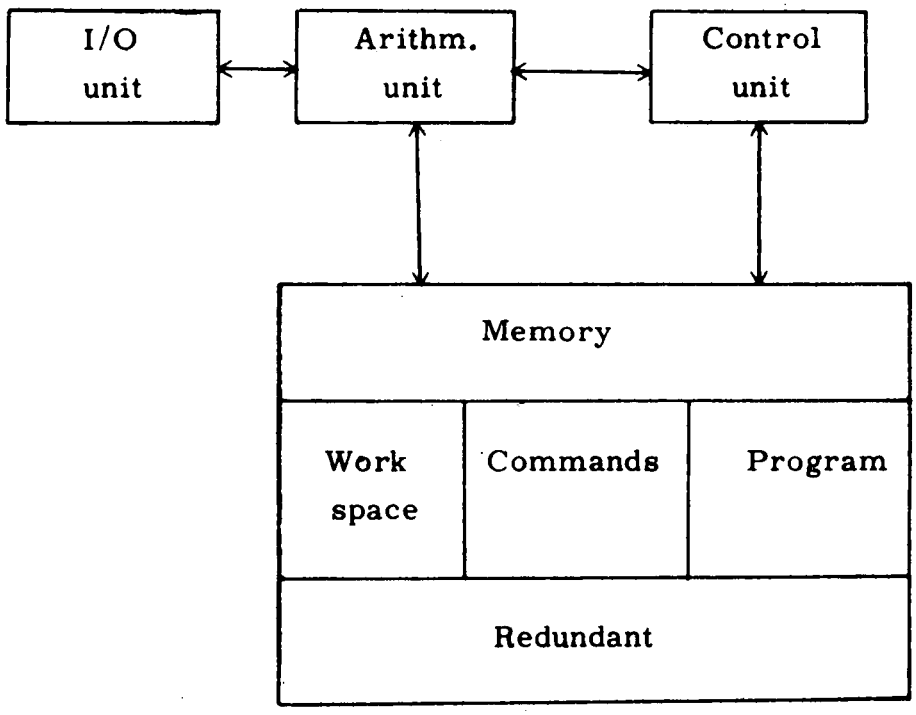


Fig. 1 Schematic diagram of the computer

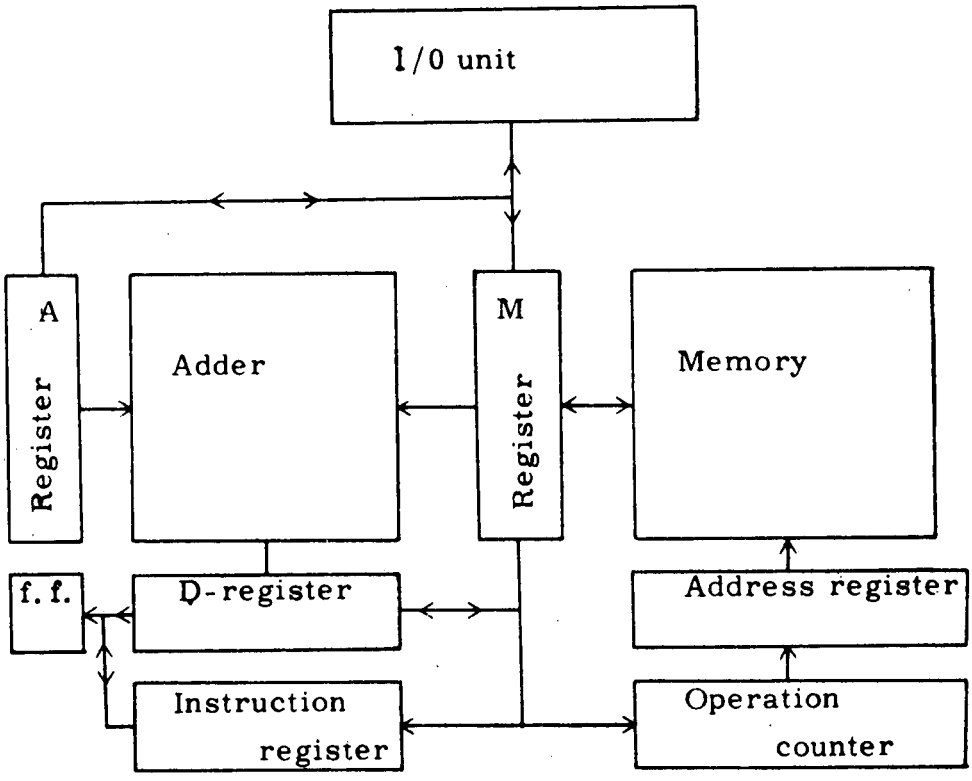


Fig. 2

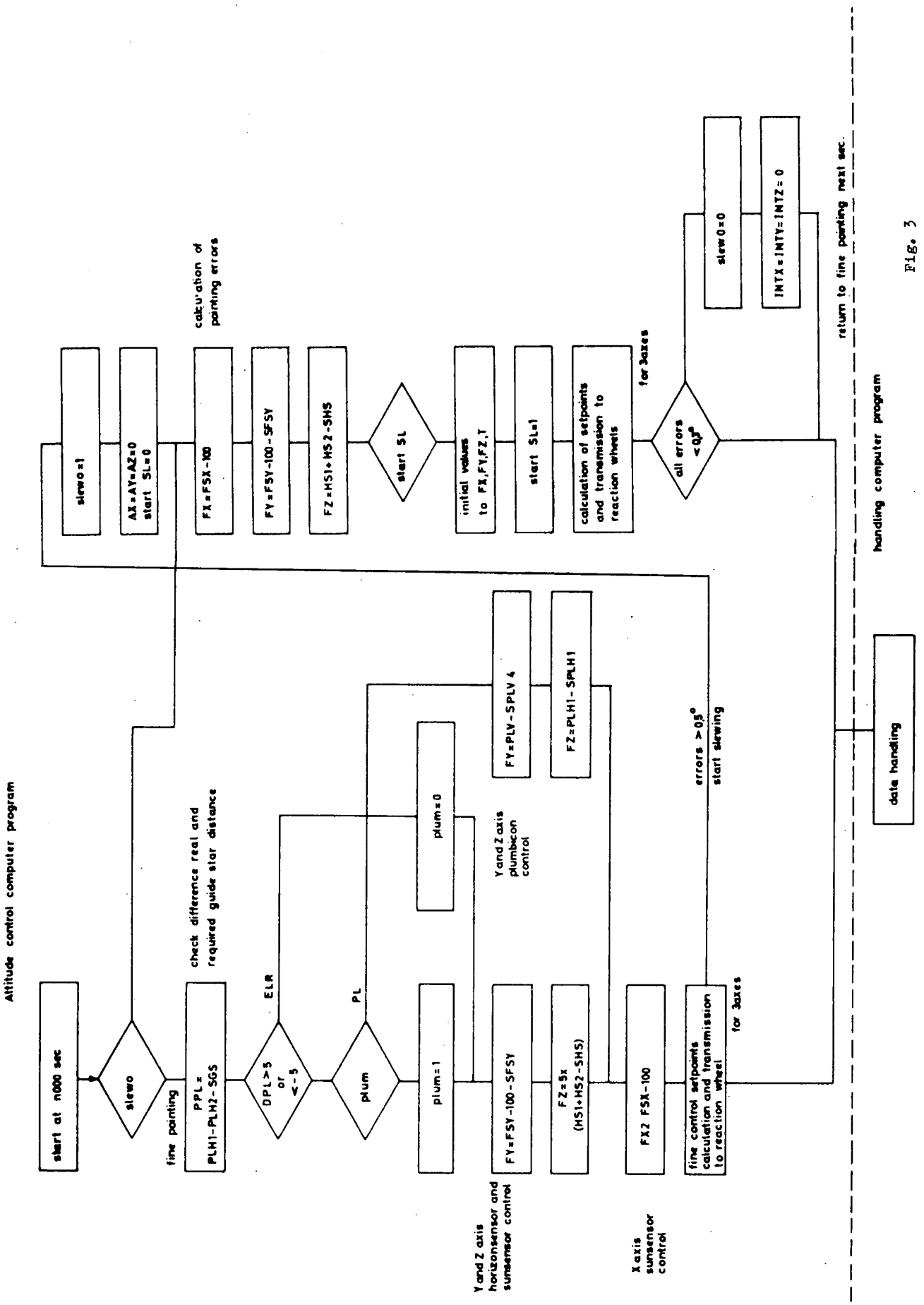

Attitude control computer program
start at n000 sec
slewo
fine pointing
PPL=
PLH1-PLH2-SGS
check difference real and required guide star distance
DPL>5 or <-5
ELR
PL
plum
plum=1
plum=0
Y and Z axis plumbicon control
FY=PLV-SPLV 4
FZ=PLH1-SPLH1
Y and Z axis horizonsensor and sunsensor control
FY=FSY-100-SFSY
FZ=Sx (HS1+HS2-SHS)
X axis sunsensor control
FX2 FSX-100
fine control setpoints calculation and transmission to reaction wheel
for 3axes
errors >0,5°
start slewing
slewo=1
AX=AY=AZ=0
start SL=0
calculation of pointing errors
FX=FSX-100
FY=FSY-100-SFSY
FZ=HS1+HS2-SHS
start SL
initial values to FX,FY,FZ,T
start SL=1
calculation of setpoints and transmission to reaction wheels
for 3axes
all errors <0,5°
slewo=0
INTX=INTY=INTZ=0
return to fine pointing next sec.
handling computer program
data handling

Fig. 3

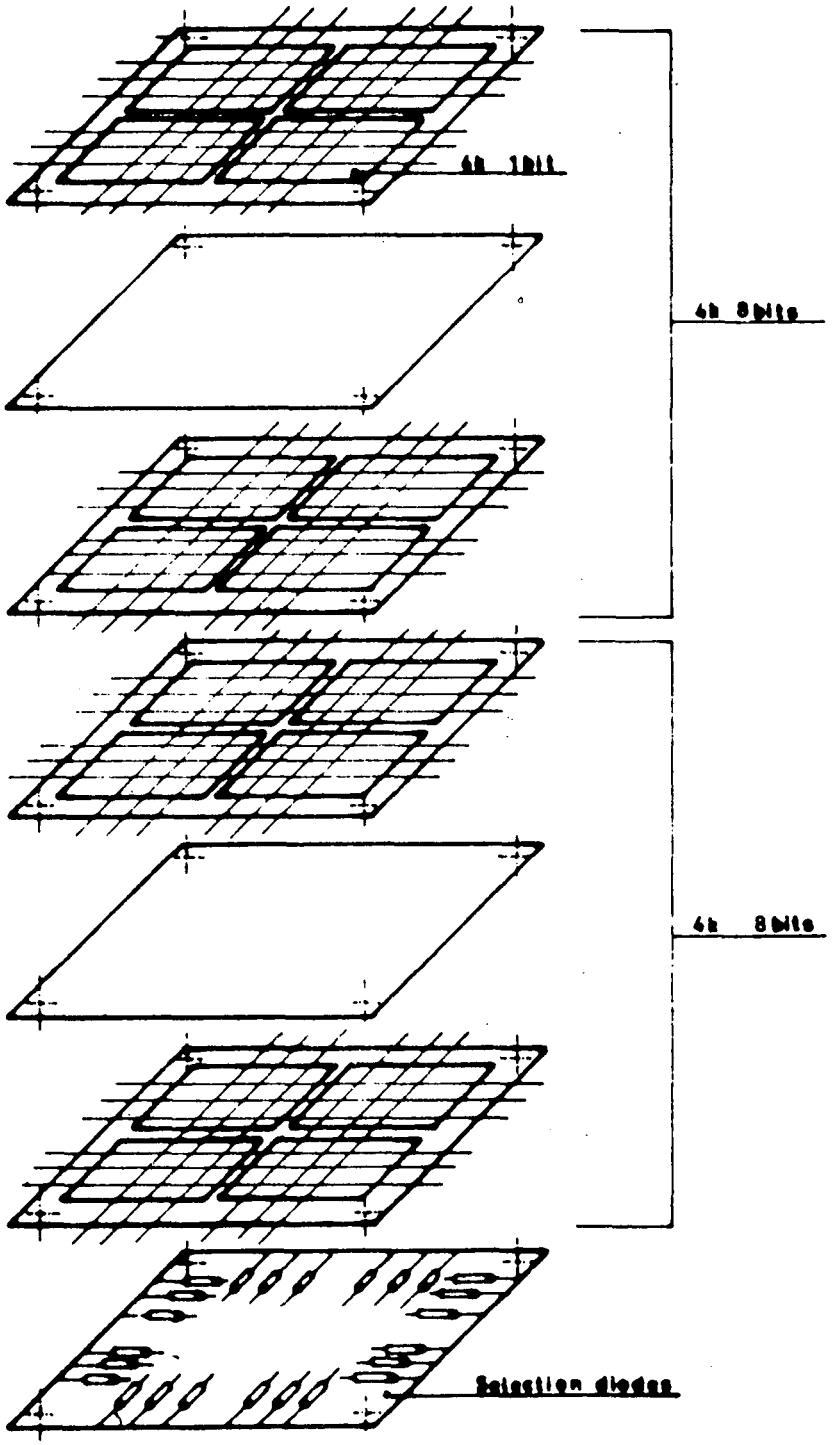


Fig. 4 Memory block configuration

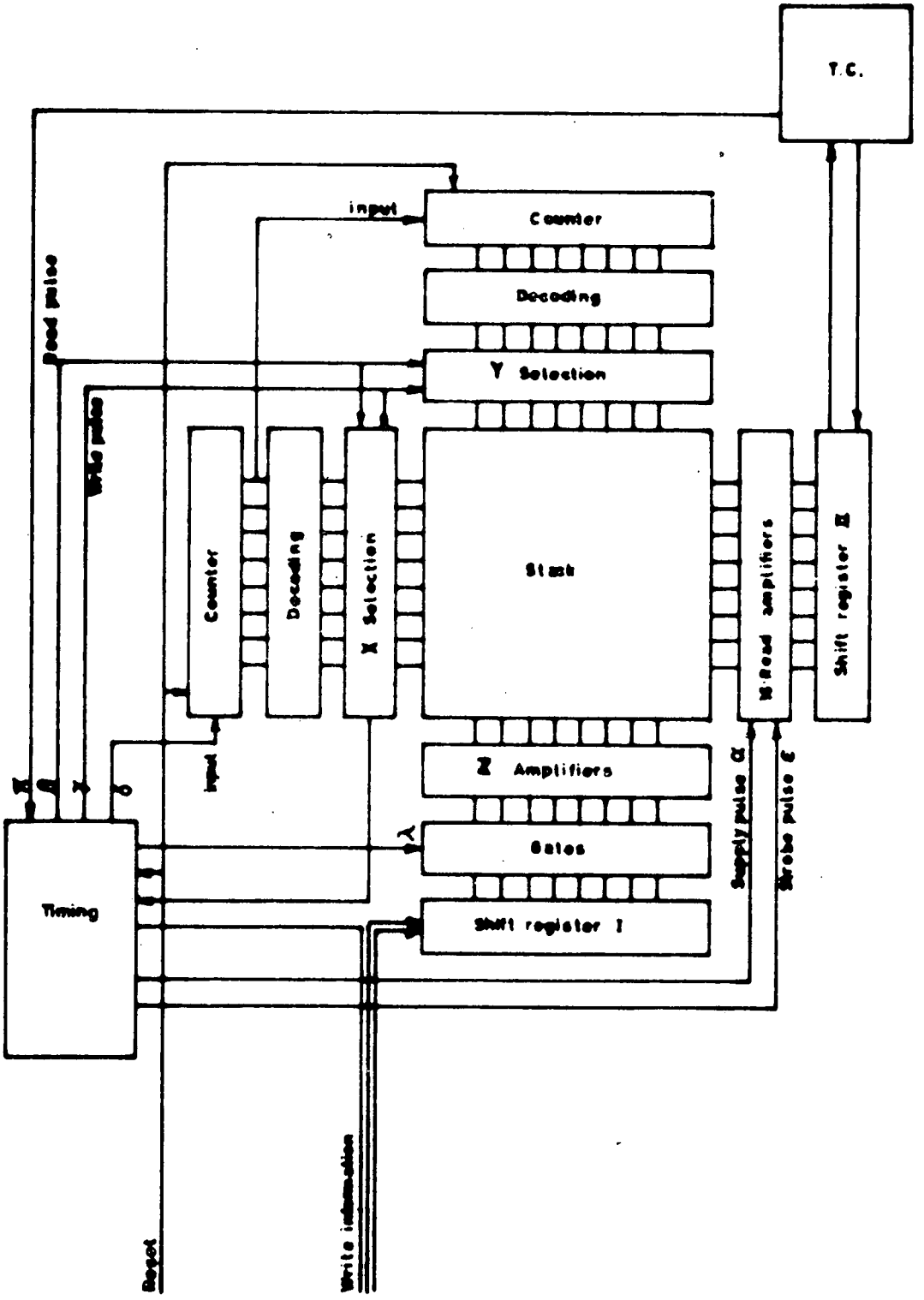


Fig. 5 Block diagram of corememory

AGARD Conference Proceedings No.68 North Atlantic Treaty Organization, Advisory Group For Aerospace Research and Development THE APPLICATION OF DIGITAL COMPUTERS TO GUIDANCE AND CONTROL Published November 1970 296 pages The papers presented at the Guidance and Control Panel sponsored Symposium on the Application of Digital Computers to Guidance and Control held in London, England in June 1970. The papers emphasize both the manner in which digital computers are being or can be used to improve the navigation and control of aerospace vehicles and on the design of the computers themselves. Papers presented at a Symposium of the Guidance and Control Panel of AGARD, held in London, England, 2 - 4 June 1970.	681.32:629.7.05	AGARD Conference Proceedings No.68 North Atlantic Treaty Organization, Advisory Group for Aerospace Research and Development THE APPLICATION OF DIGITAL COMPUTERS TO GUIDANCE AND CONTROL Published November 1970 296 pages The papers presented at the Guidance and Control Panel sponsored Symposium on the Application of Digital Computers to Guidance and Control held in London, England in June 1970. The papers emphasize both the manner in which digital computers are being or can be used to improve the navigation and control of aerospace vehicles and on the design of the computers themselves. Papers presented at a Symposium of the Guidance and Control Panel of AGARD, held in London, England, 2 - 4 June 1970.	681.32:629.7.05
AGARD Conference Proceedings No.68 North Atlantic Treaty Organization, Advisory Group for Aerospace Research and Development THE APPLICATION OF DIGITAL COMPUTERS TO GUIDANCE AND CONTROL Published November 1970 296 pages The papers presented at the Guidance and Control Panel sponsored Symposium on the Application of Digital Computers to Guidance and Control held in London, England in June 1970. The papers emphasize both the manner in which digital computers are being or can be used to improve the navigation and control of aerospace vehicles and on the design of the computers themselves. Papers presented at a Symposium of the Guidance and Control Panel of AGARD, held in London, England, 2 - 4 June 1970.	681.32:629.7.05	AGARD Conference Proceedings No.68 North Atlantic Treaty Organization, Advisory Group for Aerospace Research and Development THE APPLICATION OF DIGITAL COMPUTERS TO GUIDANCE AND CONTROL Published November 1970 296 pages The papers presented at the Guidance and Control Panel sponsored Symposium on the Application of Digital Computers to Guidance and Control held in London, England in June 1970. The papers emphasize both the manner in which digital computers are being or can be used to improve the navigation and control of aerospace vehicles and on the design of the computers themselves. Papers presented at a Symposium of the Guidance and Control Panel of AGARD, held in London, England, 2 - 4 June 1970.	681.32:629.7.05

AGARD Conference Proceedings No.68
North Atlantic Treaty Organization, Advisory Group For Aerospace Research and Development
THE APPLICATION OF DIGITAL COMPUTERS TO GUIDANCE AND CONTROL
Published November 1970
296 pages

The papers presented at the Guidance and Control Panel sponsored Symposium on the Application of Digital Computers to Guidance and Control held in London, England in June 1970. The papers emphasize both the manner in which digital computers are being or can be used to improve the navigation and control of aerospace vehicles and on the design of the computers themselves.

Papers presented at a Symposium of the Guidance and Control Panel of AGARD, held in London, England, 2 - 4 June 1970.

681.32:629.7.05

AGARD Conference Proceedings No.68
North Atlantic Treaty Organization, Advisory Group for Aerospace Research and Development
THE APPLICATION OF DIGITAL COMPUTERS TO GUIDANCE AND CONTROL
Published November 1970
296 pages

The papers presented at the Guidance and Control Panel sponsored Symposium on the Application of Digital Computers to Guidance and Control held in London, England in June 1970. The papers emphasize both the manner in which digital computers are being or can be used to improve the navigation and control of aerospace vehicles and on the design of the computers themselves.

Papers presented at a Symposium of the Guidance and Control Panel of AGARD, held in London, England, 2 - 4 June 1970.

681.32:629.7.05

AGARD Conference Proceedings No.68
North Atlantic Treaty Organization, Advisory Group for Aerospace Research and Development
THE APPLICATION OF DIGITAL COMPUTERS TO GUIDANCE AND CONTROL
Published November 1970
296 pages

The papers presented at the Guidance and Control Panel sponsored Symposium on the Application of Digital Computers to Guidance and Control held in London, England in June 1970. The papers emphasize both the manner in which digital computers are being or can be used to improve the navigation and control of aerospace vehicles and on the design of the computers themselves.

Papers presented at a Symposium of the Guidance and Control Panel of AGARD, held in London, England, 2 - 4 June 1970.

681.32:629.7.05

AGARD Conference Proceedings No.68
North Atlantic Treaty Organization, Advisory Group for Aerospace Research and Development
THE APPLICATION OF DIGITAL COMPUTERS TO GUIDANCE AND CONTROL
Published November 1970
296 pages

The papers presented at the Guidance and Control Panel sponsored Symposium on the Application of Digital Computers to Guidance and Control held in London, England in June 1970. The papers emphasize both the manner in which digital computers are being or can be used to improve the navigation and control of aerospace vehicles and on the design of the computers themselves.

Papers presented at a Symposium of the Guidance and Control Panel of AGARD, held in London, England, 2 - 4 June 1970.

681.32:629.7.05

331205.

NATIONAL DISTRIBUTION CENTRES FOR UNCLASSIFIED AGARD PUBLICATIONS

Unclassified AGARD publications are distributed to NATO Member Nations through the unclassified National Distribution Centres listed below

BELGIUM
General J. DELHAYE
Coordinateur AGARD – V.S.L.
Etat Major Forces Aériennes
Caserne Prince Baudouin
Place Dailly, Bruxelles 3

CANADA
Director of Scientific Information Services
Defence Research Board
Department of National Defence – 'A' Building
Ottawa, Ontario

DENMARK
Danish Defence Research Board
Østerbrogades Kaserne
Copenhagen Ø

FRANCE
O.N.E.R.A. (Direction)
29, Avenue de la Division Leclerc
92, Châtillon-sous-Bagneux

GERMANY
Zentralstelle fur Lüftfahrtdokumentation
und Information
Maria-Theresia Str. 21
8 München 27
Attn: Dr Ing. H.J. RAUTENBERG

GREECE
Hellenic Armed Forces Command
D Branch, Athens

ICELAND
Director of Aviation
c/o Flugrad
Reykjavik

ITALY
Aeronautica Militare
Ufficio del Delegato Nazionale all' AGARD
3, P. le del Turismo
Roma/Eur

LUXEMBOURG
Obtainable through BELGIUM

NETHERLANDS
Netherlands Delegation to AGARD
National Aerospace Laboratory, NLR
Attn: Mr A.H. GEUDEKER
P.O. Box 126
Delft

NORWAY
Norwegian Defense Research Establishment
Main Library, c/o Mr P.L. EKERN
P.O. Box 25
N-2007 Kjeller

PORTUGAL
Direccao do Servico de Material
da Forca Aerea
Rua de Escola Politecnica 42
Lisboa
Attn: Brig. General Jose de Sousa OLIVEIRA

TURKEY
Turkish General Staff (ARGE)
Ankara

UNITED KINGDOM
Ministry of Technology Reports Centre
Station Square House
St. Mary Cray
Orpington, Kent BR5 3RE

UNITED STATES
National Aeronautics and Space Administration (NASA)
Langley Field, Virginia 23365
Attn: Report Distribution and Storage Unit

★ ★ ★

If copies of the original publication are not available at these centres, the following may be purchased from:

Microfiche or Photocopy

Clearinghouse for Federal
Scientific and Technical
Information (CFSTI)
Springfield
Virginia 22151, USA

Microfiche

ESRO/ELDO Space
Documentation Service
European Space
Research Organization
114, Avenue de Neuilly
92, Neuilly-sur-Seine, France

Microfiche

Ministry of Technology
Reports Centre
Station Square House
St. Mary Cray
Orpington, Kent BR5 3RE
England

The request for microfiche or photocopy of an AGARD document should include the AGARD serial number, title, author or editor, and publication date. Requests to CFSTI should include the NASA accession report number.

Full bibliographical references and abstracts of the newly issued AGARD publications are given in the following bi-monthly abstract journals with indexes:

Scientific and Technical Aerospace Reports (STAR)
published by NASA,
Scientific and Technical Information Facility,
P.O. Box 33, College Park,
Maryland 20740, USA

United States Government Research and Development Report Index (USGDRI), published by the Clearinghouse for Federal Scientific and Technical Information, Springfield, Virginia 22151, USA

Printed by Technical Editing and Reproduction Ltd
Harford House, 7-9 Charlotte St, London. W1P 1HD