RTO-EN-022
AC/323(AVT-095)TP/69

RTO-EN-022

NORTH ATLANTIC TREATY ORGANISATION

RESEARCH AND TECHNOLOGY ORGANISATION

BP 25, 7 RUE ANCELLE, F-92201 NEUILLY-SUR-SEINE CEDEX, FRANCE

RTO EDUCATIONAL NOTES 22

Intelligent Systems for Aeronautics

(Systèmes intelligents pour l'aéronautique)

The material in this publication was assembled to support a RTO/VKI Special Course under the sponsorship of the Applied Vehicle Technology Panel (AVT) and the von Kármán Institute for Fluid Dynamics (VKI) presented on 13-17 May 2002, in Rhode-Saint-Genèse, Belgium.

Published June 2003

Distribution and Availability on Back Cover

This page has been deliberately left blank

Page intentionnellement blanche

RTO-EN-022
AC/323(AVT-095)TP/69

NORTH ATLANTIC TREATY ORGANISATION

RESEARCH AND TECHNOLOGY ORGANISATION

BP 25, 7 RUE ANCELLE, F-92201 NEUILLY-SUR-SEINE CEDEX, FRANCE

RTO EDUCATIONAL NOTES 22

Intelligent Systems for Aeronautics

(Systèmes intelligents pour l'aéronautique)

The material in this publication was assembled to support a RTO/VKI Special Course under the sponsorship of the Applied Vehicle Technology Panel (AVT) and the von Kármán Institute for Fluid Dynamics (VKI) presented on 13-17 May 2002, in Rhode-Saint-Genèse, Belgium.

The Research and Technology Organisation (RTO) of NATO

Published June 2003

Copyright © RTO/NATO 2003
All Rights Reserved

ISBN 92-837-1101-7

Printed by St. Joseph Print Group Inc.
(A St. Joseph Corporation Company)
1165 Kenaston Street, Ottawa, Ontario, Canada K1G 6S1

Intelligent Systems for Aeronautics

(RTO EN-022 / AVT-095)

Executive Summary

Intelligent Systems (IS) are nature-inspired problem solving tools and methodologies that have recently become important in information technology applications. Artificially intelligent systems use computers to emulate various faculties of human intelligence, and biological metaphors. They use a combination of symbolic and sub-symbolic systems capable of developing human-like cognitive skills and intelligence, not just systems capable of doing things humans do not do well. Intelligent systems are ideally suited for tasks such as search and optimization, pattern recognition and matching, planning, uncertainty management, control and adaptation.

These lecture notes approach IS from two perspectives: techniques and applications. The emphasis is on aeronautical and space applications of IS, rather than basic research or tool development. The objectives of this NATO Research and Technology Organisation (RTO) sponsored Lecture series is to provide a series of comprehensive lectures by leading experts:

1. to enable the understanding of the concept, history, and benefits of intelligent system technologies;
2. to enable an understanding of the areas of applicability of IS technologies to aeronautics and space;
3. to enable an understanding of the state-of-the-art applications of IS technologies in aeronautics and space.

The lectures were intended to accommodate attendees of both novice and advanced levels of technical expertise.

For this purpose, the following techniques commonly used in IS were reviewed: decision strategy tools based on game theory (Dr. J. Périaux, MDBA, France), neural network techniques for fault identification (Prof. M. Innocenti, U. Pisa, Italy), genetic algorithms (Dr. M. Anderson, Sverdrup Technology, USA; Dr. D. Quagliarella, CIRA, Italy) and multi-agent theory (Dr. I. Degirmenciyan, MDBA, France)

These techniques were illustrated by numerous applications: optimal air combat tactics (Dr. K. Krishnakumar, NASA Ames), control of unmanned air vehicles (Dr. M. Ricard, Draper Lab, USA), air combat simulation (Dr. I. Degirmenciyan, MDBA, France), space exploration (Dr. R. Doyle, JPL, USA), unmanned aircraft navigation and formation control (Prof. M. Innocenti, U. Pisa, Italy), missile design (Dr. M. Anderson, Sverdrup Technology, USA), airfoil design (Dr. D. Quagliarella, CIRA, Italy) and finally analysis of rocket plume data for condition monitoring (Prof. K. Whitaker, U. Alabama, USA). Most of these applications have a clear military aspect, whether at the system design level (missile design, control of unmanned air vehicles) or at operational level (optimal air combat tactics, unmanned aircraft formation control, air combat simulation), hence highlighting the relevance and importance of Intelligent Systems for military issues.

G. Degrez[1] & K. Krishnakumar.[2]
[1]von Kármán Institute, Belgium & [2]NASA Ames, USA
Lecture Series Editors

Systèmes intelligents pour l'aéronautique

(RTO EN-022 / AVT-095)

Synthèse

Les systèmes intelligents (SI) sont des outils et des méthodologies de résolution de problèmes, d'inspiration humaine, qui ont récemment pris de l'importance dans les applications des technologies de l'information. Les systèmes d'intelligence artificielle font appel à des ordinateurs et à des métaphores biologiques pour imiter les différentes facultés de l'intelligence humaine. Ils exploitent en combinaison des systèmes symboliques et sous-symboliques capables de développer des compétences cognitives et de l'intelligence pseudo-humaines et non pas simplement des systèmes capables de réaliser ce que l'être humain ne réalise pas bien. Les systèmes intelligents sont idéalement adaptés à des tâches telles que la recherche et l'optimisation, la reconnaissance des formes, la planification, la gestion de l'incertitude, le contrôle et l'adaptation.

Ce support de cours aborde les systèmes intelligents sous deux angles différents : techniques et applications. L'accent est mis sur les applications aéronautiques et spatiales des SI plutôt que la recherche de base ou le développement d'outils. Ce cycle de conférences organisé par l'Organisation OTAN pour la recherche et la technologie (RTO) a pour objectif de proposer une série de conférences complètes présentées par d'éminents spécialistes du domaine afin de permettre de mieux comprendre :

1. le concept, l'historique, et les avantages des technologies des systèmes intelligents;
2. les domaines d'application des technologies des SI pour l'aéronautique et l'espace;
3. les applications de pointe des technologies des SI pour l'aéronautique et l'espace.

Les conférences sont destinées à des participants de tous niveaux de compétence technique, du novice jusqu'au spécialiste.

Dans cette optique, les techniques suivantes, couramment employées en IS, ont été examinées : des outils de prise de décisions stratégiques basés sur la théorie des jeux (Dr. J. Périaux, MDBA, France), des techniques de réseaux neuronaux pour l'identification des défaillances (Prof. M. Innocenti, U. Pisa, Italie), des algorithmes génétiques (Dr. M. Anderson, Sverdrup Technology, USA; Dr. D Quagliarella, CIRA, Italie), et des théories multi-agents (Dr. I. Degirmenciyan, MDBA, France)

Ces techniques ont été illustrées par de nombreuses applications : les tactiques optimales de combat aérien (Dr. K Krishnakumar, NASA, Ames), le pilotage des véhicules aériens sans pilote (Dr. M. Ricard, Draper Lab, USA), la simulation du combat aérien (Dr. I. Degirmenciyan, MDBA, France), l'exploration de l'espace (Dr. R. Doyle, JPL, USA), la navigation et le contrôle de formation des véhicules sans pilote (Prof. M. Innocenti, U. Pisa, Italie), la conception des missiles (Dr. M. Anderson, Svedrup Technology, USA), la conception des profils aérodynamiques (Dr. D. Quagliarella, CIRA, Italie), et finalement l'analyse des données de panache des moteurs fusées pour le contrôle de l'état du moteur (Prof. K. Whitaker, U. Alabama, USA). La plupart de ces applications ont des implications militaires évidentes, soit au niveau de la conception du système (conception des missiles, pilotage des véhicules aériens sans pilote) soit au niveau opérationnel (les tactiques optimales de combat, le contrôle d'aéronefs sans pilote en formation, la simulation du combat aérien), ce qui fait ressortir l'intérêt et l'importance de systèmes intelligents pour applications militaires.

G. Degrez[1] & K. Krishnakumar.[2]
[1]von Kármán Institute, Belgium & [2]NASA Ames, USA
Lecture Series Editors

Contents

† Paper not provided in time for publication.

Preface

Intelligent Systems (IS) embody varying degrees of representation of biological and cognitive systems for efficient solution of complex problems. In our society today, intelligent systems have become extremely fascinating tools for significantly improving the trends in information technology applications. Intelligent systems are ideally suited for tasks such as search and optimization, pattern recognition and matching, planning, uncertainty management, control and adaptation. This lecture notes will approach IS from two ends: techniques and applications. The emphasis is on aeronautical and space applications of IS, rather than basic research or tool development.

The objectives of this NATO Research and Technology Office (RTO) sponsored Lecture series is to provide a series of comprehensive lectures by leading experts.

1. To enable the understanding of the concept, history, and benefits of intelligent system technologies;
2. To enable an understanding of the areas of applicability of IS technologies to aeronautics and space;
3. To enable an understanding of the state-of-the-art applications of IS technologies in aeronautics and space. The lectures are intended to accommodate attendees of both novice and advanced levels of technical expertise.

G. Degrez[1] & K. Krishnakumar.[2]
[1]Von Kármán Institute, Belgium & [2]NASA Ames, USA
Lecture Series Editors

List of Authors

Lecture Series Director

Mr. K. Krishnakumar
NASA Ames Research Centre
MS 269-1, Moffett Field
CA 94035-1000
email: kkumar@mail.arc.nasa.gov

VKI Local Coordinator

Prof. G. Degrez
von Kármán Institute
72, Chaussee de Waterloo
1640 Rhode Saint Genese
email: degrez@vki.ac.be

Authors

FRANCE
Dr. I. Degirmenciyan
Dassault Aviation
DGT/DPR/ESA
78, quai Marcel Dassault
92552 St. Cloud Cedex 300
email: irene.degirmenciyan@dassault-aviation.fr

Dr. J. Periaux
Direction de la Prospective
DPR
78, quai Marcel Dassault
92214 St Cloud Cedex
email: jacques.periaux@dassault-aviation.fr

ITALY
Prof. M. Innocenti
University of Pisa
Department of Electric Systems and Automation
via Diotisalvi 2, 56126 Pisa

Dr. D. Quagliarella
CIRA
via Maiorise
81043 Capua (CE)
d.quagliarella@cira.it

UNITED STATES
Dr. M. Anderson
Sverdrup Technology Inc.
Bldg 260, TEAS Group
308 West D. Drive,
Eglin AFB, FI 32542-0935
email: anderson@eglin.af.mil

Dr. R. Doyle
Jet Propulsion Lab
Information Tech and Software Systems
MS 126-221/4800 Oak Grove Drive
Pasadena, CA 91109-8099
email: rdoyle@jpl.nasa.gov

Mr. S. Kolitz
The Charles Stark Draper Laboratory
MS 3F, 555 Technology Square
Cambridge, MA 02139 3563
email: kolitz@draper.com

Dr. M. Ricard
The Charles Stark Draper Laboratory
555 Technology Square
Cambridge, MA 02139
email: mricard@draper.com

Prof. K. Whitaker
University of Alabama
P.O. Box 870200
Tuscaloosa, AL 35487-0154

Intelligent Systems For Aerospace Engineering – An Overview

K. KrishnaKumar
NeuroEngineering Laboratory
NASA Ames Research Center
MS 269-1, Moffett Field
Cambridge, MA 02139-3563 USA

kkumar@mail.arc.nasa.gov

1 Abstract

Intelligent systems are nature-inspired, mathematically sound, computationally intensive problem solving tools and methodologies that have become extremely important for advancing the current trends in information technology. Artificially intelligent systems currently utilize computers to emulate various faculties of human intelligence and biological metaphors. They use a combination of symbolic and sub-symbolic systems capable of evolving human cognitive skills and intelligence, not just systems capable of doing things humans do not do well. Intelligent systems are ideally suited for tasks such as search and optimization, pattern recognition and matching, planning, uncertainty management, control, and adaptation. In this paper, the intelligent system technologies and their application potential are highlighted via several examples.

2 Defining Intelligent Systems

Science has evolved with our efforts towards understanding and mimicking nature, through inventions and discoveries, hypotheses and proofs, success and failures. The evolution of computers marks the era of our success in building systems that can perform actions of a repetitive kind, those which are difficult or time consuming if done by humans. This has helped enhance our efforts towards studying and understanding the intelligence of biological systems and applying this knowledge towards building artificially intelligent systems.

We attribute intelligence to various faculties. Intelligence is often identified in terms of competence, expertise, talent, schooling, IQ, and social interaction. From the perspective of computation, the intelligence of a system is characterized by its flexibility, adaptability, memory, learning, temporal dynamics, reasoning, and the ability to manage uncertain and imprecise information. Capabilities such as information gathering, understanding, making inferences, and applying it to understand and solve new problems efficiently are observed to be critical features of such systems. Intelligence has been defined in several ways:

- the ability to learn or understand from experience
- ability to acquire and retain knowledge
- mental ability
- the ability to respond quickly and successfully to a new situation
- use of the faculty of reason in solving problems, directing conduct, etc. effectively

Although Artificial Intelligence (AI) is the corner stone on which Intelligent System (IS) technologies are built, there are distinguishing differences between AI and IS. These differences are highlighted next.

AI has evolved with two distinct dimensions (Russell and Norvig, 1995), one is "humanistic AI" and the other is "rationalistic AI". Humanistic AI examines machines that think and act like-humans, whereas rationalistic AI examines machines that can be built on the understanding of intelligent human behavior. Some definitions of AI related to these ideas are presented below.

Humanistic AI
"The art of creating machines that perform functions that require intelligence when performed by people" (Kurzweil, 1990)
"The study of how to make computers do things at which, at the moment, people are better" (Rich and Knight, 1991)

Rationalistic AI
"A field of study that seeks to explain and emulate intelligent behavior in terms of computational processes" (Schalkoff, 1990)
"The branch of computer science that is concerned with the automation of intelligent behavior" (Luger and Stubblefield, 1993)

Intelligent systems as envisioned today are mostly modeled after rationalistic AI. They examine intelligent behavior using the models of human systems that enable intelligent behavior. In addition, IS has distinguished itself from AI by including intelligent behavior as seen in nature as a whole. This intelligent behavior includes biological genetics (distributed information), evolution (survivability), chaos (structured randomness), and natural adaptation (sufficability). Also, innovations in IS are driven by the need to solve complex problems with improving efficiencies. This improvement could be over a period of computing time, real-time, or over a set of problems encountered. The idea is not to converge to an optimal solution but to find solutions that are better than their predecessors. The underlying assumption here is that the problem domain is either too enormous with few good solutions or is non stationary.

So how can one define intelligent systems? This is indeed a difficult question and is subject to a great deal of debate. From the author's view point, an intelligent system is one that emulates some aspects of intelligence exhibited by nature. These include:

1. Learning
2. Adaptability
3. Robustness across problem domains
4. Improving efficiency (over time and/or space)
5. Information compression (data to knowledge)
6. Extrapolated reasoning

The debate of what is an intelligent system will be around as long as the definition of intelligence itself eludes us. From the practical view point, IS technologies have made it easier to achieve some level of complexity in aerospace applications just as much computers have enabled certain level of complexity

3 Role of Intelligent Systems in Aerospace Engineering

Intelligent System (IS) applications have gained popularity among aerospace professionals in the last decade due to the ease with which several of the IS tools can be implemented. In addition to this ease of implementation, IS has been shown to solve difficult problems more efficiently. Another advantage that IS practitioners have seen is complex ideas can be implemented and tested with rapid development cycles. All of these rely heavily on the ubiquitous nature of computing machines with the power of 20th century supercomputers. The applications have gained popularity among the technical and user communities for both intellectual curiosity and for practical reasons. Some of the novel ideas using IS include spacecraft autonomy, aircraft control, modeling, airfoil design, satellite operations, missile design, and vehicle health management. In essence one can say that the IS technologies help achieve efficiency, robustness in addition to providing human-like capabilities such as pattern recognition, learning, long-term optimization, planning, and self-improvement.

The role of intelligent systems in aerospace engineering is two-fold: (1) function as intelligent assistants to augment human expertise; and (2) act as a substitute for human expertise in endeavors that save cost, time, and life. For example, intelligent systems assist humans in solving difficult optimization problem by their shear ability to robustly search through myriad of choices. In contrast, intelligent systems are used on autonomous rovers to both save cost and human lives.

In the next several sections, we present some areas of applications using intelligent systems and highlight the benefits using examples related to aerospace engineering.

3.1 Intelligent Systems for Modeling

Modeling could be thought of as a representation of available information. Intelligent systems provide two very important features for modeling: generalization and robustness. Generalization implies that the model could be used not only to represent just the data gathered but the knowledge the data represents. Robustness can be defined as the system's ability to perform within certain bounds of its nominal (without uncertainty) performance in the presence of bounded uncertainty. Several techniques such as neural networks, fuzzy logic, expert systems, etc have been routinely used by aerospace engineers for modeling. Knowledge representation in general contains syntax and semantics. Syntax is the constitution of sentences and Semantics is the interpretation of sentences. Examples of knowledge representation include

- **Mathematical Equations (Ex: ARMA Models**: Autoregressive moving average models are created using the least-square technique to represent linear relationship between inputs and outputs.)
- **Rule-based systems.** Reasoning is a process of arriving at a conclusion based on a collection of premises. Expert system based reasoning is one of the popular rule-based inferencing system that is in use today. This consists of three parts: a knowledge base (a set of rules and known facts); acquired data (derived facts and data); and an inference engine (reasoning logic).
- **Fuzzy Models:** Given the choice of system input and output variables, their linguistic modifiers with the associated fuzzy membership functions, an appropriate implication function, aggregation function, and defuzzification operator, if so desired, a fuzzy model that represents the system can be specified by a set of rules, their structure, and the fuzzy membership function parameters. Fuzzy systems model qualitative and quantitative non-linearity of systems. Attractive features include reduced design complexity, rapid prototyping, flexibility, simplicity, cost effectiveness, and inherent parallelism. This

explains the popularity of fuzzy systems in diverse application areas such as control, prediction, evaluation, cognition, analysis, and information management.

- **Neural Models:** Artificial Neural Networks (ANNs) are brain-inspired connectionist models that consist of many similar linear and nonlinear computational elements connected in complex patterns. The simple computational elements, also known as neurons, when associated in complex patterns, have the ability to perform tasks such as memory recall, pattern recognition, and learning. The ability of neural networks to learn from repeated exposure to system characteristics has made them a popular choice for many applications in image processing, system identification and control, pattern recognition and classification, financial prediction, and signal processing. Neural models can be sigmoidal networks, associative neural networks, radial basis function neural networks, or clustering networks. The choices are many.
- **Tree structures:** A tree structure is an algorithm for placing and locating objects in a database. The algorithm finds data by repeatedly making choices at decision points called nodes. A node can have branches (also called children) whose number can vary from two to several dozens. The structure is straightforward, but in terms of the number of nodes and children, a tree can be gigantic.

In aerospace applications, modeling is ubiquitous. Intelligent modeling tools are used for operator behavior modeling, flight test data modeling, aerodynamic modeling for design, etc.

3.1.1 Improved Robustness using Fuzzification

One of the benefits of fuzzy logic modeling is the fuzzy granularization that is obtained by defining fuzzy sets for the inputs and outputs of a system. Krishnakumar and Kulkarni (1998) outlined an approach that combines this granularization advantage with the knowledge available through an existing linear dynamic controller, to arrive at a powerful hybrid technique. This technique provides an outer fuzzy shell to existing control techniques and therefore combines the strengths of conventional as well as fuzzy control methodologies. The equivalent fuzzy dynamic controller is shown in Figure 1. It was also shown via a non-linear robustness analysis, that fuzzification of the inputs and outputs lead to better robustness to system uncertainties via the manipulation of the distance between the membership functions.

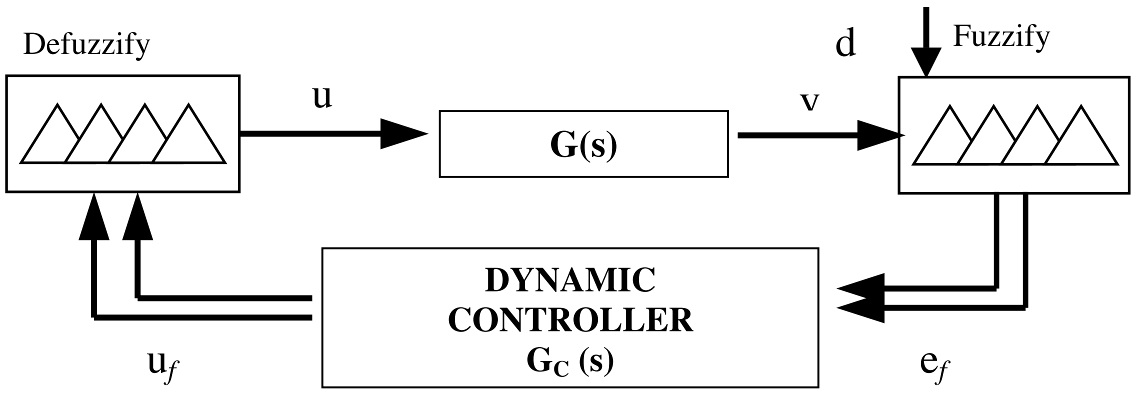


Figure 1. Equivalent Fuzzy Dynamic Controller Architecture

3.2 Intelligent Systems for Search

If the solution to a task can be represented by a set of N parameters, then the job of finding this solution can be thought of as a search in an N-dimensional space. This is referred to simply as the *search space*. More generally, if the solution to a task can be represented using a representation scheme, R, then the search space is the set of all possible configurations that may be represented in R.

Search techniques are also employed in problems where we want to determine if we can reach the desired goal state from an initial state, to minimize the cost in reaching the goal state, etc. The state-space approach is one of the primary representations for such problems. The state-space consists of an *initial state* and a set of *operators*. Application of the operators produces a sequence of new states called the *path.* A *goal test* is defined and tested to determine when a new state is the desired goal state.

Search techniques are routinely employed in combinatorial optimization. Some tasks involve combining a set of entities in a specific way (e.g. the task of building a combat tactics plan). A general combinatorial task involves deciding (a) the specifications of those entities (e.g. what type of aircraft, the opponents aircraft descriptions, number of aircraft, etc), and (b) the way in which those entities are brought together (e.g. various elementary tactics formations and their relative positions). If the resulting combination of entities can in some way be given a fitness score, then combinatorial optimization is the task of designing a set of entities, and deciding how they must be configured, so as to give maximum fitness. Some of the popular search techniques used by the intelligent systems community includes:

- **Golden Section**: The most used one-dimensional search technique that guarantees an optimum in a finite amount of time is the Golden section search technique. This technique belongs to the interval reduction family in which the interval is reduced by throwing out regions that definitely do not contain the optimum.

- **Breadth-first Search:** Search a state space by constructing a tree consisting of a set of leaves and branches. The algorithm defines a way to move through the tree structure.

- **Depth-first Search:** Instead of completely searching each level of the tree before going deeper, the algorithm follows a single branch of the tree down as many levels as possible until a solution or a dead-end is reached.

- **Heuristic Dynamic Programming (HDP):** HDP is based on an attempt to approximate Howard's form of the Bellman equation (Howard, 1960):

$$J(x_t) = \underset{u_t}{Min}\left\{ U_{PM}(x_t) + \gamma J(x_{t+1}) \middle| \quad x_{t+1} = f(x_t, u_t, noise) \right\}$$

  where x_t is the state vector, u_t is the control vector, $U_{PM}(.)$ is the one stage Performance Measure function, $f(.,.,.)$ is the model of the system, and $\gamma(0 < \gamma \leq 1)$ is the discount factor. Variations of HDP are used in solving combinatorial optimization problems.

- **A*Search:** This is the most famous search algorithm used by the AI community. Here we combine the cost estimate *J(n)* of traversing from step *n* to the goal state and *U(n)* which is the known path cost from the start node to step *n*.

- **Genetic Algorithms:** A type of evolutionary computation devised by John Holland (Holland, 1975). A model of machine learning that uses a genetic/evolutionary metaphor. Implementations typically use fixed-length character strings to represent their genetic information, together with a population of individuals which undergo crossover and mutation in order to find interesting regions of the search space.

In aerospace applications, intelligent search techniques are used typically for searching a design space, searching for an optimal scheduling and planning schemes, and for solving combinatorial optimization problems. In the example presented next, we examine the use of genetic algorithms for finding optimal tactics formation.

3.2.1 Application: Search for Optimal Air Combat Tactics

The complete design and specification of air combat tactics for many-vs.-many engagements poses a considerable challenge to tactical planners. While solutions have been developed for one-vs.-one or few-vs.-few encounters, the results may not generalize to larger engagements where formation tactics become increasingly important. The most effective formation tactics employ a basic fighting unit of two aircraft (called a *section* or *element*) (Shaw, 1988). Since this is how all fighter pilots learn their craft, it would be most effective for optimized tactics not to deviate from this established tactical doctrine. Accordingly, software tactics modules that employ basic fighting units of two aircraft were developed. Since large numbers of fighter aircraft are difficult to control, formation tactics for large groups may be developed using a hierarchical structure consisting of smaller units or divisions; for example, a four-airplane division called the *fluid four* consists of two elements. Each element consists of two aircraft, but they are treated as a unit. This hierarchical concept is used to develop a GA-based approach to search for optimal air combat tactics. Given a palette of air combat maneuvers and standard small-formation tactics as building blocks, GAs are used to determine how they can be integrated to produce large fighting groups that optimize overall combat effectiveness.

Tactics implementation proceeds as follows (The complete simulation environment is presented in Figure 2. For more details, see (Mulgund et al, 2001)):

- Define a set of commonly-used element and division formations as well as the underlying tactical maneuvers and attack tactics.
- Develop a set of principles for aggregating the small formation tactics for large MvN engagements, and implement a method for doing so in the GA software. To illustrate, consider a team consisting of four aircraft. Using only the fighting wing and double-attack, the possible team formations are shown in Figure 3 (assuming both elements use the same two-ship formation). A similar approach can be used to develop large division formations from smaller 2-ship and 4-ship groupings.
- Use the resultant formation tactics to drive the engagement, and evaluate the results via the performance metric generator.
- Search for the best MvN engagement tactics so as to optimize the performance metrics.

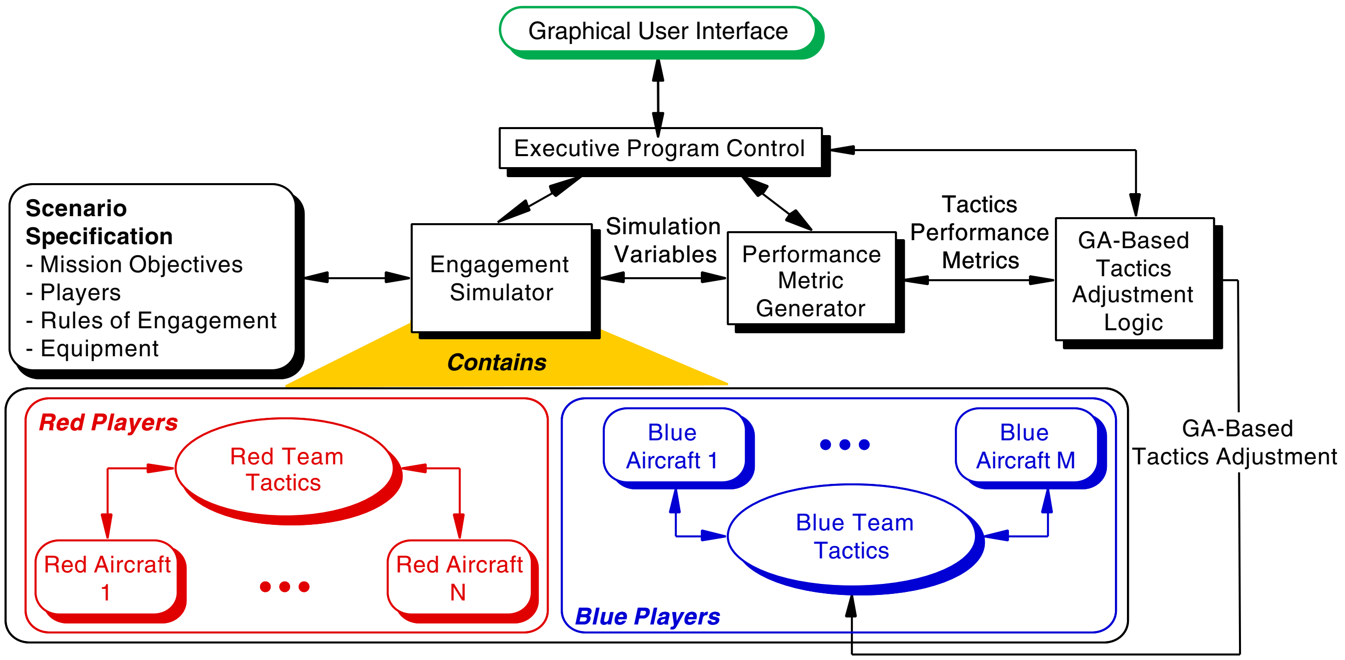


Figure 2. Combat Tactics Design Environment

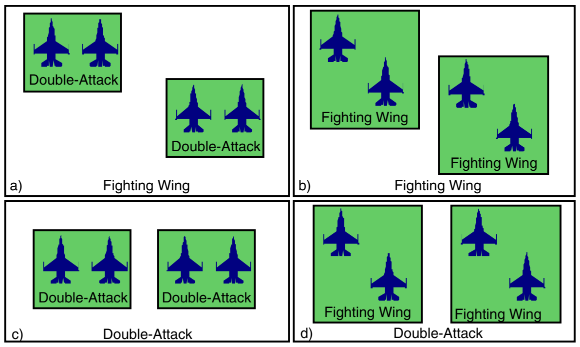


Figure 3. Potential Four-Aircraft Formations.

3.3 Intelligent Systems for Design

Aerospace systems are complex in nature and their design interdisciplinary. For example, design optimization conducted individually on subsystems such as wing, propulsion, and automatic control will not integrate without extensive redesign. Even individual design of these subsystems require interaction of several subsystems. Intelligent systems can help in many ways to enhance this experience. Several researchers have shown the benefits of neural networks and fuzzy logic for representation of known data and designer expertise. Genetic algorithms have been routinely used for searching through large spaces with several constraints and subsystem interactions. A generic representation of the design philosophy using intelligent system features are shown in Figure 4. In the design application presented next, Neural networks are used for representation of data (modeling) and genetic algorithms are used to search for the type of inputs to be used for the neural networks.

3.3.1 Application: Engine Estimator Design

One of the common objectives of aircraft engine control is to enhance engine performance under deteriorated conditions. To maximize engine performance efficiently under degraded conditions, a fault tolerant engine control scheme can be applied. The first step to implement the fault tolerant engine control architecture is developing an engine performance estimator. This application focuses on designing an engine performance estimator using a combination of a genetic algorithm (GA) and a radial basis function neural network (RBFNN) for the implementation.

Generally, traditional engine performance estimators, such Kalman filter estimator, involves intensive computational procedures because of engines' physical complexity which requires a large number of measurements to be taken and processed. To overcome computational complexity, model estimation using neural networks has emerged. Neural network-based model estimation has been applied to areas such as optics, robotics, and system control. Attracted by the advantages of neural networks, the recent studies of fault tolerance have employed neural network architectures. In these studies, the input selection is executed by simple inspection of data files. This manual inspection can be replaced by automatic inspection using GAs. The need for the design that involves selecting the best inputs is driven by the cost of sensors. In this case fewer the sensors the better the cost.

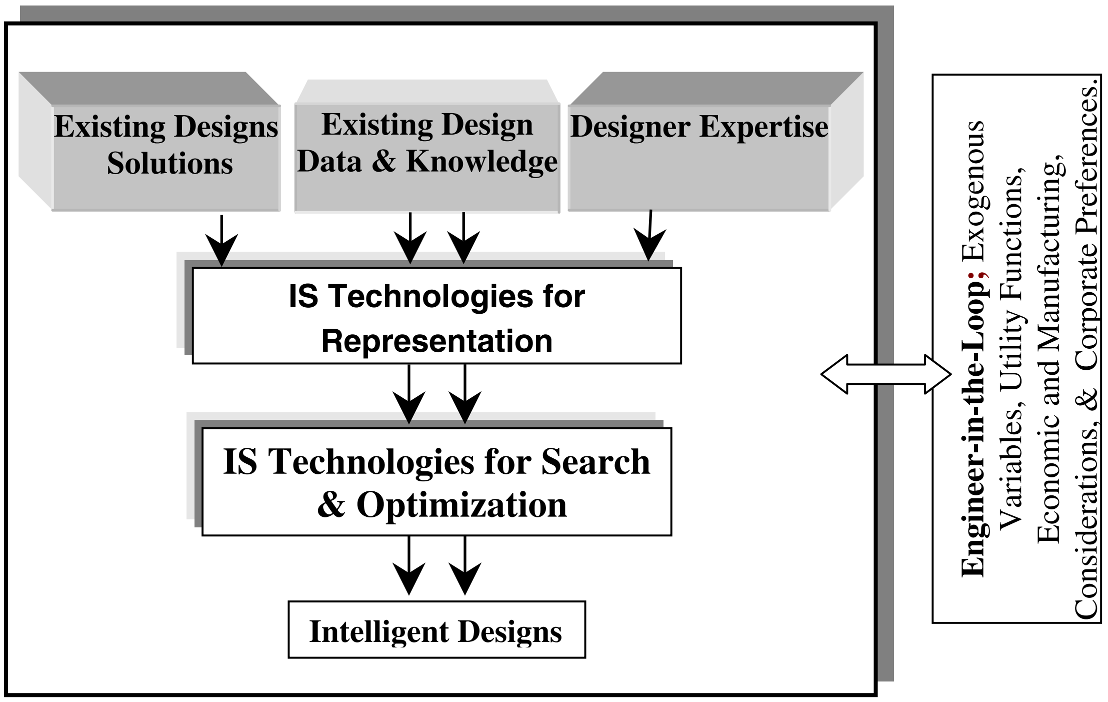

Existing Designs Solutions
Existing Design Data & Knowledge
Designer Expertise
IS Technologies for Representation
IS Technologies for Search & Optimization
Intelligent Designs
Engineer-in-the-Loop; Exogenous Variables, Utility Functions, Economic and Manufacturing, Considerations, & Corporate Preferences.

Figure 4. An Intelligent Design Approach

To apply GAs to this problem, the parameter performance can be evaluated by an objective function (performance index or PI). Regularly, a quadratic error measure is used for the objective function. The error is defined as the difference between desired performance and the current design's performance.

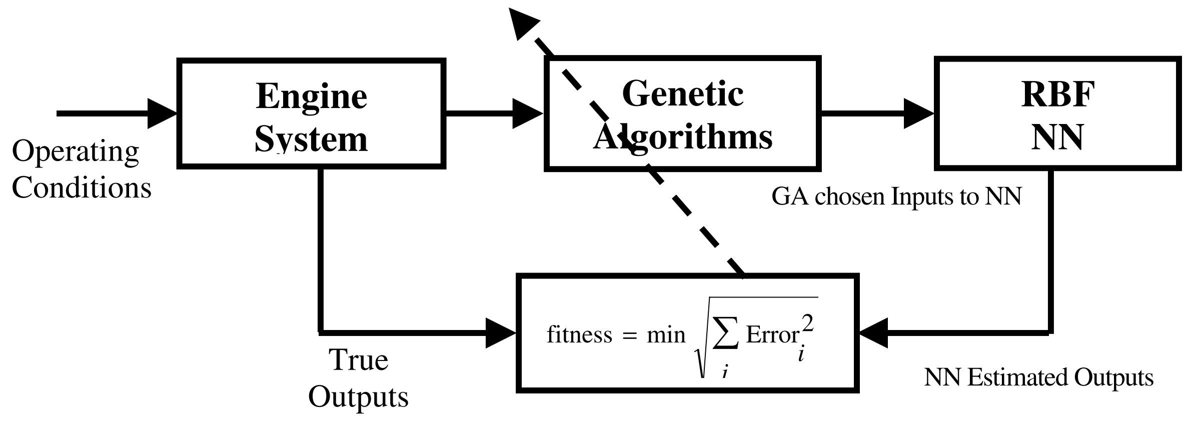


Figure 4. GA-RBFNN Architecture

GA-RBFNN Architecture: The GA-RBFNN architecture is shown in Figure 5. The Genetic Algorithms picks "n" inputs out of the possible 21 variables shown in table 1. The quantity "n" is a user choice. For example, if n=5, the GA has the following chromosome representation

10110	10011	11011	10111	00110
Input 1	Input 2	Input 3	Input 4	Input 5

We use a 5 bit representation to cover the 21 possibilities. Since 5 bits give us 32 possibilities, there will be multiple mapping for some of the variables.

The data from the first 150 seconds (300 data points with 0.5 seconds of step size) is used for the RBFNN's training. For the GA fitness, all the 500 data points are utilized. This way, we incorporate a combination of training and validation data into the GA fitness.

The RBFNN produces one output, either compressor **stall margin** (***sm27***) or **thrust** (***fn***). The output is then compared with a desired performance of the corresponding performance measure (desired *sm27* or *fn*). The difference between the output from the RBFNN and the desired performance becomes the estimation error. Squaring the error and summing it up over the time range (500 data points) results in a fitness function of the entire system. For more details, please see Krishnakumar and Hachisako (2000).

3.4 Intelligent Systems for Control

There are two main aspects to intelligent control: (1) the "intelligence" to analyze the changing environment; and (2) the resources to respond to the changing environment. Intelligence connotes the analytical ability to comprehend and react to the changing environment. Resources connote the physical components of the system that are necessary to react to the environment. In this work, we concentrate on the need to harvest and interpret the information from the network of sensors and to apply it for control such that good performance is maintained under any of the following situations:

- Loss of control due to failure

- Aircraft characteristics change due to damage (center of gravity, inertia, etc.)
- Changing operating conditions (altitude, mach, etc.)
- Environmental effects due to wind and turbulence

Intelligent control applications focus on control problems that otherwise cannot be solved, or cannot be solved in a satisfactory way by traditional control techniques alone. Intelligent control as practiced today encompasses many fields from conventional control such as optimal control, robust control, stochastic control, linear control, and nonlinear control, as well as the more recent fuzzy, genetic, and neuro-control technologies. In the next subsection, intelligent control is classified based on the idea of self-improvement as the goal toward higher levels of intelligence.

3.4.1 Application: Levels of Intelligent Control

In a general sense, an intelligent controller design can be stated as the following:

given the dynamic system as:

$X(t+1) = f(X(t),U(t),t)+\eta$; where X are the state variables, U is the control vector, and η is an unknown disturbance

a set of goals generated as a function of time as

$Xg(t+1) = g(Xg(t),X(t),t)$

a performance measure as:

$J(t+1)= \Im(M(Xg(t),X(t),U(t),t))$; where $\Im$ is an operator (usually summation over T)

and a planning function as

$P(t+1) = p(X(t),P(t),t,\nu)$; where ν is system faults and emergencies

the intelligent controller needs to arrive at a control, U(t), such that the system (in the order of priority)

- is locally stable (includes handling quality of the system)
- follows closely the desired path (closeness defined by the performance measure)
- constantly optimizes long-term and short-term goals
- reacts to changing environments by properly adapting the planning functionality.

Over the past decade, several innovative control architectures utilizing the intelligent control tools have been proposed. We believe that a practical way to accommodate the above needs is to approach the system as having various levels of capabilities for self-improvement. Self-improvement is an important goal of human intelligence. Self-improvement is quantifiable and measurable in various ways. By defining intelligent Control with various levels of intelligence, the definition is left 'open ended' such that it will not become obsolete, and it will accommodate easily the innovations that will inevitably come from the contributions of such fields as cognitive science, computer hardware, sensors and actuators, learning theory, and control architectures.

KrishnaKumar (1997) has proposed a classification scheme based on the ability of the control architecture for self-improvement (see Table 1). The classification scheme divides the control architectures among levels of intelligent control (LIC). For instance, most of the proposed architectures can be divided among level 0, level 1, level 2, and level 3 intelligent control schemes. Based on this classification scheme, several seemingly differing control architectures can be looked at as achieving similar goals.

Table 1. The Levels of Intelligent Control

Level	**Self improvement of**	**Description**
0	Tracking Error (TE)	Robust Feedback Control: Error tends to zero.
1	TE + Control Parameters (CP)	Adaptive Control: Robust feedback control with adaptive control parameters (error tends to zero for non-nominal operations; feedback control is self improving).
2	TE+CP+ Performance Measure (PM)	Optimal Control: Robust, adaptive feedback control that minimizes or maximizes a utility function over time.
3	TE+CP+PM+ Planning Function	Planning Control: Level 2 + the ability to plan ahead of time for uncertain situations, simulate, and model uncertainties.

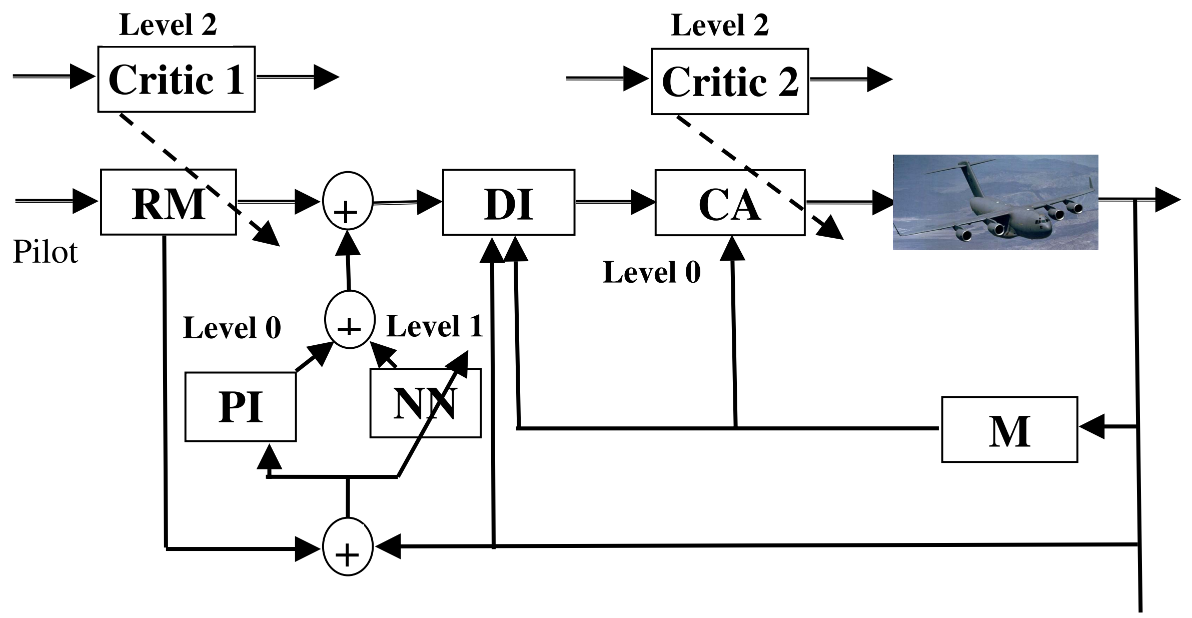
Level 0 Intelligent Control -- A Robust Controller: Self-improvement of Tracking Error (TE) is an important goal of many control techniques. To achieve this, one designs robust feedback controllers with constant gains that improve the error as time goes to infinity. We consider this as "Level 0 Intelligent Control".

Level 1 Intelligent Control -- An Adaptive Controller: Self-improvement of control parameters towards the goal of achieving better tracking error or some error oriented goal, is the next level in intelligent control. We consider this as "Level 1 intelligent Control". This level is essentially a robust feedback controller with adaptive parameters that helps the error tend to zero for non-nominal operations and feedback controller is self improving.

Level 2 Intelligent Control -- An Optimal Controller: Self-improvement of an estimate of the performance error (or some measure of performance over time) towards the goal of minimization or maximization of an utility function over time (error tends to zero and a measure of performance is optimized) is the next level in intelligent control. We consider this as "Level 2 Intelligent Control". This level is essentially a robust feedback controller with adaptive parameters that helps the error tend to zero for non-nominal operations, feedback controller is self improving, and a measure of performance that is self-improving is optimized over time.

Level 3 Intelligent Control -- A Planning Controller: In addition to level 2 capabilities, Level 3 Intelligent Control includes self- improvement of planning functions. Planning functions include contingency planning, planning for emergencies, planning for faults, etc. These planning functions could be static for Level 2 but needs to be self-improving for Level 3.

Figure 6 presents the implementation of a Level 2 Intelligent Control on a C-17 test bed at NASA Ames research center. The levels as outlined earlier are labeled in the figure. It should be noted that Level 0 is non-adaptive whereas Level 1 is adaptive. Level 1 is non-optimal whereas Level 2 is optimal.



<u>**RM:** Reference Models:</u> The pilot commands roll rate and aerodynamic normal and lateral accelerations through stick and rudder pedal inputs. These commands are then transformed into body-axis rate commands, which also include turn coordination, level turn compensation, and yaw-dampening terms. First-order reference models are used to filter these commands in order to shape desired handing qualities.
<u>**PI** Error Controller:</u> Errors in roll rate, pitch rate, and yaw rate responses can be caused by inaccuracies in aerodynamic estimates and model inversion. Unidentified damage or failures can also introduce additional errors. In order to achieve a rate-command-attitude-hold (RCAH) system, a proportional-integral (PI) error controller is used to correct for errors detected from roll rate, pitch rate, and yaw rate (p, q, r) feedback.
<u>**NN:** On-Line Learning Neural Networks:</u> The on-line learning neural networks work in conjunction with the error controller. By recognizing patterns in the behavior of the error, the neural networks can learn to remove biases through control augmentation commands. These commands prevent the integrators from having to windup to remove error biases. By allowing integrators to operate at nominal levels, the neural networks enable the controller to provide consistent handling qualities.
<u>**DI:** Dynamic Inversion:</u> Dynamic inversion is based upon feedback linearization theory. No gain-scheduling is required, since gains are functions of aerodynamic stability and control derivative estimates and sensor feedback. To perform the model inversion, acceleration commands are used to replace the actual accelerations in the quasi-linear model. The model is then inverted to solve for the necessary control surface commands
<u>**CA:** Control Allocation:</u> An optimal control allocation technique is used to ensure that conventional flight control surfaces will be utilized under normal operating conditions. Unconventional flight control surface allocations are only utilized when the primary flight control surface commands exceed the known limits of deflection. For example, in the longitudinal axis pitch rate control is normally provided through symmetric elevator deflections. If this command should saturate, then the remaining portion of the command is applied to symmetric ailerons. If the symmetric aileron command saturates, then the remaining portion of that command is applied to symmetric thrust. The symmetric aileron command is limited, by the differential aileron command, so that secondary pitch control does not interfere with primary roll control.
<u>**M:** Pre-Trained Neural Network Model:</u> A Levenberg-Marquardt (LM) multi-layer perceptron is used to provide dynamic estimates for model inversion. The LM network is pre-trained with stability and control derivative data generated by a Rapid Aircraft Modeler. This block can be replaced by other on-line derivative (parameter) estimation techniques.
<u>**Critic 1 and 2:**</u> Adaptive Critics are utilized to optimize the control allocation scheme and to shape the reference model dynamics in the event of a failure.

Figure 6. Level 2 Intelligent Flight Control System

3.5 Intelligent Systems for Training

Many characteristics of intelligent systems are readily applicable to training. Although training entails many different areas of aerospace engineering, our experience has been with pilot training. Figure 7 presents an implementation of an automated hover training system. This system was implemented in a fixed-base simulation facility and was shown to provide basic hover training skills with no human intervention. The neural network based intelligent system adapts the helicopter dynamics to the student pilot and automatically changes the dynamics of the helicopter as learning progresses. For more details, please see KrishnaKumar et al (1994)

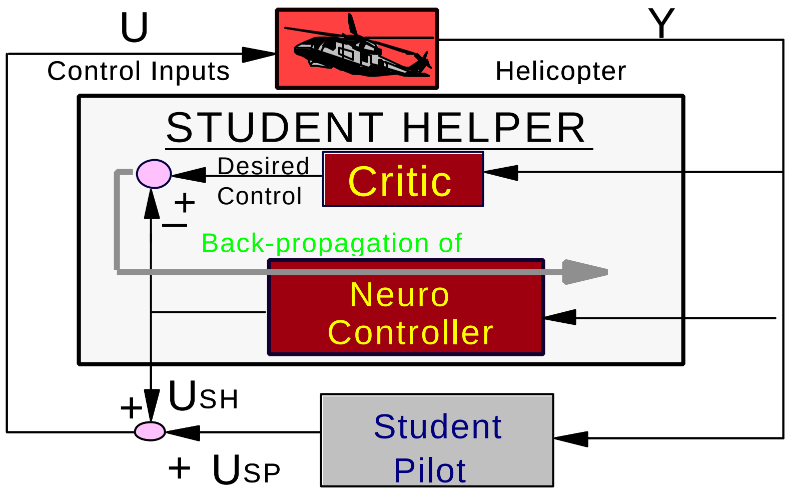


Figure 7. Automated Training Using an Intelligent System

3.6 Intelligent Systems for Autonomous Agents

An Agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors. An autonomous agent is shown in Figure 8. A rational agent does the right thing. How successful the agent is measured using a performance measure that is defined using goals. It is assumed that the agent has some percepts to satisfy observability or sensing requirements. Knowledge of the environment (Model) and actions that can be performed (Controllability) are implicit to the agent. Some of these are pre-programmed and some are learnt on-line. An ideal rational agent optimizes the measure of performance. Intelligent systems are used in different capacities such as knowledge representation, adaptability, agent-to-agent interaction, planning, and optimization. For more details on agents, see reference (Russell and Norvig, 1995).

3.7 Other Areas of Application

Intelligent systems can be embedded in almost any application where information needs to be processed to provide a usable output. Some areas that were not specifically addressed earlier include:

- Intelligent Systems for Decision Making
- Intelligent systems for planning and scheduling
- Intelligent systems for health management
- Intelligent systems for prediction

- Intelligent systems for knowledge discovery

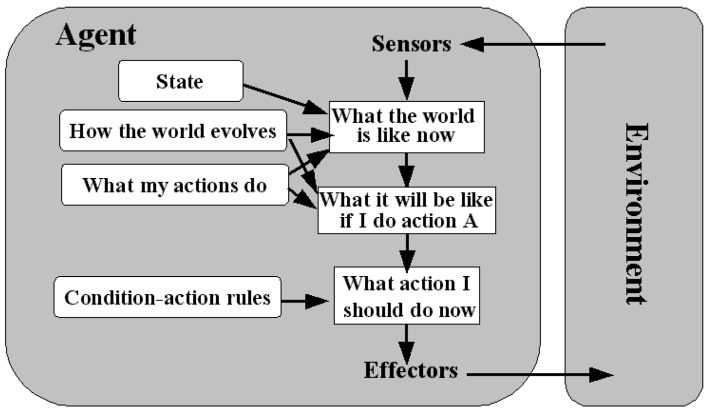


Figure 8. Intelligent Autonomous Agent Architecture (Russell and Norvig, 1995).

4 Future of Intelligent System Applications

Intelligent systems provide a means by which complex problems can be addressed and in many cases solved to a satisfactory level. The benefits can be categorized as either immediate or in-the-future. The immediate benefits are in applications of intelligent systems to areas where existing methodologies are marginally satisfactory and incorporating intelligent systems provide better efficiencies and solutions. Examples include: inverse design, adaptive control, optimal search, etc. The future benefits are more exciting. Intelligent systems will help formulate and solve problems such as brain-like control and decision-making, human-machine collaborative work, instant speech recognition, thought control, human capability enhancement, advanced pattern recognition, real-time scheduling, automated design, intelligent maneuvering for unmanned aerial vehicles, and autonomous security search. On a cautious note, intelligent system researchers should examine closely the analytical framework of their innovations. Analytical framework along with standardization has been shown to be important for the ultimate ticket to real implementations in aircraft applications.

5 Acknowledgement

Part of the work reported here was conducted while the author was a faculty at the University of Alabama. All of the students and colleagues who worked on these projects are hereby acknowledged for their contributions to the work highlighted above.

6 References

Holland, J. H. (1975). *Adaption in Natural and Artificial Systems*, University of Michigan Press.

Howard, R. A. (1960). *Dynamic Programming and Markov Process*. MIT Press, Cambridge, Massachusetts.

KrishnaKumar, K. & Hachisako, Y. (2000). *Two Applications of Genetic Algorithms, GA for optimization in Aeronautics and Turbomachinery*, KrishnaKumar, K. (1997). *Levels of Intelligent Control*, AIAA Tutorial at New Orleans, LA, August.

Krishnakumar, K. (2000). *Intelligent Control for Aerospace Systems*, Global Aerospace Technology, World Market Research Center.

Krishnakumar, K.S., Kulkarni, N., “Equivalent Dynamic Fuzzy Controllers and their Application to Engine Control”, 34th AIAA Joint Propulsion Conference, Cleveland, Ohio, July, 1998.

KrishnaKumar, K., Sawhney, S., and Wai, R. (1994). *Neuro-controllers for adaptive helicopter hover training*, IEEE Transactions on Systems, Man, and Cybernetics, August, 1994 Vol. 24, No. 8. pp. 1142-1152.

Kurzweil, R. (1990). *The Age of Intelligent Machines*. MIT Press, Cambridge, Massachusetts.

Luger, G. F. and Stubblefield, W. A. (1992). *Artificial Intelligence*, Benjamin-Cummings, Redwood City, California

Mulgund, S., Harper, K., Krishnakumar, K., & Zacharias, G. (2001). *Air Combat Tactics Optimization Using Genetic Algorithms*, AIAA Journal of Guidance Control and Dynamics,

Rich, E. and Knight, K. (1991). *Artificial Intelligence,* McGraw Hill, New York.

Russell, S. and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*, Prentice Hall, New Jersey.

Schalkoff, R. J. (1990). *Artificial Intelligence: An Engineering Approach*, McGraw Hill, New York.

Shaw, R. (1988). *Fighter Combat: Tactics and Maneuvering*, Annapolis, MD: Naval Institute Press.

The ADEPT Framework for Intelligent Autonomy

Dr. Michael Ricard / Dr. Stephan Kolitz
The Charles Stark Draper Laboratory, Inc.
555 Technology Square
Cambridge, MA 02139 USA

Abstract

This paper describes the design and implementation of Draper Laboratory's **All-Domain Execution and Planning Technology** (ADEPT) architecture for *intelligent autonomy*. Intelligent autonomy is the ability to plan and execute complex activities in a manner that provides rapid, effective response to stochastic and dynamic mission events. Thus, intelligent autonomy enables the high-level reasoning and adaptive behavior for an unmanned vehicle that is provided by an operator in man-in-the-loop systems.

Draper's intelligent autonomy has architecture evolved over a decade and a half beginning in the mid 1980's [3, 4, 6 and 12] culminating in an operational experiment funded under DARPA's Autonomous Minehunting and Mapping Technologies (AMMT) unmanned undersea vehicle program [15]. ADEPT continues to be refined through its application to current programs that involve air vehicles, satellites and higher-level planning used to direct multiple vehicles. The objective of ADEPT is to solidify a proven, dependable software approach that can be quickly applied to new vehicles and domains.

The architecture can be viewed as a hierarchical extension of the sense-think-act paradigm of intelligence and has strong parallels with the military's Observe-Orient-Decide-Act (OODA) loop [14]. The key elements of the architecture are planning and decision-making nodes comprising modules for situation assessment, plan generation, plan implementation and coordination. A reusable, object-oriented software framework has been developed that implements these functions. As the architecture is applied to new areas, only the application specific software needs to be developed.

This paper describes the core architecture in detail and discusses how this has been applied in the undersea, air, ground and space domains.

Introduction

In recent years, there has been a spectrum of development programs for air, space, ground, and underwater vehicles with the desire for increasing degrees of autonomy. Enabled by advances in intelligent autonomy, autonomous vehicles will ultimately be employed by the military as a force multiplier and as a means of reducing risk to military personnel and in nonmilitary applications in remote and hazardous locations, including search and rescue during a fire or natural disasters. For example, future missions of autonomous Uninhabited Combat Air Vehicles (UCAVs) may include lethal air to ground missions for suppression of enemy air defenses (SEAD), intelligence, surveillance, reconnaissance and targeting (ISRT), and logistics re-supply. To successfully carry out these classes of missions, UCAVs will require highly adaptive autonomous mission planning and control, real-time obstacle avoidance of both static and dynamic obstacles and path planning for high speed flight in complex terrain.

Here *Intelligent Autonomy* refers to a vehicle's capability of sensing its own state and the state of its environment, and, based on this situational awareness, of planning and executing actions that are designed to achieve specific objectives under defined constraints. Thus, intelligent autonomy requires the development of automated planning and control systems that plan the vehicle's mission prior to its deployment, control the vehicle's state during the execution of the planned mission and replan the mission in order to accommodate unanticipated events that may arise during mission execution.

As explained below, the challenges that intelligent autonomy must address are daunting: 1) developing and executing plans of activities that meet mission objectives and honor constraints, 2) dealing with uncertainty, and 3) providing a capability for dynamically adjusting a vehicle's plan in real time.

First, multiple (and often conflicting) objectives must be pursued in the face of a variety of both implicit and explicit constraints. Representative mission objectives for an Unmanned Combat Air Vehicle (UCAV) might include surveillance, reconnaissance, resupply, support and strike [10]. Implicit constraints are constraints that are imposed by the vehicle design (e.g., its fuel carrying capacity, weapon stores carrying capacity, performance envelope and subsystem capabilities). Explicit constraints are those that may be imposed by a higher planning authority including:

1. a required probability of survival or mission success,
2. time or ordering constraints that may be imposed on the pursuit of specific mission objectives,
3. navigation constraints, and
4. constraints imposed by the time available to plan.

Second, the fact that intelligent autonomy must accommodate the significant uncertainty in the system's knowledge of the current and future "state of the world" (i.e., state of the vehicle and its environment) also contributes to the complexity of the mission planning problem. Because of this uncertainty, the formulation of detailed plans far into the future or planning in advance for all possible contingencies is generally a futile exercise. As a consequence, the ability to replan onboard the mission is essential for autonomous vehicle applications, especially in poorly characterized environments where long endurance or a high probability of mission success is required.

The third challenge, to replan the mission in real time due to changes in the environment, commander's intent, changing opportunities, etc., adds another layer of complexity to the mission planning problem. This is due not only to the uncertainty with respect to the future state of the world that inevitably prevails at the time the mission is originally planned, but also to unforeseen, externally influenced events that can arise during the execution of the mission requiring a response in real time. For example, a higher planning authority may redefine the mission objectives or alter the constraints that are imposed on the pursuit of those objectives, the mission environment (threats, weather, etc.) might change, the vehicle might sustain failures or damage, or opportunities to accomplish additional objectives might arise.

Draper has designed the core ADEPT architecture to meet these intelligent autonomy challenges. Draper's ADEPT architecture is a hierarchical, closed-loop, real-time mission and trajectory planning capability for autonomous vehicles. This paper describes the development of ADEPT, and its application to a variety of domains. The paper is organized as follows. Section 2 discusses the functional components of the architecture and its roots from which it was developed, implemented and deployed in the highly successful DARPA Autonomous Minehunting and Mapping Technology (AMMT) Program. Section 3 describes the object–oriented design and implementation. Section 4 discusses the diverse domains employing this architecture and software.

Architecture Functional Description

ADEPT is based on Draper's core approach to autonomous mission planning systems [3]. The first fielded *operational implementation* of this core architecture was for DARPA's Advanced Minehunting and Mapping Technologies program. Additional capabilities have been introduced under both corporate sponsored research and externally funded programs. To make the solution of complex problems tractable, the problems are decomposed into simpler, decoupled subproblems that can be solved (nearly) independently.

Temporal hierarchical decompositions are often employed to simplify the implementation of the solution to real-time, closed-loop planning problems [4, 8]. These decompositions are characterized by higher levels that create plans with the greatest temporal scope (longest planning horizon) but with the least detail. At lower levels, the planning horizon becomes shorter (nearer term), but the level of detail of planned activities increases. The less detailed plans at the higher levels coordinate or guide the generation of solutions generated at the lower levels.

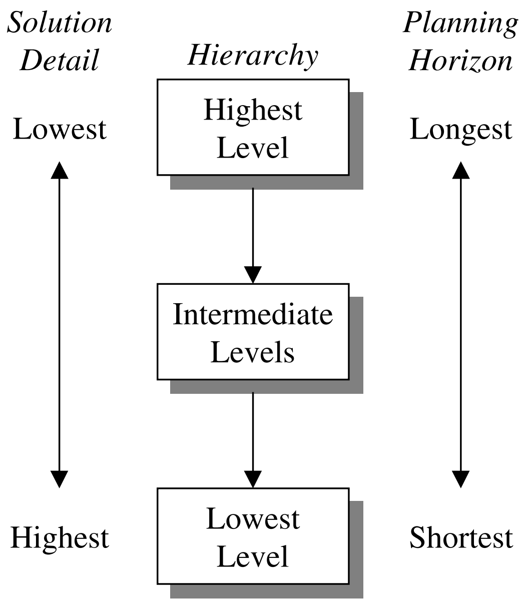


Figure 1: Characteristics of Solutions at Various Levels of the Hierarchy

Indeed, planning actions over extended periods of time at a high level of detail is *futile* because detailed actions planned on the basis of a specific prediction of the future may become obsolete well before they are to be executed due to an inability to accurately predict the future and *impractical* because the computational resources required to develop detailed plans for complex objectives over extended periods of time may be prohibitive.

Figure 2 shows the details of the functions performed at each level of the architecture in Figure 1. The key elements are modules for situation assessment, plan generation, plan implementation, and coordination. Figure 2 shows plan implementation providing input to the *system to be controlled*. Note that in the three tier hierarchical system shown in Figure 1, the higher level's plan implementation output consists of planning inputs to the next tier down.

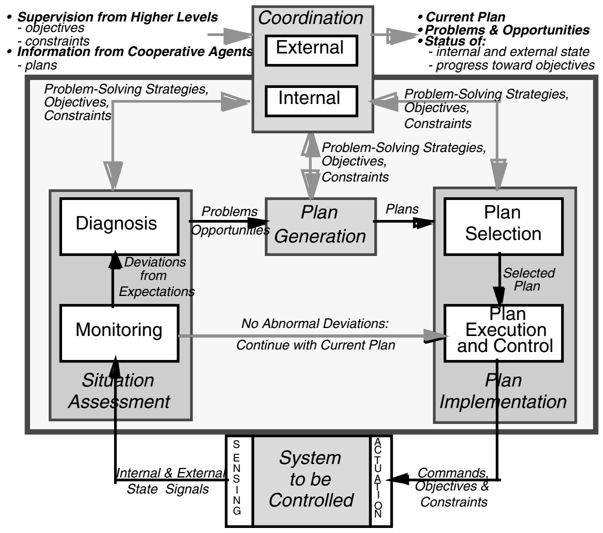

• Supervision from Higher Levels
- objectives
- constraints
• Information from Cooperative Agents
- plans
Coordination
External
Internal
• Current Plan
• Problems & Opportunities
• Status of:
- internal and external state
- progress toward objectives
Problem-Solving Strategies, Objectives, Constraints
Problem-Solving Strategies, Objectives, Constraints
Problem-Solving Strategies, Objectives, Constraints
Diagnosis
Problems
Opportunities
Plan Generation
Plans
Plan Selection
Deviations from Expectations
Selected Plan
Monitoring
No Abnormal Deviations:
Continue with Current Plan
Plan Execution and Control
Situation Assessment
Plan Implementation
Internal & External State Signals
SENSING
System to be Controlled
ACTUATION
Commands, Objectives & Constraints

Figure 2: Functional Decomposition of an Autonomous System

The *Monitoring Module* validates best estimates of the sensed data and monitors the operation of the system being controlled, as well as the environment in which it is operating, to detect departures from expectations. Any departure could represent a developing problem or a new opportunity, and is passed along to the

Diagnosis Module for interpretation. If no departures are observed, the Plan Execution Module continues to execute the current plan.

The *Diagnosis Module* analyzes departures identified by the Monitoring Module to determine their root cause or, alternatively, their impact on the capabilities of the system being controlled. The Diagnostic Module then initiates the actions required to respond to the root causes or changes in capabilities. If root causes cannot be determined, the Diagnostic Module attempts to reconfigure the system being controlled to retain as much of the systems capability as possible. The diagnosis of what has happened is passed to the Plan Generation Module to determine if the current plan needs to be modified to accommodate the changed circumstances.

The *Plan Generation Module* modifies the current plan when circumstances identified by the Diagnosis Module require such modifications. A new plan may be required in response to either problems or opportunities that have been identified by the Diagnosis module. In support of planning by a superior planning level or in a decision support role for a human planner, it may be desirable for the Plan Generation function to create a variety of plans that trade off among different levels of constraint and/or different objectives. Furthermore, a variety of algorithms may be applied in generating plans and the choice of algorithm is guided by the strategy input. For example, a quick heuristic may be required if a new plan is needed immediately to accommodate a serious (potentially mission or safety critical) problem that has been diagnosed. In other situations, it may be acceptable to continue to pursue the current plan while a more considered search of the plan space is executed in an attempt to refine the current solution to include additional opportunities or to accommodate minor degradations in the capabilities of the system-to-be-controlled.

The *Plan Selection Module* evaluates the utility of the revised plan relative to the current plan (and any other previously generated plans) to determine which plan to execute. A variety of plans may be generated, each based on one of multiple objectives, and given a strategy for selecting among a set of plans, Plan Selection makes the choice of a single plan to be executed. A change in plan may not be warranted if there is no plan in the set of generated plans whose value sufficiently exceeds the value of the current plan. The interpretation of "sufficiently exceeds" is made in the context of strategy inputs from the coordination function. The selected plan is passed to the Plan Execution and Control Module for execution.

Given the current state of the system, the *Plan Execution and Control Module* interprets the current (or selected) plan and issues commands to a subordinate planning level to guide its planning or, for the lowest level of planning, to the physical system-to-be-controlled. Execution of these "set-point" commands results in the pursuit of the current plan. Note that even when there are no detected abnormal deviations, under normal operations there typically will be minor deviations from the expected state. Thus, in addition to providing set-point commands for the pursuit of the current plan, an auxiliary role of the Plan Execution and Control function is to determine plan "perturbation" control commands that will attempt to correct for the range of normally expected deviations of the actual system state from the planned state.

The *External Coordination Module* provides an interface between the system being controlled and other control or controlled elements, which include human operators or users as well as other systems with which the system being controlled interacts. It allows for the receipt of new instructions, which can take the form of new objectives to be pursued and/or new constraints to be imposed on the objectives being pursued. It also provides for reporting out the current plan that is being pursued, the progress that has been made in the execution of that plan, the current state of the system, diagnostic information related to any problems that may have been encountered, and/or any new opportunities that have arisen along the way. Thus, External Coordination is responsible for assembling and transmitting information, deciding: what to transmit, when to transmit it and to whom to transmit.

The *Internal Coordination Module* harmonizes the efforts of the Monitoring, Diagnosis, Plan Generation, Plan Selection and Plan Execution and Control Modules to ensure that

1. Operations proceed smoothly in the absence of any problems or new opportunities,
2. Problems are accurately diagnosed and that the appropriate corrective actions are taken,
3. New opportunities are recognized and appropriately exploited,

4. New plans, when needed, are generated in a timely manner,
5. The transition from one plan to another is effected in a seamless manner, and
6. The current plan is properly executed.

In effecting that harmony, Internal Coordination develops strategies for controlling the other individual modules

1. By monitoring the assessed situation including: progress toward the current solution and the state of both the system-to-be-controlled and the external environment and
2. By taking into consideration any plans developed by other agents and objectives and constraints input from higher level authorities. Included in those strategies are:
   - The criteria for deciding when replanning is required,
   - The time allocated to generating a solution and
   - Cost/objective functions and constraints to be employed in generating a solution.

In each of the functional modules, and each level in the hierarchy, intelligent decisions are required to allocate system resources. Nonetheless, most computational resources will be devoted to the Plan Generation Module. Each level requires a unique mathematical formulation accounting for the specific objective, available resources, and relevant constraints.

Software Architecture

The conceptual implementation of the ADEPT architecture has its roots in the AMMT software, shown in Figure 3. That software was written in C and implemented in a real-time, priority based UNIX environment. The system consists of four tasks:

1. Mission Planner
2. Mission Evaluator
3. Guidance Interface
4. Asynchronous Input Handler

In the context of the current ADEPT architecture, the monitoring and diagnosis functions take place in the Mission Evaluator task, the plan generation function takes place in the Mission Planner task and the plan execution function is in the Guidance Interface task. Since this software has evolved into the current ADEPT architecture, it will be insightful to briefly discuss its implementation.

The Mission Planner task operates at the lowest priority and is responsible for sequencing the activities of a single level in the hierarchy. It considers permutations of the current plan to see if a better plan exists.

The Mission Evaluator incorporates the output from the Planner task into the currently executing plan. It also controls the input to the planner. This input includes which level the planner task should be working with and when it should restart the planning process. Its most important job is to verify the safety and the feasibility of the currently executing plan. If it determined that the environmental conditions have changed so that the current plan is no longer safe, it invokes an immediate replan or provides reactive measures for the near term.

The Guidance Interface operates at the highest priority and is the planner's only interface with the fault tolerant processor. This task converts the “executing” activity into a guidance command that the vehicle's guidance system understands. It also monitors subsystem health, controls subsystems and reads environmental sensors.

The Asynchronous Input Handler is responsible for processing terrain and target messages and updating the map. Moreover, it responds to requests from the sonar and the host ship.

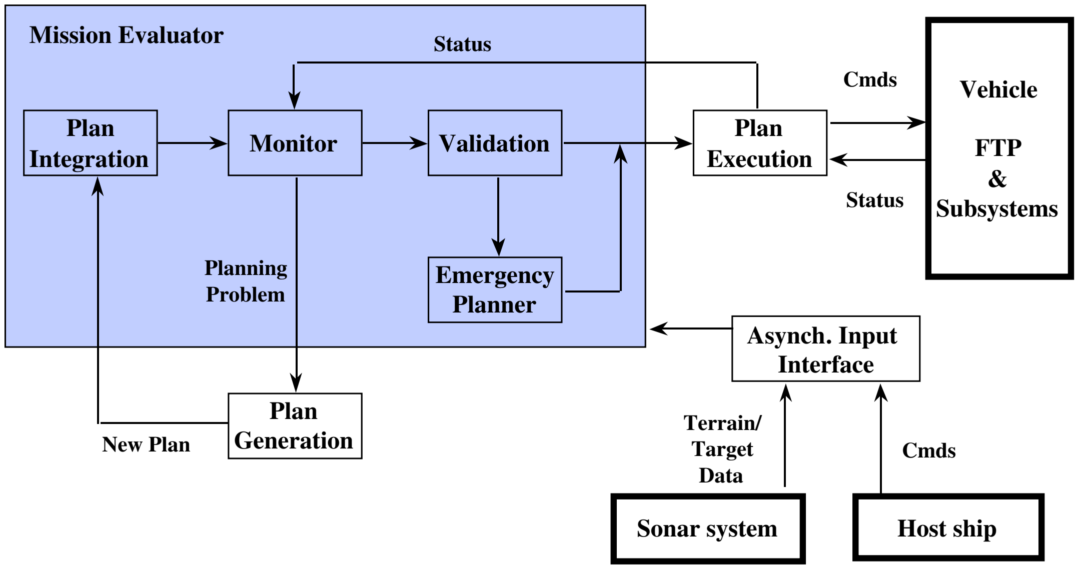


Figure 3: AMMT Software Architecture

The AMMT software has been used as the basis for extending the ADEPT concept into a reusable, object-oriented software framework. In the current architecture, the monitor and diagnosis tasks are separated to allow for a more modular implementation and to allow for the ability to execute the two tasks at different rates if necessary. Conceptually, a single level of the ADEPT architecture consists of an object that has a member function for each of modules that comprises one level of the architecture. Factory and bridge patterns [11] are employed to allow for ease of modification of the functions that are performed in each of the modules. The factory method provides an interface for creating planning objects for different domains while the bridge class separates the implementation from the abstraction. The base planner class itself then just executes the generic functions *monitor()*, *execute()* etc. without regard to the actual implementation being used. The use of the bridge pattern allows the actual *monitor()* function to be activated. Figure 4 shows the layout of the planner class with implementation classes for both an unmanned air vehicle (UAV) and an unmanned undersea vehicle (UUV).

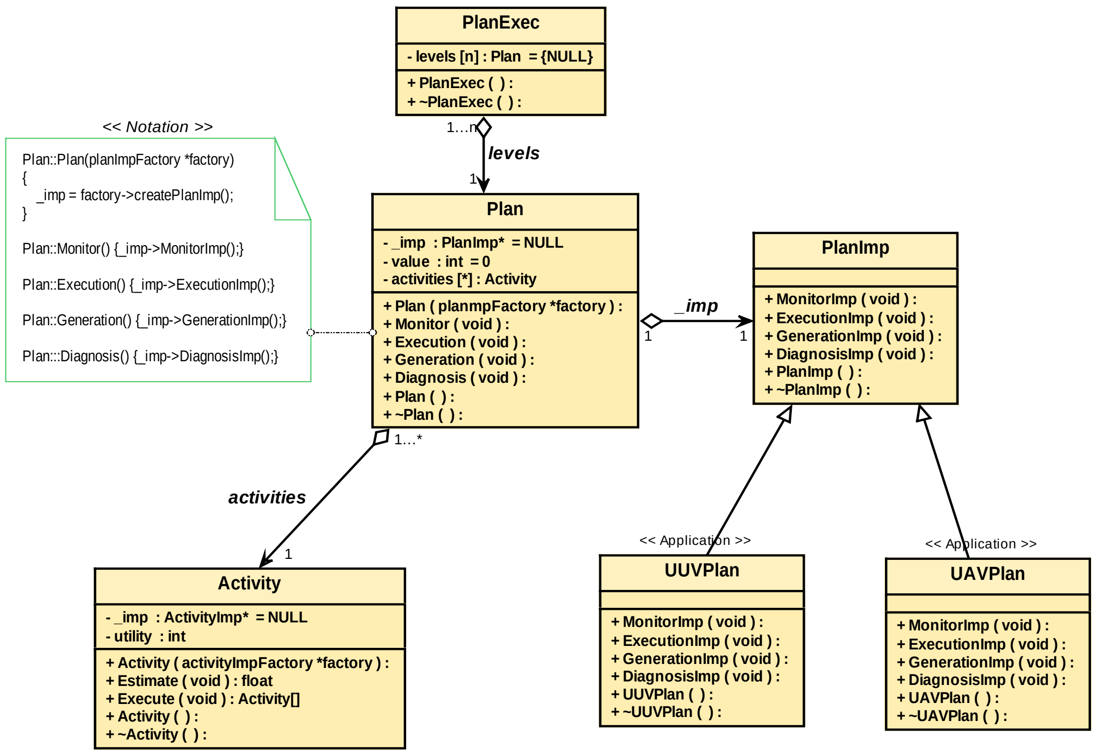

PlanExec
- levels [n] : Plan = {NULL}
+ PlanExec ( ) :
+ ~PlanExec ( ) :
1...n
levels
1
<< Notation >>
Plan::Plan(planImpFactory *factory)
{
_imp = factory->createPlanImp();
}
Plan::Monitor() {_imp->MonitorImp();}
Plan::Execution() {_imp->ExecutionImp();}
Plan::Generation() {_imp->GenerationImp();}
Plan:::Diagnosis() {_imp->DiagnosisImp();}
Plan
- _imp : PlanImp* = NULL
- value : int = 0
- activities [*] : Activity
+ Plan (planmpFactory *factory) :
+ Monitor (void) :
+ Execution (void) :
+ Generation (void) :
+ Diagnosis (void) :
+ Plan ( ) :
+ ~Plan ( ) :
_imp
1
1
PlanImp
+ MonitorImp (void) :
+ ExecutionImp (void) :
+ GenerationImp (void) :
+ DiagnosisImp (void) :
+ PlanImp ( ) :
+ ~PlanImp ( ) :
1...*
activities
1
Activity
- _imp : ActivityImp* = NULL
- utility : int
+ Activity (activityImpFactory *factory) :
+ Estimate (void) : float
+ Execute (void) : Activity[]
+ Activity ( ) :
+ ~Activity ( ) :
<< Application >>
UUVPlan
+ MonitorImp (void) :
+ ExecutionImp (void) :
+ GenerationImp (void) :
+ DiagnosisImp (void) :
+ UUVPlan ( ) :
+ ~UUVPlan ( ) :
<< Application >>
UAVPlan
+ MonitorImp (void) :
+ ExecutionImp (void) :
+ GenerationImp (void) :
+ DiagnosisImp (void) :
+ UAVPlan ( ) :
+ ~UAVPlan ( ) :

Figure 4: ADEPT Planner class, including 2 domain implementations

The software has been successfully demonstrated in cooperative operations with air vehicles and ground vehicles. This work was supported by Draper internal research and development funding and continues to be extended under externally funded programs. The ADEPT architecture has been embedded into the more encompassing vehicle/ground station architecture developed under internal funding and shown in Figure 5. The ADEPT code runs within the "Mission Planning and Control" block on both the ground station and the autonomous vehicles. While running on the ground station, it is often used to assist an operator in generating plan input objectives. To support this, the planning code can operate in conjunction with a simulated environment on the ground station. When the ADEPT architecture is controlling a vehicle, the communication link to the vehicle passes objectives to the ADEPT software running on-board.

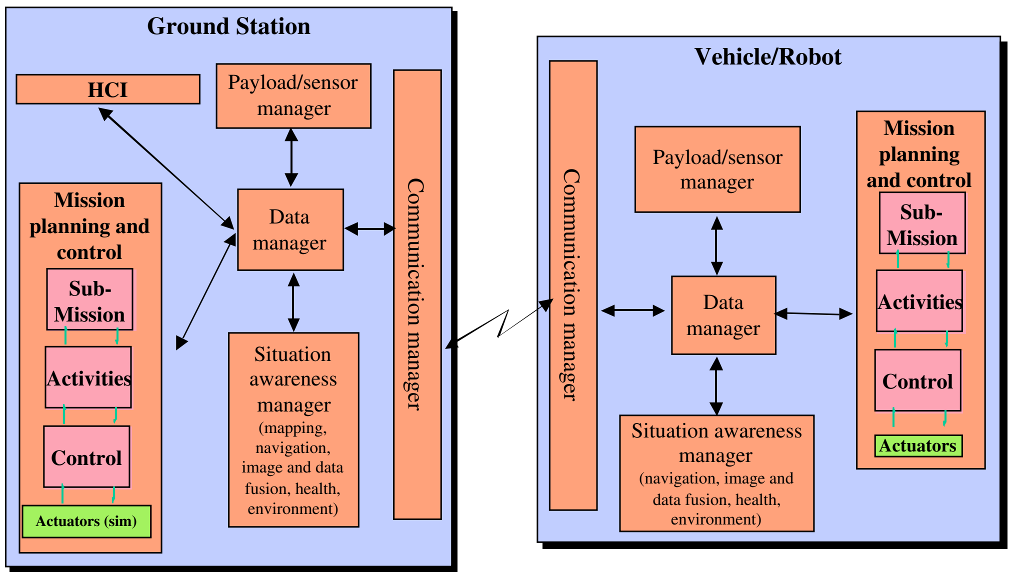

Ground Station
HCI
Payload/sensor manager
Mission planning and control
Sub-Mission
Activities
Control
Actuators (sim)
Data manager
Situation awareness manager
(mapping, navigation, image and data fusion, health, environment)
Communication manager
Vehicle/Robot
Communication manager
Payload/sensor manager
Data manager
Situation awareness manager
(navigation, image and data fusion, health, environment)
Mission planning and control
Sub-Mission
Activities
Control
Actuators

Figure 5: Draper's Autonomous Vehicle Architecture, including ADEPT for mission planning

Applications of ADEPT

The following sections summarize four programs supported by the ADEPT architecture. The first is an ONR program for cooperative multi-autonomous vehicle operations; the second is a NASA program for cooperative constellations of satellites and high altitude UAVs for earth observations. The final two represent higher-level control problems. They include a DARPA program for automating coordinated theater-wide air operations and a program to develop advanced air traffic flow management.

UCAV Intelligent Autonomy

ADEPT was used to provide the mission planning capability for a series of demonstrations performed for ONR, highlighting key technologies necessary to perform ship-based ISRT missions [2, 10]. The planned mission planning capabilities of the system included:

1. dynamic replanning to avoid pop-up threats in a threat-dense environment;
2. planning a two UCAV cooperative geolocation mission, followed by an unmanned ground vehicle (UGV) plan to the geolocated target; and
3. performing a search for a landing platform (surrogate for a ship in sea-state 3) and, working with automatic target recognition (ATR) and target tracking algorithms, maintaining lock on the landing site in order to perform an autonomous shipboard landing.

The first two of these capabilities have been demonstrated.

Supervisory and Autonomous Satellite Operations

ADEPT is being utilized in this NASA program [1] to meet key earth science objectives for rapid response to and tracking of environmental phenomena, and detailed study of the Earth through optimal allocation of resources within a sensor web that includes orbiting satellites and UAVs. Three autonomous mission planning tiers comprise the architecture. The highest, System Level, performs its computations at a centralized control center, and is responsible for allocating the ground based resources. The middle level allocates data collection tasks among individual sensor web elements to meet prioritized input objectives, using centralized or distributed planning. The lowest, Individual Platform Level, allocates the resources for the individual satellite or UAV.

JFACC (Joint Forces Air Component Command)

The ADEPT hierarchical architecture was employed on this DARPA program to plan and execute missions for five wings of aircraft over a seven-day campaign [7, 9]. The levels of the architecture are illustrated and the planning problem solved at each level is illustrated in Figure 6.

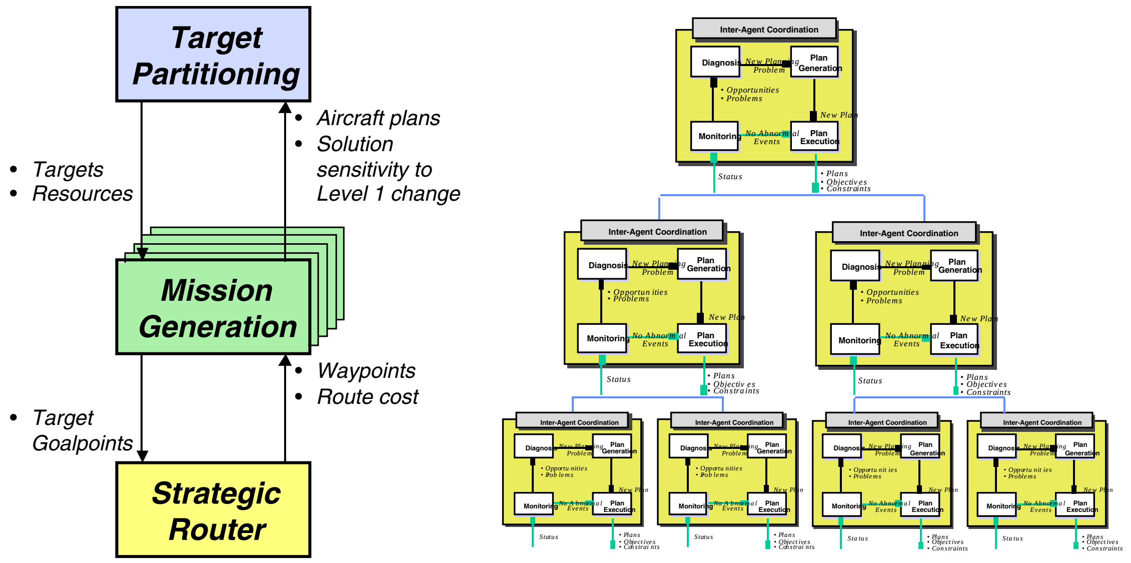
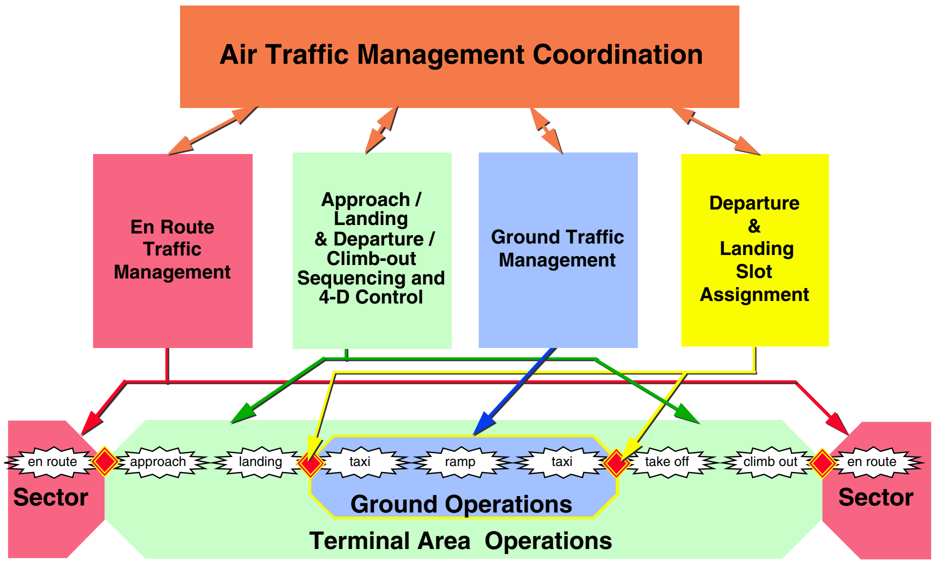


Figure 6: Air Operations Planning and Execution Hierarchy

The *Target Partitioning* level allocates targets and aircraft resources to *Mission Generation* Level. The Mission Generation level forms strike packages over time, assigning specific aircraft to specific targets, employing route costs developed by the *Strategic Router* Level. The Strategic Router Level optimizes paths from bases through tankers to target goalpoints and back, minimizes the route cost comprising a combination of attrition risk and cost of time. The ADEPT architecture allows a truly closed-loop system that is capable of replanning in response to system feedback and new target definitions at intervals as short as a few hours.

Air Traffic Flow Management

During the time ADEPT was evolving, Draper IR&D research on Air Traffic Flow Management (ATFM) was taking place. The ADEPT architecture was used in the development of advanced ATFM concepts and algorithms, some of which are described in [5] and further developed in [13]. ATFM is the top level in the hierarchical decomposition illustrated in Figure 8 and coordinates among planning and control functions at and near the airport and planning and control functions in the en route airspace.



Air Traffic Management Coordination
En Route Traffic Management
Approach / Landing & Departure / Climb-out Sequencing and 4-D Control
Ground Traffic Management
Departure & Landing Slot Assignment
en route
approach
landing
taxi
ramp
taxi
take off
climb out
en route
Sector
Ground Operations
Terminal Area Operations
Sector

Figure 7: Coordination of Air Traffic Flow Management

Acknowledgments

This material is based upon work supported by the National Aeronautics and Space Administration's Glenn Research Center (contract #NAS3-00163), the Defense Advanced Research Projects Agency (contracts #MDA972-94-C-0011 and F30602-99-C-0206) and the Office of Naval Research (UCAV demo program). The opinions, findings, conclusions and recommendations expressed in this paper are those of the authors and do not necessarily reflect the views or official policies, either expressed or implied, of the National Aeronautics and Space Administration, the Defense Advanced Research Projects Agency, the Office of Naval Research or the U.S. Government.

References

[1] Abramson, M., et al., "The Design and Implementation of Draper's Earth Phenomena Observing System (EPOS)," in *Proceedings of the AIAA Space 2001 Conference*, Albuquerque, NM, August 2001.

[2] Abramson, M.; Cleary, M.; Kolitz, S., "Steps Toward Achieving Robust Autonomy," in *Proceedings of the AAAI Spring Symposium*, Stanford, CA, March 2001.

[3] Adams, M. B., "Functional Analysis/Decomposition of Closed-Loop, Real-time Processes," in *AGARD Lecture Series 200: Knowledge-Based Functions in Aerospace Systems*, AGARD Document # AGARD-LS-200, November 1995.

[4] Adams, M.B., and Beaton, R. M., "Planning and Planning Management for Autonomous and Semi-Autonomous Vehicles," *Proceedings of the AGARD Guidance and Control Panel 51st Symposium*, Knowledge Based System Applications for Guidance and Control, Madrid, Spain, Sept. 18-21, 1990, AGARD-CP-474.

[5] Adams, M., Kolitz, S., Milner, J. and Odoni, A. R., "Evolutionary Concepts for Air Traffic Flow Management", ATC Quarterly, Vol.4. No. 4, 1997

[6] Adams , M. B., Deutsch, O. L., and Harrison, J.V., "A Hierarchical Planner for Intelligent Systems," *Proceedings of SPIE - The International Society for Optical Engineering*, Vol 548, Arlington, VA, April 9-11, 1985.

[7] Adams, M, et al., "Closed-Loop Operation of Large-Scale Enterprises: Application of a Decomposition Approach," in *Advances in Enterprise Control*, Minneapolis, MN, July 2000.

[8] Albus, J. S., Lumia, R., and McCain, H., "Hierarchical Control of Intelligent Machines Applied to Space Station Telerobots," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 24, No. 5, September 1988.

[9] Barth, C.D., "Composite Mission Variable Formulation for Real-Time Mission Planning," SM Thesis, Massachusetts Institute of Technology , 2001

[10] Cleary, M et al., "Hierarchical Decomposition Of Autonomy Requirements For Naval UCAVs," Draper report CSDL-2000-044, July 2000.

[11] Gamma, E., Helm, R., Johnson, R. and Vlissides, J., *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, Reading, MA, 1995.

[12] Hall, W., "Resource-Coordinated Hierarchical Planning for Real-Time Autonomous Systems," SM Thesis, Massachusetts Institute of Technology, 1992.

[13] Hall, W., "Efficient Capacity Allocation in a Collaborative Air Transportation System," PhD Thesis, Massachusetts Institute of Technology, 1999.

[14] Osborne, W. LTC, United States Army, et al. Information Operations: A New War-Fighting Capability, August 1996. (http://www.au.af.mil/au/2025/-volume3/chap02/vol3ch02.pdf).

[15] Ricard, M., "Mission Planning for an Autonomous Undersea Vehicle: Design and Results," *Proceedings of the Technology and the Mine Problem Symposium*, Monterey, CA, November 1996.

This page has been deliberately left blank

Page intentionnellement blanche

Intelligent Management Control for Unmanned Aircraft Navigation and Formation Keeping

M. Innocenti, F. Giulietti and L. Pollini
Department of Electrical Systems and Automation
University of Pisa, Via Diotisalvi 2, 56126 Pisa, Italy

Summary

The paper presents some aspects that have become critical in the context of guidance, navigation, and control of unmanned aerial vehicles. The envisioned cost-effectiveness of unmanned air vehicles in support of a variety of military and civilian applications has introduced basic and applied research challenges in areas such as levels of autonomy and "intelligence" of the platform, interaction with manned control centers, reliability and safety of operations, and control management of single vehicle, as well as formation flights. The paper addresses issues such as dynamic modeling, formation management and control, and guidance aspects. Their origin and potential solutions are presented with particular attention to flexible formation keeping.

Issues on UAV Guidance, Navigation, Control, and Formation Flight

As of today, unmanned aerial vehicles (UAV) are being proven as cost-effective platforms for supporting military operations. Indeed, as the operational roles of such vehicles expand beyond reconnaissance, intelligence, and data gathering, to electronic warfare, ground strike, and possibly air combat, many forecast that in the near future most combat aircraft will be unmanned. Remotely controlled, semi autonomous and autonomous vehicles will also play a key role in providing services for the civilian community. UAV's can provide efficient platforms for surveillance, search and rescue, relief in natural disaster areas, and law enforcement operations. However, as the operational capabilities of unmanned vehicles develop, there is a real need for an increase in their level of autonomy, decisional capabilities among different tasks, reliable control of multi-vehicle operations, and in fight and ground command and control center system integration. The successful accomplishment of these objective depends on the operational specifics of the onboard instrumentation, especially position sensors and communications links, but also on the capability of developing methodologies based on reliable and advanced basic research results. The present paper outlines some of these aspects, although not in an exhaustive manner, with particular reference to formation dynamic modeling, its stability and control, and automated guidance and navigation.

Increased performance in unmanned aerospace vehicles is leading toward mainly tailless new aerodynamic configurations for stealth capabilities, with perhaps high angles of attack for post stall aerodynamic regimes maneuverability, including thrust vectoring (agility, performance, and survivability following battle damage). In addiction, the use of multi-aircraft formations with remote control of a single or a few formation leader(s), with real time information exchange between the elements of the formation - supervised by some form of intelligent control - for higher levels of mission effectiveness is becoming increasingly appealing. The present effort is related to the latter aspect that is formation flying of unmanned air vehicles.

In the past, UAVs have been primarily used as test bed. They were used for instance to test flight envelope expansion, and flight control system for the manned aircraft or as a target for weapon systems. Over the past ten years, they have been envisioned as operational airborne platforms and have received considerable attention especially in many military strategic plans. From an engineering point of view, this was made possible and affordable in part by the miniaturization trend of flight control system components (sensors, actuators, CPUs, data acquisition systems, etc.), and by the increasing performance and speed of communication links.

The advantages of UAV technology appear several and clear. These vehicles, either flying in formation or individually, have high maneuverability potential, and in fact the only g-limitations are due to structural constraints because of the absence of a human pilot. In addition to eliminating the risk of human losses, UAVs have additional design advantages: a sophisticated cockpit is not required, no high redundant flight control systems, and lower weight than a similar size manned aircraft. This could in turn be used, partially or fully, to increase the payload and/or the range. Another less apparent advantage is of a logistic nature and it is due to the fact that the deployment of UAVs can take place in very remote locations, without the need of deploying military rescue personnel in case a pilot rescue mission become necessary.

Operational potential of UAVs could strongly be improved in some cases by making them flying within a close formation. The first advantage comes from aerodynamic effects. It is well known in fact that aircraft with large aspect ratio wings have better overall aerodynamic efficiency because of reduction in drag for a given lift. However, large aspect ratio implies large wingspan for a given area, this means that the resulting structure will be unreasonably flexible and fragile for lightweight design. A similar improvement in overall efficiency can be achieved by flying multiple aircraft in close formation. In an idealized case of *n* identical aircraft, each with the aspect ratio *AR* flying in tip-to-tip formation, the effect would be that of a single aircraft with *nxAR* aspect ratio. The aerodynamic benefits are due to favorable wake-vortex encounters. Wind tunnel tests and analytical studies have shown that the benefits increase as additional aircraft are added to the formation. Moreover, from an operational point of view, many aircraft involved in a mission can be better managed if they fly in a formation, rather than in an undefined structure.

An important challenge in the study of formation flight is represented by the complexity of the aerodynamic coupling. It is clear that the aerodynamic interference between different aircraft in the formation needs to be fully evaluated, modeled, and quantified since it may have critical effects on overall dynamic performances. For this purpose a primary factor is the size of the different aircraft within the formation; this, in turn, affects the aerodynamic interference induced by each one on the others of the formation, and, ultimately, dictates the relative distance allowed within the formation itself. Mathematical modeling of the aerodynamic interference between different aircraft in a formation was first approached by Bloy and others. They considered the problem of aerodynamic interference on lateral directional stability during air-to-air refueling maneuvers. In Reference [3], Myatt and Blake proposed an aerodynamic database with experimental data for the follower aircraft in a two-vehicle formation. In more recent studies, Gingras and co-workers proposed wind tunnel testing to acquire data for close vehicles simulation, while Nelson and Jumper used the Horseshoe Vortex theory to discuss the response of an airplane following an encounter with the trailing vortex wake by another. Blake, D'Azzo and Multhopp further discussed the problem of aerodynamic interference in a close formation flight, with a Leader/Wingman structure. The Leader generates vortices behind its wing. Such vortices exert actions on the Wingman lifting surfaces. In this approach, the Leader's wake is modeled via Horseshoe Vortex theory, and the drag reduction, sideslip force and induced angle of attack are introduced in the Wingman dynamics through additional stability derivatives leading to a mathematical aircraft model for a formation flight.

The other main challenge of close formation dynamics is the control system design. Close formation flight control, intended in its broad aspects of guidance, navigation and control, was originally studied for a classic leader/wingman configuration. An aircraft (Leader) is selected to direct the formation, following a prescribed path, and all the others (Wingmen) are expected to maintain a fixed relative distance with respect to the lead airplane, in order for the formation to have a desired geometrical shape. D'Azzo and co-workers analyzed the kinematical coupling effect of a two-aircraft configuration, and introduced a proportional integral (PI) controller for formation control. Wolfe, Chichka, and Speyer introduced the concept of decentralized control for UAVs in formation flight in reference 11. The advantages of using decentralized controllers are clear, especially for a large size UAVs formation, a full state feedback solution for the problem could be quite unfeasible due to the very high number of states. In addition, a decentralized control approach would provide more flexibility for time-varying number of UAVs in the formation. Later work by Chichka, Speyer, Chichka, Speyer, and others, in references 11 and 12 respectively, focused on a peak seeking control approach for close formation flight where the overall formation control was approached as an optimal control problem with the objective of minimizing the overall drag. A fairly intensive and inclusive study on formation flight control - with the use of a PI control scheme- was presented by Proud, Pachter, and D'Azzo for a 2-D formulation of the problem, and then recently extended to a 3-D formation dynamics problem. Schumacher and Kumar recently presented results on the use of adaptive control within the area of formation control. This adaptive scheme featured an optimal LQR design for the outer loop and a Dynamic Inversion design for the inner loop.

The paper is structured as follows: In the next section some classical approaches to formation dynamics and control are presented. Then, a procedure for aerodynamic interference is suggested, in order to improve the aircraft models. Given a lifting surface system, the proposed technique permits the calculation of induced velocities and then the forces due to coupling effects between aerodynamic surfaces. Simulation results with a three-aircraft formation have shown that each aircraft experiences an increase in lift and a drag reduction by flying in a close formation. This result does not appear if a single Horse-Shoe Vortex technique were used, because such method is based on the hypothesis that only the follower is affected by the wake. Thus, the above results are used to obtain the dynamic model of an aircraft flying within a formation. A different approach to formation flight is presented where each aircraft does not refer to the preceding one or to the formation leader, but keeps its position with respect to an imaginary point in the formation whose dynamics depend on all the others. The approach is based on the apparent behavior of some migratory birds, that during flight 'wait' for the those birds which have changed the original geometry of the formation by flying in a different path. The formation controller is constituted by two subsystems: a trajectory controller which provides tracking of a prescribed path, and a position controller which permits formation geometry keeping. These control laws are mixed by a parameter that depends on the position error. Finally, a waypoint-based guidance scheme is suggested as a potentially appropriate navigation loop.

Traditional Formation Control

The formation control problem can be traditionally approached, by designing first an inner loop, that allows tracking commanded velocity, altitude and heading. The outer loop consists of a Formation Controller (FC) that generates a reference path command for the inner loop, in order to follow a desired trajectory and to maintain the formation geometry.

The basic dynamic model for inner loop controller design, has a standard linearized form given by

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases}$$

with $x \in \Re^n, y \in \Re^p, u \in \Re^m$ as state, output and input vectors respectively. In particular, we refer to a sample trim condition corresponding to *Mach* = 0,4 and altitude of 10,000 *ft*. The dynamics described above include four inputs (symmetrical and differential taileron deflection, rudder deflection and engine throttle, and 14 states inclusive of 6 body rates, Euler angles, center of mass position and engine states.

Inner Loop Synthesis

Assuming dynamic separation between longitudinal and lateral-directional modes, the state and control vectors can be partitioned as follows

$$\begin{cases} \dot{x} = \begin{bmatrix} \dot{x}_{long} \\ \dot{x}_{lat} \end{bmatrix} = \begin{bmatrix} A_{long} & 0 \\ 0 & A_{lat} \end{bmatrix} x + \begin{bmatrix} B_{long} \\ B \end{bmatrix} \begin{bmatrix} u_{long} \\ u_{lat} \end{bmatrix} \\ y = \begin{bmatrix} y_{long} \\ y_{lat} \end{bmatrix} = Cx + Du \end{cases}$$

An LQ-Servo system was developed, shown in Figure 1 with the longitudinal inner-loop controller based on the above and capable of providing adequate tracking of commanded velocity and altitude.

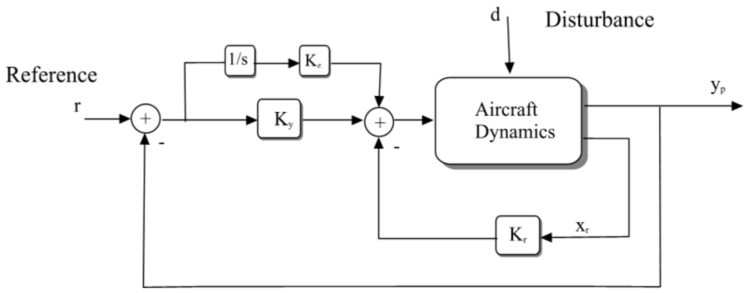


Figure 1. LQ-Servo Block Diagram

the full state feedback gain matrix K_{long} was obtained from the solution of the algebraic Riccati equation associated to the quadratic cost function in

$$\int_0^{\infty}\left(y_{long}^T Q y_{long} + u_{long}^T R u_{long}\right)dt$$

where

$$y_{long} = \begin{bmatrix} u & w & q & \theta & h \end{bmatrix}^T, u_{long} = \begin{bmatrix} \delta_e & \delta_{th} \end{bmatrix}^T. \text{ with}$$

$$\begin{cases} u_{long} = -K_{long} x_{long} & A_{ong}^T P_{long} + P_{long} A_{long} + C_{long}^T Q C_{long} \\ K_{long} = R^{-1} B_{long}^T P_{long} & -P_{long} B_{long}^T R^{-1} B_{long} P_{long} = 0 \end{cases}$$

The lateral-directional inner loop was synthesized in a fashion similar to the longitudinal one. The full state feedback controller is given by

$$u_{lat} = -K_{lat} x_{lat}$$

and the gain matrix K_{lat} is obtained with a minimization of a performance index such as the one above. Tracking is achieved as shown in Figure 2 below, and more detailed results can be found in [9].

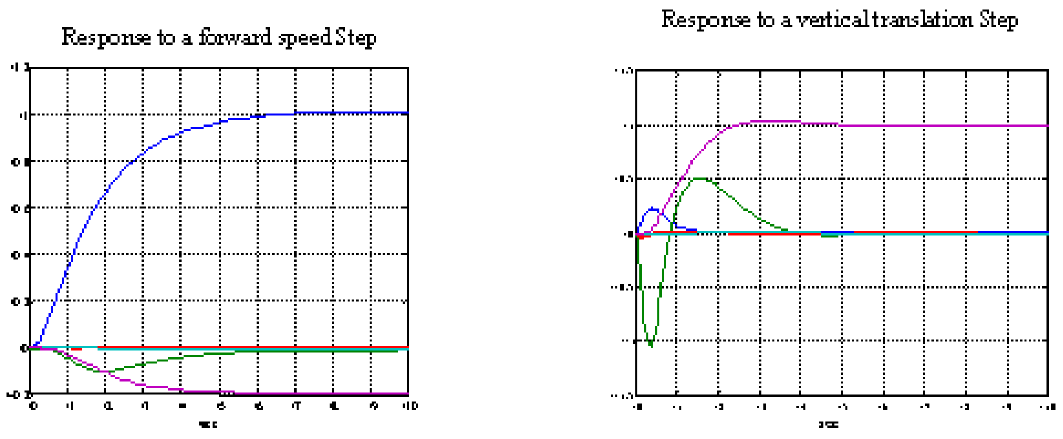


Figure 2. Inner Loop Controller Performance

Formation Control

In a typical formation flight, the Wingman follows the trajectory of the Leader, taking the other aircraft as reference to keep its own position inside the formation. In a large, rigidly flying formation, intra-aircraft distances must be kept constant. Then formation trajectory definition is usually the primary responsibility of the Leader. Many important aerodynamic effects influence the specifications of the overall control system, such as the vortex leaving the trailing edge of the wing for instance. In order to attain a more realistic simulation of a formation, a horseshoe vortex model of a wake was introduced. To simplify the calculation of the induced velocities, the tail creates no vortices, under the assumption that the main wing lifting effect is much larger. The vortex produces an up-wash on the wing of the following aircraft. The main up-wash effect is the increase of angle of attack, with respect to the isolated flight condition, thus more lift is generated. The vortex-induced velocity decreases with distance; hence the left side of the wing will be more affected than the right side. This causes a rolling moment, thus an attitude automatic control is necessary. Creation of a yawing moment has been neglected due to its weak contribution.

The primary objective of the formation controller (FC), also based on LQR techniques, is to maintain the formation geometry. To compute the distance to its reference, each aircraft acquires its position $P = (X, Y, H)$ from a GPS-based position sensing system, and receives, through appropriate communication channels, other aircraft positions $P_R = (X_R, Y_R, H_R)$. The formation controller is also responsible for having each aircraft follow a prescribed path. Each FC receives path information in terms of velocity, heading and altitude $T_R = (V_R, H_R, \psi_R)$, from another Wingman, the Leader or a ground station. Then the received data vector becomes $R_R = (X_R, Y_R, H_R, V_R, \psi_R)$. The commanded trajectory for the inner loop controller can be defined then as $T_C = T_R + \Delta T$, where ΔT is the output of the formation controller. The position error inside the formation is $\Delta P = P_R - P$, and it is composed of the error in the formation plane $\Delta P_{XY} = [\ X_R\text{-}X,\ Y_R\text{-}Y\ ]$ and error in the vertical plane $\Delta P_H = [\ H_R\text{-}H\ ]$: $\Delta P = [\ \Delta P_{XY},\ \Delta P_H\ ]$. These coordinates come from two GPS receivers, and are expressed in an earth fixed reference frame; they must be rotated into the aircraft body-fixed frame: $\Delta P' = [\Delta P'_{XY},\ \Delta P'_H\ ]$ using aircraft Euler angles $[\phi, \theta, \psi]$ in order to be used by the formation controller. In fact, only ΔP_{XY} must be rotated in order to keep the formation in the X-Y plane, otherwise $\Delta P'_H$ would depend on the (ϕ) or pitch (θ); then $\Delta P'_H = \Delta P_H$, and

$$[\Delta P'_{XY}, 0] = {}^{B}R_E\,(\phi, \theta, \psi\,) \cdot [\ \Delta P_{XY}\,,\ 0\ ]$$

where ${}^{B}R_E$ is the earth-to-body frame rotation matrix. The effective formation controller was designed with linear quadratic control techniques and takes $\Delta P'$ as input. Figure 3 shows the complete control block diagram. The resulting control law is:

$$\Delta T = K_{FC}(s)\Delta$$

The FC applies the corrections ΔT to the reference trajectory T_R to generate trajectory commands T_C for the aircraft autopilots. These corrections take into account the changes in velocity, heading and altitude, to the reference trajectory, that are necessary to maintain the formation geometry. Each aircraft can take more than one reference from the other elements of the formation. A convex combination ΔP_W of the distance errors ΔP_i, from the *i-th* aircraft taken as reference, can be used as a unique position error by the formation controller as:

$$\Delta P_W = \sum_{i=1}^{m} k_i \Delta P_i \qquad \sum_{i=1}^{m} k_i = 1 \qquad k_i > 0 \quad \forall i$$

Using ΔP_W, in place of ΔP, allows the formation to possess a certain degree of "elasticity"; that is, the formation stretches in the direction of keeping the average position error to a minimum.

To validate the overall control system performance, several computer simulations were performed with a three-aircraft diagonal formation shown in Figure 4. The relative nominal distances between each aircraft were set to be 15 and 10 meters along the x and y-axis respectively while the altitude was the same for all. With respect to the sequential references chain, two types of simulations were performed with two different strategies:

Leader Mode: Both Wingman1 and Wingman2 take the trajectory references from the Leader of the formation.

Front Mode: Each aircraft takes its reference from the preceding one. In this case, Wingman1 is referred to Leader and Wingman2 is referred to Wingman1.

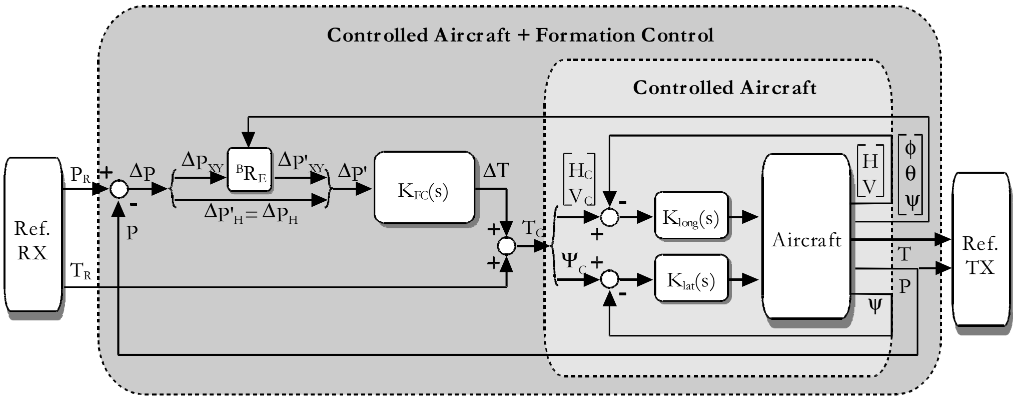


Figure 3. Complete control scheme.

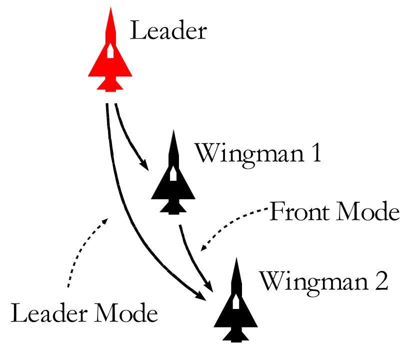


Figure 4. Three aircraft diagonal formation geometry.

Figures 5 and 6 show the responses of the Wingman2 to airspeed and altitude commands in Leader Mode and Front Mode. In the Front Mode structure, Wingman2 presents a poorer transient response, due to error propagation. Although no optimized synthesis was performed in the computation of the control gains, and each vehicle posts the same set of gains, on the basis of the preceding simulation results a formation controller based on Leader Mode suggests better transient responses because of the absence of error propagation along the references chain. On the other hand, this type of formation structure may be more critical, because in this case, Wingman2, which is directly connected to the Leader, has no information about its distance to Wingman1 therefore it would be not capable of avoiding a collision with Wingman1. Optimizing the autopilot, which is not the focus of the present work, can reduce error propagation.

Improved Strategy for Formation Control

In this section we present a more accurate dynamic modeling, and a modified formation controller structure. A traditional single Horseshoe Vortex technique consists in replacing the whole lifting surface by single vortex made with two free segments and one wing-fixed segment. Such model is simple and introduces minor modifications to the dynamics equations. However, by modeling a Leader/Wingman close formation by the single Horseshoe Vortex theory, only one aircraft is affected by the wake efforts, and the effect of the wake along the *x*-axis is not modeled to obtain a close analytical solution. This may cause the results not to be very accurate or complete.

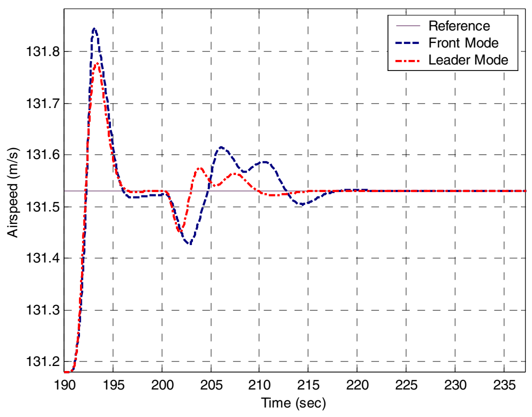


Figure 5. Wingman2 response to speed command.

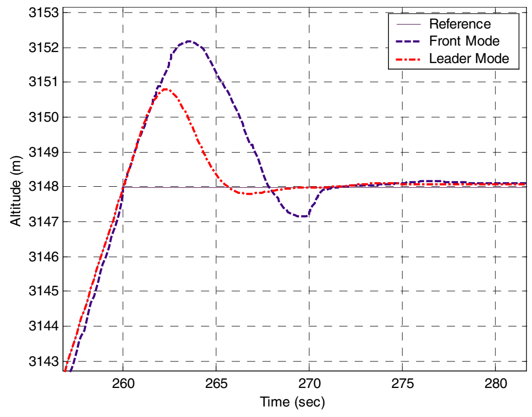


Figure 6. Wingman2 response to an altitude command.

In fact, in a formation flight problem, the mathematical modeling for all the aircraft involved in the formation is a critical issue. On the other hand, experimental data from wind tunnel tests could describe completely the aerodynamics interference effects but may become too costly in case of an increasing number of vehicles. Here, a different 3-D approach, based on a wing-distributed Horseshoe vortex system is presented. Given a system of lifting surfaces, this technique features the calculation of induced velocities as well as the forces and moments due to the coupling effects between all the aerodynamics surfaces. The first step consists in defining the lifting surfaces, only single tapered wings are considered here. To define each lifting surface, the following vector containing geometrical and aerodynamic parameters of the root profile is introduced:

$$R_p = \begin{bmatrix} \delta_h & \Lambda & \sigma & c_r & c_t & C_{L\alpha} & C_{m_0} & \alpha_\infty \end{bmatrix}$$

Once a number *s* of lifting surfaces has been defined, each surface is modeled through a straight line, also known as *lifting line*, passing through the aerodynamic center of the local profile. The aerodynamic center of the single profile is approximately set at 26% of the geometric chord. Let us now consider the *r- th* lifting surface. First the two halves of its lifting line are divided in *nr/*2 segments, then a vortex system, composed by a number *nr* of horseshoe vortices, is distributed on each lifting line. Each horseshoe vortex is modeled with a wing-fixed portion and two free portions that extend to infinity. Thus the two following subsets of points are introduced:

$$P_r = \left\{ P_{k_r} \in \mathbb{R}^3;\ k = 1 \ldots n_r + 1 \right\}$$
$$m_r = \left\{ m_{k_r} \in \mathbb{R}^3;\ k = 1 \ldots n_r \right\}$$

The first subset P_r contains the points where the free portions of each horse-shoe vortex leaves the lifting line. The second subset m_r contains the points where the induced velocity is evaluated. They are the middle points of the wing-fixed portions of each horse-shoe vortex. Figure 7 shows, as an example, the generic *r- th* lifting surface with the vortex system along with the points P_{kr} and m_{kr} .

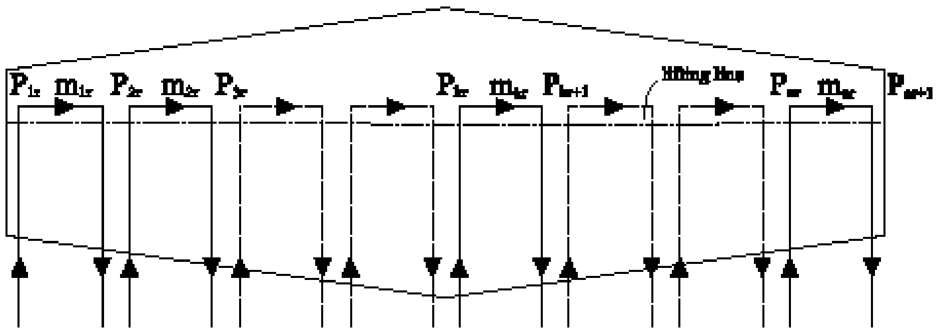


Figure 7. Lifting surface and vortex system

Once the lifting surfaces have been defined, in terms of lifting line and vortex systems, the induced velocity and then the strength of the vortices may be evaluated. The aerodynamic interference is calculated by locally using the fluid dynamic analogue of the Biot-Savart law from electromagnetism. Such law is applied both to the wing-fixed portion and to the free portions of each vortex yielding the computation of the vortex system strength. At this point, aerodynamic forces, moments, and coefficients can be calculated, which for a conventional air vehicle can be expressed as:

$$C_{F_1} = \sum_{t=1}^{3} C_{F_t}, \dots, C_{F_{n_a}} = \sum_{t=s-2}^{s} C_{F_t}$$

$$C_{M_1} = \sum_{t=1}^{3} C_{M_t}, \dots, C_{M_{n_a}} = \sum_{t=s-2}^{s} C_{M_t}$$

for a n_a aircraft formation. Aerodynamic calculations were performed for a close formation, consisting of three unmanned aircraft systems. The relative nominal distances between aircraft were chosen to be 4 and 5 m along the *x and y* respectively, with the altitude held constant. The geometries of the close formation are shown in Figure 8. Aerodynamic and geometry data used for this calculations are listed in Table 1, while the evaluated force and moments coefficients, for the formation configurations based on the previous data are provided in Table 2, compared to the values for 'isolated' flight. From the results it can be seen that the rear aircraft gain an increase in lift and an induced drag reduction by flying in formation. It is also important to notice that the front aircraft has an increase in lift coefficient and a drag reduction compared to solo flying. This last result is not found by using simple horseshoe wake models.

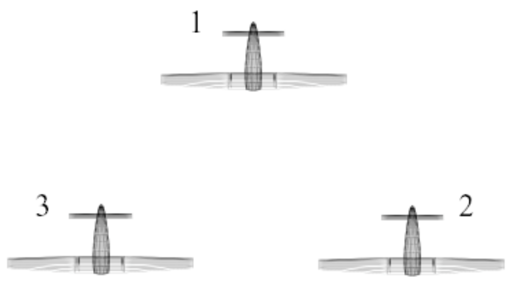


Figure 8. Formation geometry for Wake contribution evaluation

Parameter	Value
Wing area, S, m^2	0.697
Wing span, b, m	3.0
Tail span, b_t, m	1.1
Vertical tail height, h_v, m	0.25
Wing root cord, c_r	0.28
Wing tip cord, c_t	0.16
Tail root cord, c_r	0.12
Tail tip cord, c_t	0.06
Vertical tail root cord, c_r	0.22
Vertical tail tip cord, c_t	0.12
Root lift curve slope, $C_{L_\alpha}, 1/rad$	5.878
Root tail curve slope, $a_t, 1/rad$	5.878
Root vertical tail curve slope, $a_{v_t}, 1/rad$	5.878

Table 1: Aerodynamic data

Aerodynamic	Isolated	Formation Flight		
Coefficients	flight	1	2	3
C_{x_t}	0.022346	0.022796	0.023635	0.023635
C_y	0	0	-0.000066	-0.000066
C_z	-0.672993	-0.676335	-0.682787	-0.682787
C_l	0	0	-0.000781	0.000781
C_m	0.133341	0.133068	0.132804	0.132804
C_n	0	0	-0.000109	0.000109

Table 2: Close formation evaluated coefficients (body axis)

Formation Dynamics

In this section, the mathematical model of an aircraft flying within a formation is outlined. The main difference with respect the model of an 'isolated' aircraft consists in the value of the aerodynamic coefficients. The modeling procedures outlined before are introduced. Once the mathematical model is developed, the kinematic equations describing the relative distances between the aircraft are presented. These equations are the basis of formation control system design. The aerodynamic force and moment coefficients can then be directly introduced in the flight dynamic equations. However, is useful to split the aerodynamic coefficients according two different contributions:

$$\begin{pmatrix} C_F \\ \\ C_M \end{pmatrix} = \begin{pmatrix} C_{Fi} \\ \\ C_{Mi} \end{pmatrix} + \begin{pmatrix} C_{Ff} \\ \\ C_{Mf} \end{pmatrix}$$

The first term on the right hand side of the previous equation is the coefficient in case of 'isolated' flight, while the second is related to the contributions associated with formation flying. This can be useful within the design of the control laws, since the wake effects can be treated as disturbances to be rejected by the control system.

The basic model used here is a 3DOF dynamic system shown below, with standard notation:

$$\begin{aligned}\dot{V} &= \frac{(T-D)}{m} - g\sin\gamma \\ \dot{\gamma} &= \frac{g}{V}\left(n\cos\phi - \cos\gamma\right) \\ \dot{\chi} &= \frac{g\,n\sin\phi}{V\,cos\gamma}\end{aligned}$$

The state variables describing the aircraft dynamics are the airspeed V , the flight path angle $\gamma$and the heading angle χ, while T, n and ϕ, are thrust, load factor and bank angle and they are the input variables; D is the aerodynamic drag and m the aircraft mass. According to the aerodynamic coefficients separation, the system can be rewritten introducing changes in lift, induced drag and side force. This leads to:

$$\dot{V} = \frac{T-D}{m} - g\sin\gamma - \frac{\Delta D_i}{m} \quad \dot{\gamma} = \frac{g}{V}\left[(n + \frac{\Delta L}{mg})\cos\phi - \cos\gamma\right] \quad \dot{\chi} = \frac{g(n+\frac{\Delta L}{mg})\sin\phi}{V\cos\gamma} + \frac{\Delta Y}{mV}$$

or, in compact form

$$\dot{\boldsymbol{X}}_f = \dot{\boldsymbol{X}}_i + \frac{1}{2}\rho V^2 S\left(\boldsymbol{\Delta}_f \cdot \boldsymbol{C}_{Ff}\right)$$

$$\boldsymbol{\Delta}_f = \begin{pmatrix} 1/m & 0 & 0 \\ 0 & \cos\phi/mV & 0 \\ 0 & \sin\phi/(mV\cos\gamma) & 1/mV \end{pmatrix}$$

To maintain the formation geometry, each aircraft must keep its prescribed distance from a reference. Such reference may be the Leader of the formation or a neighborhood aircraft or an imaginary point within the formation. To calculate the relative distance of the *i-th* aircraft from its reference *r*, three reference frames are introduced: an inertial, Earth-Fixed frame F_O and two kinematic frames F_{ki} and F_{kr}, with the origin on aircraft *i ¡ th* and on reference *r* respectively. The relationship between position derivatives in the variuos reference frames is given by

$$\dot{\boldsymbol{P}}_r^O = \dot{\boldsymbol{P}}_i^O + \boldsymbol{V}_{i,r}^O + \boldsymbol{\Omega}_i^O \times \boldsymbol{R}_{i,r}^O$$

$$\boldsymbol{V}_r^i = \boldsymbol{V}_i^i + \boldsymbol{V}_{i,r}^i + \boldsymbol{\Omega}_i^i \times \boldsymbol{R}_{i,r}^i$$

with

$$\boldsymbol{V}_r^i = \begin{bmatrix} V_r\cos\gamma_e\cos\gamma_e \\ V_r\cos\gamma_e\sin\chi_e \\ -V_r\sin\gamma_e \end{bmatrix} \qquad \begin{bmatrix} \dot{dx} \\ \dot{dy} \\ \dot{dz} \end{bmatrix} = \begin{bmatrix} V_r\cos\gamma_e\cos\gamma_e \\ V_r\cos\gamma_e\sin\chi_e \\ -V_r\sin\gamma_e \end{bmatrix} - \begin{bmatrix} V_i \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \dot{\gamma}_i dy - \dot{\chi}_i dz \\ -\dot{\chi}_i dx \\ \dot{\gamma}_i dx \end{bmatrix}$$

$$\gamma_e = \gamma_r - \gamma_i \text{ and } \chi_e = \chi_r - \chi_i.$$

The previous method for relative distances calculation is typically used when, during the mission, aircraft within the formation exchange each others trajectory information, particularly airspeed, flight path angle, and heading angle. If the trajectory information are not available, and the if aircraft could know only the absolute position (e.g. by GPS), the relative distances between aircraft could be computed first referred to the inertial frame and then rotated to the kinematic frame of each aircraft.

Formation Strategy

The Leader/Wingman structure may not be the best strategy to perform complicated maneuvers, especially in multi-aircraft formation flight. In fact, each aircraft in the formation must cooperate in order to maximize the possibilities for a formation to obtain and retain its structure. The aircraft in the formation must work together to achieve a common goal; then, to overcome the limitations of Leader/Wingman structure, a different strategy is proposed, based on the behavior of migrational birds. The aircraft in a formation are not longer referring to each other, but they are required to keep a specified distance from an imaginary point called Formation Geometry Center (FGC). The FGC position depends on the relative distances between the aircraft in the formation itself; this allows each aircraft to have the capability of sensing other vehicles movement from the nominal position in the formation. In the presence of disturbances, for instance, if one of the aircraft loses its position, the other senses the change, and depart momentarily from the prescribed trajectory, manoeuvreing all-together in order to reconstitute the formation geometry. Once the geometry has been reached again, all aircraft continue to follow the prescribed trajectory. This approach is typical of several species of migrational birds.

For long distance migrational flight, birds of several species tend to move in a formation, flying close to each other maintaining a defined geometrical shape. One of the most interesting considerations is the fact that during a migrational flight if one or more elements of the group loses its position in the formation, the others leave the migration trajectory and ‘wait’ for the lost ones until the formation shape is reconstituted. There are two main reasons why birds fly in a formation; the first reason is related to aerodynamic considerations while the other is due to the ‘social behavior’ of some species of birds. In terms of aerodynamics effects, birds take advantage of the induced velocity produced by the wing tip vortex phenomenon: the inner wing of each bird in a V formation, for example, gains an increase in lift and then a reduction in induced drag from the upward rising side of the vortex left by the outer wing of the bird ahead. For this reason, it is important that each bird in the group keeps its position; losing position by one or more birds in a migrational flight means to lose a non-negligible portion of aerodynamics efficiency. On the other hand, birds like geese and swans form ‘family’ groups: offspring remain with parents one or two seasons after the birth and fly together with them during the migration. The elements in these formations know each other and try to remain together independently from aerodynamics advantages while other birds like storks live in non-familiar group, but fly together with others just in order to improve the efficiency of the migration (especially during food searching or meeting season). From these motivations it’s possible to assume that the preceding consideration to be reasonable. A two-aircraft formation is considered in the present work. Each aircraft in the formation will be represented using the modified three-degrees of freedom point-mass model described in the previous section.

In this approach, aircraft do not refer to each other and, in order to maintain formation geometry an imaginary point called Formation Geometry Center (FGC) is introduced and each aircraft must keep a prescribed distance from this point. The FGC dynamics for an *N*-aircraft formation can be represented by the following differential equations:

$$\begin{aligned}
\dot{x}_{FGC} &= \sum_{i=1}^{n_a} \frac{V_i \cos\gamma_i \cos\chi_i}{n_a} \\
\dot{y}_{FGC} &= \sum_{i=1}^{n_a} \frac{V_i \cos\gamma_i \sin\chi_i}{n_a} \\
\dot{z}_{FGC} &= \sum_{i=1}^{n_a} \frac{-V_i \sin\gamma_i}{n_a}
\end{aligned}$$

By integrating the preceding equations, the position of the FGC in the Earth-fixed frame (P_{FGC}) is found and the distance between the FGC and the *i-th* aircraft, referred to the frame F_O, is defined as:

$$\boldsymbol{d}_i = \boldsymbol{P}_{FGC} - \boldsymbol{P}_i$$

Prior to the design of any formation controller, each aircraft of the formation must feature adequate tracking of a commanded trajectory through an inner loop controller. The innerloop controller is based on the model for an individual aircraft in undisturbed air described in the previous section. The following three lag filters are introduced to model the fact that the aircraft cannot change bank angle (*ϕ*), load factor (*n*) or thrust (*T*) instantaneously.

$$\dot{T} = (T_c - T)/\tau_t$$

$$\dot{\phi} = (\phi_c - \phi)/\tau_b$$

$$\dot{n} = (n_c - n)/\tau_n$$

To implement the inner-loop controller, a dynamic inversion law is used. The desired trajectory vector is written in terms of a commanded airspeed (*Vc*), flight path angle (*γ*), and heading angle (*χ*). It is assumed that the desired trajectory will be governed by the following linear time-invariant equations:

$$\dot{\gamma} = \omega_\gamma(\gamma_c - \gamma)$$

$$\dot{V} = \omega_V(V_c - V)$$

$$\dot{\chi} = \omega_\chi(\chi_c - \chi)$$

The bandwidth dictates how quickly the actual aircraft trajectory changes with respect to the new desired value. To derivate the dynamic inversion control law, the commanded rates are set equal to the associated actual, and then solving for thrust, bank angle, and load factor. The computed thrust, bank angle, and load factor become the commanded values needed and are given by:

$$T_c = D + \omega_v(V_c - V)m + mg\sin\gamma$$

$$n_c \cos\phi_c = \frac{V}{g}\left[k_\gamma\left(\gamma_c - \gamma\right) + \cos\gamma\right] = c_1$$

$$n_c \sin\phi_c = \frac{V}{g}k_\chi\left(\chi_c - \chi\right)\cos\gamma = c_2$$

from which

$$n_c = \sqrt{{c_1}^2 + {c_2}^2}$$

$$\phi_c = tan^{-1}\left(\frac{c_2}{c_1}\right)$$

Once the autopilot structure is defined as above, the model for a *N*-aircraft formation is obtained by introducing the force coefficients variations and the relative distances dynamics:

$$\begin{aligned}
\dot{\boldsymbol{x}}_1 &= f_1(\boldsymbol{x}_1, \boldsymbol{u}_1) + \tfrac{1}{2}\rho V_1^2 S_1 \left(\boldsymbol{\Delta}_{f_1} \cdot \boldsymbol{C}_{F_1 f}\right) \\
&\dots \\
\dot{\boldsymbol{x}}_n &= f_2(\boldsymbol{x}_n, \boldsymbol{u}_n) + \tfrac{1}{2}\rho V_n^2 S_n \left(\boldsymbol{\Delta}_{f_n} \cdot \boldsymbol{C}_{F_n f}\right) \\
\dot{\boldsymbol{d}}_1 &= g_1(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n) \\
&\dots \\
\dot{\boldsymbol{d}}_n &= g_n(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)
\end{aligned}$$

A Formation Controller (*FC*) is needed to reproduce the natural behavior of migrational birds. For the purpose of *FC* design, linearization around a selected flight condition of the above nonlinear model yields the linearized set of equation described by the following standard state space model:

$$\begin{cases} \dot{x} &= Ax + Bu \\ y &= Cx \end{cases}$$

Each aircraft has its *FC* constituted by two control systems: trajectory controller (*KT*) and position controller (*KP*). The main objective of the trajectory controller is to provide tracking of a prescribed path in terms of desired velocity, flight path and hading angle, while the position controller is designed to maintain inter-aircraft distances in order to give the formation a desired geometric shape. Trajectory and position controllers receive path error information, that is the difference between current and desired path, and FGC-distance error, that is the difference between current and desired distance from FGC, respectively:

$$\begin{aligned} e_T &= r_T - y_T \\ e_P &= r_P - y_P \end{aligned}$$

and they generate trajectory and position commanded vectors defined as:

$$\begin{aligned} u_T &= K_T e_T \\ u_P &= K_P e_P \end{aligned}$$

An LQ-Servo controller was developed, and gain matrices K_T and K_P were obtained through the minimization of the quadratic cost indexes:

$$J_{(\cdot)} = \int_{t_0}^{\infty} \big[u' \, R_{(\cdot)} \, u + e'_{(\cdot)} \, Q_{e(\cdot)} \, e_{(\cdot)} + \\ + x'_{r(\cdot)} Q_{r(\cdot)} x_{r(\cdot)}\big] \, dt$$

where Q_e regards the minimization of the desired output vector, while Q_r is related to the residual state vector. The resulting commanded vector *uc* is a convex combination of the two previous control
laws:

$$u_c = (1 - f)u_T + f u_P$$

where $f \in [0; 1]$, is a function of the position error e_P. The prescribed path can be commanded by one of the aircraft of the formation or by a ground station o by a Virtual Leader. In the presence of disturbances, there may be one or more aircraft with a position error with respect to the nominal configuration. In this case, the command vector u_C depends both on position error e_P and trajectory error e_T and so the aircraft in the formation may not be able to follow the prescribed path exactly until the geometry is re-established. When the formation has the desired shape, the position error e_P is zero for each aircraft and f=1. This causes the commanded vector be given only by the contribution of the trajectory controller and all the aircraft will follow the desired trajectory r_T.

Simulation Results

To validate the overall control strategy, simulations were performed for close formations consisting of two aircraft. The aircraft model includes aircraft/autopilot, the distributed HorseShoe Vortex code and *FGC* relative distance dynamics.

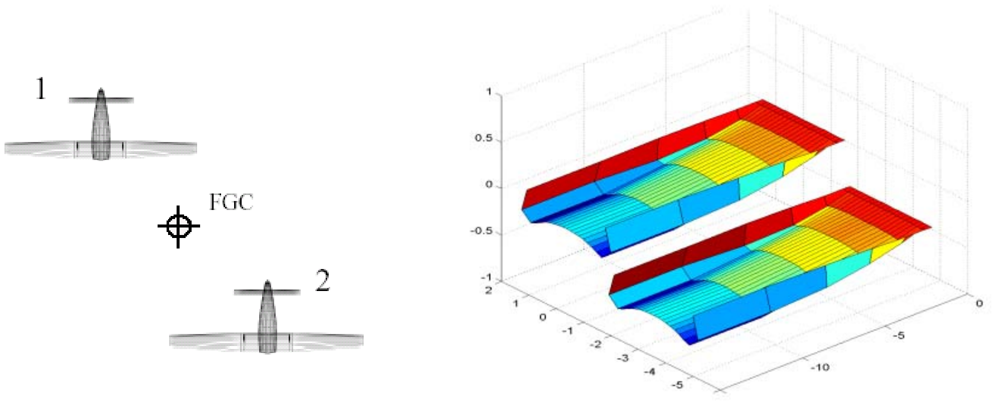


Figura 9. Two-aircraft configuration for simulation

The whole Formation Control scheme for such formation is shown in Figure 10 and is simulated in a MATLAB environment. The relative nominal distances between aircraft and *FGC* are selected to be 5 m along the *x* and *y* axes. The altitude is the same for both aircraft at 300m.

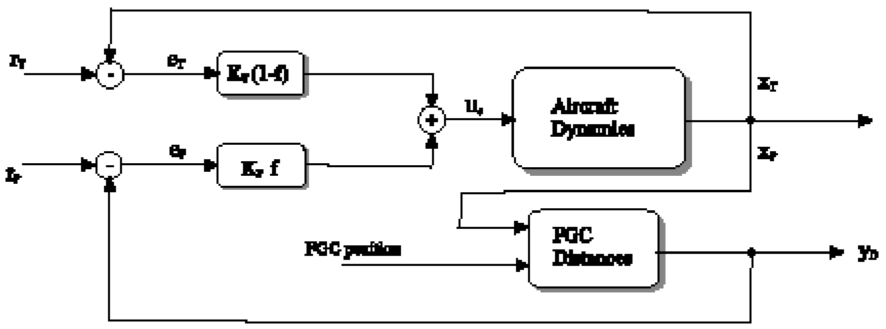


Figure 10. Complete formation controller for a Two-aircraft configuration

Additional aircraft data required for this effort are listed in Table 3 and the evaluated aerodynamic coefficient variations for this formation structure are listed in Table 4.

Parameter	Value
Velocity time constant, ω_V, s	0.3
Flight path time constant, ω_γ, s	0.2
Heading time constant, ω_χ, s	0.2
Thrust time constant, τ_t, s	0.9
Load factor time constant, τ_n, s	0.5
Bank angle time constant, τ_b, s	0.5
Gross mass, m, kg	10
Velocity, $V_\infty, m/sec^2$	20
Air density, $\rho, kg/m^3$	1.219

Table 3: Aircraft value for control design

Aerodynamic	Aircraft	
Coefficients	1	2
$\boldsymbol{\Delta} C_{D_t}$	-0.0015	-0.0020
$\boldsymbol{\Delta} C_Y$	-0.000004	0.000078
$\boldsymbol{\Delta} C_L$	0.0214	0.0297

Table 4: Evaluated force coefficient variations

The first simulation a path following task. Each aircraft receive the commanded trajectory from the VL. Starting from a steady-level flight condition at 20 m/sec, each aircraft is commanded to perform a heading change of -30 deg. Trajectory tracking is achieved and the relative distances between the aircraft are maintained at nominal values. Figure 11 shows aircraft response to the heading command, while Figure 12 shows the relative distance between aircraft and the *FGC* during the same manuever.

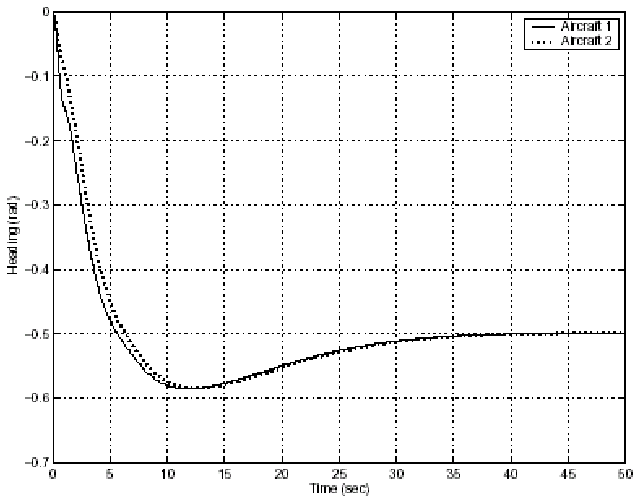


Figure 11. Aircraft responses to a 30 deg. Heading change

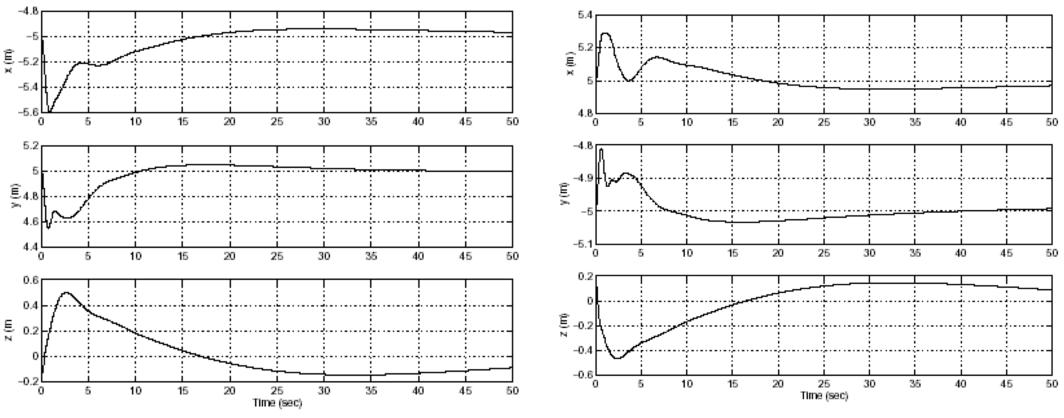


Figure 12. Relative distance of the two aircraft from *FGC*.

The second simulation is related to a formation geometry variation. Each aircraft is commanded to increase its relative distances from *FGC* from 5 to 7 m along *x* and *y*-axis. Figure 13 shows the relative distance between aircraft and the *FGC* while Figure 14 shows the airspeed time history during such maneuver.

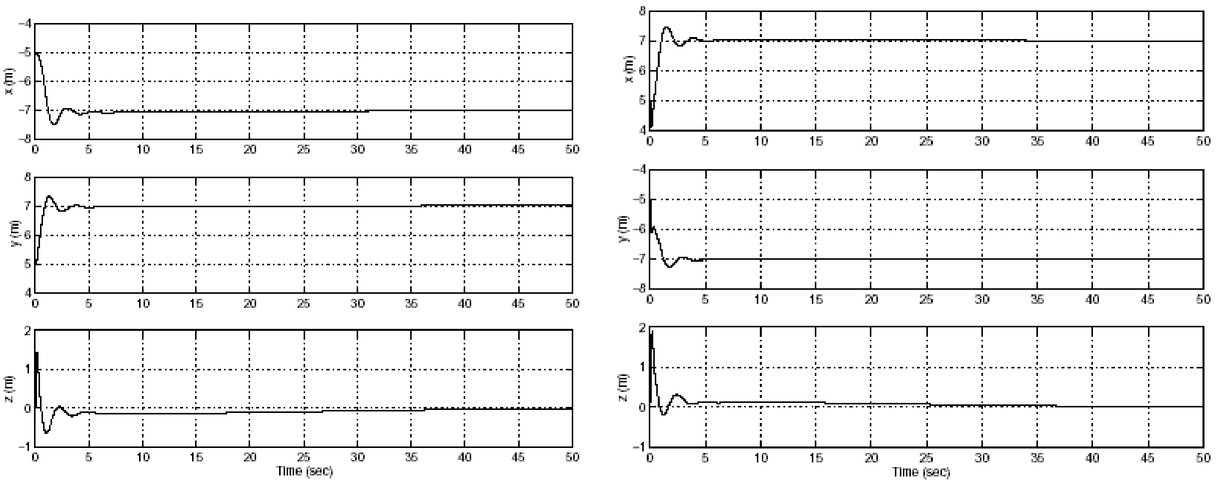


Figure 13. Relative distance of the two aircraft from *FGC*.

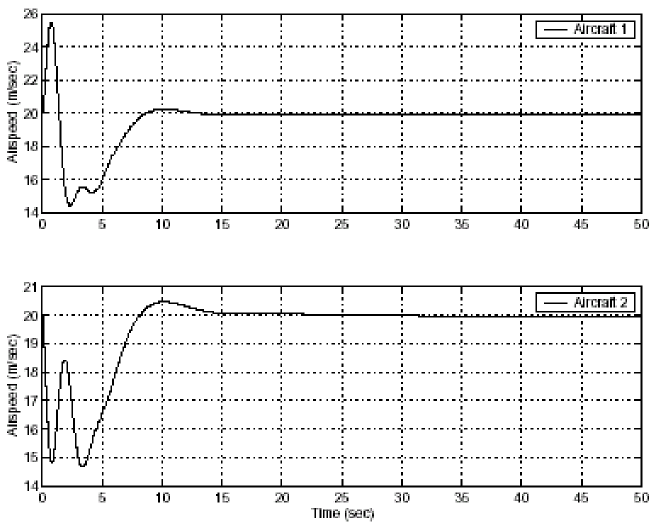


Figure 14. Airspeed time histories during formation geometry change.

The last simulation shows how the FC could reproduce the behavior of migrational birds. The initial condition is a steady-level flight. Each aircraft has an initial airspeed of 20 ft/sec that is the commanded trajectory. A speed perturbation is introduced on aircraft 2, which slows down with respect to aircraft 1. Figure 15 shows speed profiles for the two aircraft. It is clear how aircraft 1 stops following the prescribed velocity and 'waits' for aircraft 2 in order to maintain the nominal distances.

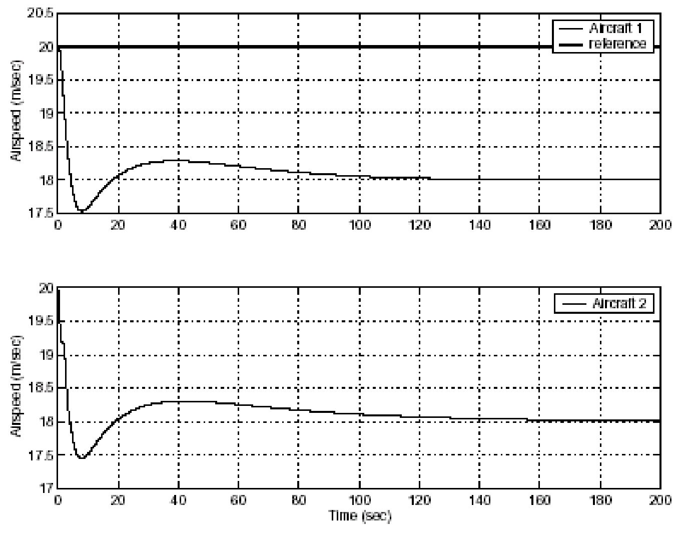


Figure 15. Airspeed time histories during perturbed flight.

Expert Guidance and Navigation Issues

In this section, a brief description of a waypoint apporach to guidance of unmanned air vehicles is presented. The work is relative to a single UAV, however current research is under way in order to extend the methodology to multiple cooperative aircraft.

A typical operation for an UAV in surveillance and search & rescue tasks can be with with a set of sequential waypoints to be passed by the aircraft. Since traditional proportional guidance techniques do not allow specifying desired waypoint's crossing direction, a fuzzy guidance scheme was developed. This allows a better trajectory description by specifying the single waypoint position in space, crossing heading and velocity. The procedure is based on a fuzzy controller that commands the aircraft, via its autopilot, to approach a list of waypoints. Implementation of a fuzzy guidance controller may be very expensive in terms of computational power/time, to solve this problem, a fuzzy controller was designed using a new approach based on Takagi-Sugeno fuzzy systems. This technique solves also previous 3 FGC version problems relative to non-smooth autopilot input generation.

The aircraft guidance problem is addressed, based on by an inner nonlinear control loop described previously in the paper, which allows tracking of commanded velocity , flight path, and heading angles. Then, an outer loop, that is the actual fuzzy guidance system (FGS), generates a reference path command in term of desired velocity, flight path and heading for the inner loop, in order to reach the desired waypoint. The Guidance System presented here is the evolution of the one presented by the authors in previous work. The main disadvantages of the old guidance system were the use of Mamdami Fuzzy networks that are too complex for real-time implementation, the use of triangular membership functions that resulted in too "square" trajectories and furthermore the fuzzy laws gave a good performance around the design aircraft speed only. One thing that remains unaltered from previous work is the use waypoint-relative coordinates to design guidance laws independently of waypoint orientation. To overcome the disadvantages of old approach, this new work proposes the use of Takagi-Sugeno Fuzzy networks with gaussian membership functions that helped generating smooth trajectories and resulted in less fuzzy sets and laws. Furthermore a non-linear scaling factor has been introduced to guarantee laws validity over a large speed range.

Two components constitute the guidance system: the Waypoint Generator (WG) and the Fuzzy Guidance System itself(FGS). The desired trajectory is specified in terms of a list of waypoints without any requirement on the path between two successive waypoints. A waypoint is given in cartesian-space coordinates ($Xw; Yw;Hw$), and a desired crossing speed (Vw) and heading angle (χw) are used to obtain a preferred approaching direction and velocity, thus the waypoint belongs a five-dimensional space $W = [Xw; Yw;Hw; Vw; \chi w]$. The WG holds a listof waypoints (WL) in 5-D, checks aircraft position, and updates the desired waypoint when the previous one has been reached within a given tolerance. When all waypoints have been reached, it holds the last one so that the aircraft loops around it. The waypoint generator's only task is to present the actual waypoint to the FGS. No dead-reckoning or navigational errors are modeled so the WG and the FGS know the exact aircraft positions, velocity and heading. Between theWG and the FGS, a coordinate rotation system transforms earth-fixed-frame position errors into waypoint-frame relative errors. Each waypoint defines a coordinate frame centered in the waypoint position (*Xw; Yw;Hw*) and rotated by χw around the *H*-axis. This coordinate transformation allows the synthesis of a fuzzy rule-set valid in the waypoint-fixed coordinated frame that is invariant with respect to the desired approach direction . When a waypoint is reached, the next one is selected, the actual reference value W is changed and the rotation matrix is updated to transform position and orientation errors into the new waypoint coordinate frame.

As described above, the aircraft autopilots are designed to track desired airspeed, heading and flight path angles. Using the completed decoupled implementation of guidance laws, three independent Takagi-Sugeno Fuzzy Controller were designed to constitute the FGS.

One FC generates the desired flight path angle for the autopilot using altitude error

$$e_H = (H_w - H);$$

$$\gamma_d = f_\gamma(e_H)$$

The second fuzzy controller computes desired aircraft velocity:

$$V_d = V_w + f_V(V_w - V) = V_w + f_V(e_V)$$

The third, and most complex FC is demanded to generate the desired heading angle using the position errors along the *X* and *Y* axes of the actual waypoint-frame and heading error. A fuzzy rule-set designed at a fixed airspeed value can produce a lack of tracking performance when the desired waypoint crossing-speed *Vw* differs significantly from the design value. The solution to this problem is achieved by introducing a speed-depending scale coefficient for position errors leading to.

$$\chi_d = \chi_w + f_\chi(e^w_{X_C}, e^w_{Y_C}, e_\chi)$$

with

$$\begin{pmatrix} e^w_X \\ e^w_Y \end{pmatrix} = Rot(\chi_w) \cdot \begin{pmatrix} e_X \\ e_Y \end{pmatrix} = Rot(\chi_w) \cdot \begin{pmatrix} X_w - X \\ Y_w - Y \end{pmatrix}$$

$$\begin{pmatrix} e^w_{X_C} \\ e^w_{Y_C} \end{pmatrix} = S(V_w, V^*) \cdot \begin{pmatrix} e^w_X \\ e^w_Y \end{pmatrix}$$

$$S(V_w, V^*) = \frac{V^*}{V_w}$$

where *V** is the fuzzy design point airspeed. A schematic block diagram of the control structure is swhown in figure 16.

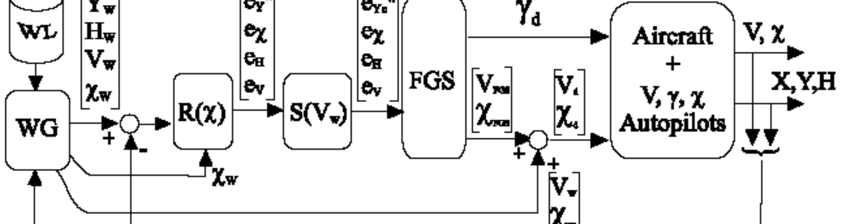

WL
WG
X_w Y_w H_w V_w χ_w
+
-
R(χ)
χ_w
e_X^w e_Y^w e_χ e_H e_V
S(V_w)
$e_{X_C}^w$ $e_{Y_C}^w$ e_χ e_H e_V
FGS
γ_d
V_{FGS} χ_{FGS}
+
+
V_w χ_w
V_d χ_d
Aircraft + V, γ, χ Autopilots
V, χ
X,Y,H

Figure 16. Implementation of Fuzzy guidance to a single UAV.

The fuzzy rules have been defined according to the desired approaching behavior and angular rates limitations of the aircraft. Fuzzy knowledge base was designed to generate flyable trajectories using the maximum linear and angular velocities and accelerations that are typical of a small propeller-engine aircraft.

The FGS provides with different desired flight path and heading angle commands for different values of distance from the waypoint. Based on the complete uncoupled aircraft model, it is possible to describe each fuzzy controller separately from the others. The Altitude and the Velocity FC systems are less complex than the Heading controller, and they are both implemented using Takagi-Sugeno model. For the first one the only input is the altitude error and four fuzzy set are designed to map this input and four for the desired output. The Velocity FC has similar complexity: 3 input fuzzy sets for the velocity error and 3 for the resulting desired output. As stated before, guidance in the horizontal (*X-Y*) plane is more complex than guidance in the vertical (*X -H*) plane. The horizontal plane fuzzy controller takes its inputs from scaled position errors and heading error. The scaled position error in the x-direction is coded into five gaussian fuzzy sets, while three sets are also defined for the scaled error in the y-direction. Considering the Takagi-Sugeno output function for this fuzzy controller:

$$y = \frac{\sum_{i=1}^{m}\mu_i(x)u_i}{\sum_{k=1}^{m}\mu_k(x)} = \frac{1}{c(x)}\sum_{i=1}^{S}\sum_{j=1}^{K}\mu_i^{xy}(e^w_{X_C}, e^w_{Y_C})\cdot\mu^X_{ij}(e_X)u_{ij} = \frac{1}{c(x)}\sum_{i=1}^{S}\mu_i^{xy}(e^w_{X_C}, e^w_{Y_C})\cdot\delta^X_{ij}(e_X)$$

S is the number of zone dividing the flight space and *K* is the number of subsets (dependent on the x-direction error) defined for each zone. The above equation can be simplified, yielding

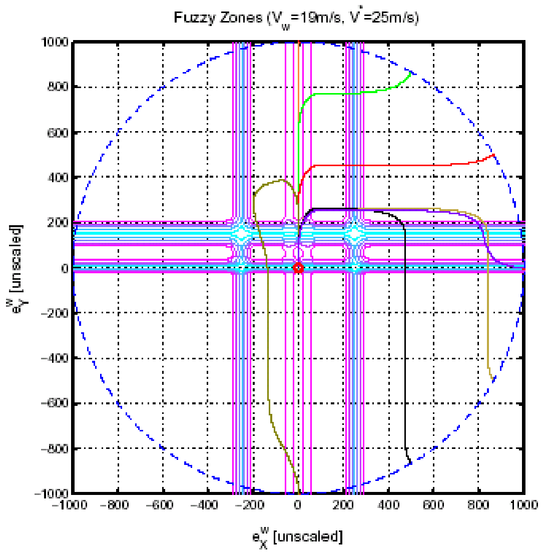
$$\sum_{i=1}^{S}\frac{\mu_i^{xy}(e^w_{X_C}, e^w_{Y_C})}{c(x)}\cdot\delta^X_{ij}(e_X) = \sum_{i=1}^{S}\overline{\mu}_i^{xy}(e^w_{X_C}, e^w_{Y_C})\cdot\delta^X_{ij}(e_X)$$

Fixing the scaled errors in the middle of *P*th zone, under the assumption that the contribution from the other zones is near to zero:

$$y\Big|_{\substack{e^w_{X^P_C}\\ e^w_{Y^P_C}}} = \overline{\mu}_P^{xy}(e^w_{X^P_C}, e^w_{Y^P_C})\cdot\delta^X_P(e_X) + \sum_{\substack{i=1\\ i\neq P}}^{S}\overline{\mu}_i^{xy}(e^w_{X_C}, e^w_{Y_C})\cdot\delta^X_{ij}(e_X) \simeq \overline{\mu}_P^{xy}(e^w_{X^P_C}, e^w_{Y^P_C})\cdot\delta^X_P(e_X)$$

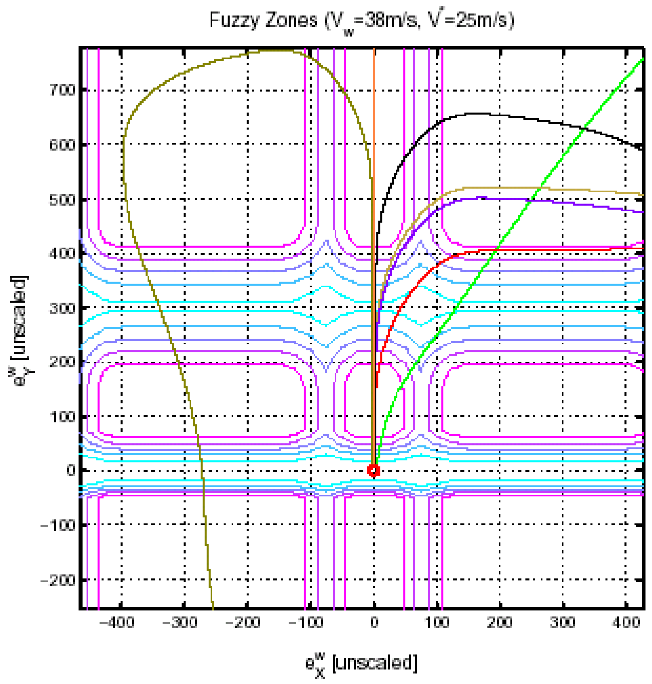
The above equation shows that the definition of fuzzy sets for the x-direction error error should be computed looking at each single set partitioning the flight space, and then looking at cumulative result. Under this assumption seven fuzzy sets have been defined.

Figures 17 and 18 show the effect of parameter S previously defined; the trajectories maintain a smooth shape but the turn radius changes: in particular the turn radius increases at higher speed and decreases at speed lower than *V**.



$u_i^{xy}(e^w_{X^P_C}, e^w_{Y^P_C})$

Figure 17. Contour plots for membership functions



$u_i^{xy}(e^w_{X_C^P}, e^w_{Y_C^P})$

Figure 18. Contour plots for membership functions

Simulation Results

The following simulation shows the results of a non planar trajectory. First the aircraft is driven to waypoint $W1$, then to align with $W2$, then to $W3$ that is 150 meters lower in altitude and very near on the (X,Y) plane and finally to $W4$ that is at altitude 100 with a desired approach angle rotated by 90 degrees with respect to the previous waypoint. Figure 19 shows this trajectory. The required descent from W2 to W3 is too steep for the aircraft capabilities, as decided in the design phase of fuzzy rule-set. A s a matter of fact, when the aircraft reaches the X,Y coordinates of W3 its altitude is still too high, and it starts a turn to come back to the waypoint at the prescribed altitude. In fact, the aircraft begins a spiral or a 8-shaped descent, centered on the waypoint vertical axis, decreasing altitude with the descent rate limitation given by FGS, until the waypoint altitude is reached and then it proceeds to next waypoint. In this particular case, half turn is enough to reach W3 altitude, thus, when it reaches the desired altitude, it holds it and crosses successfully waypoint W3 and, successively, waypoint W4.

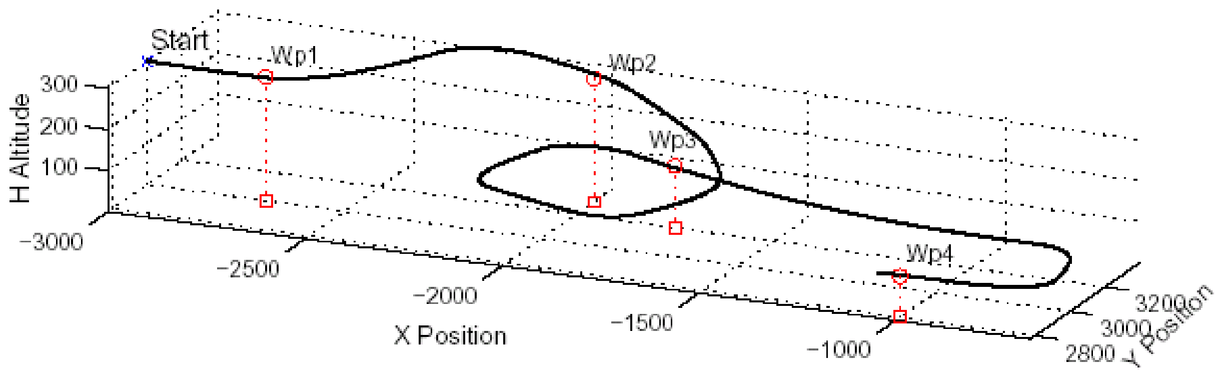


Figure 19. A 4-waypoint generated trajectory

Conclusions

The paper presents some issues on guidance, navigation and control problems related to unmanned aircraft. Methodolodgies based of maintaining desired formations shapes, and autonomous guidance were outlined.

References

[1] Bloy, A. W., West, M. G., Lea, K. A., and Jouma'a, M. "Lateral Aerodynamic Interference Between Tanker and Receiver in Air-to-Air Refuelling". *Journal of Aircraft*, Vol. 30 : pp. 705–710, 1993.
[2] Bloy, A. W. and Jouma'a, M. "Lateral and Directional Stability Control in Air-to-Air Refuelling". *ImechE, Part G Journal of Aerospace Engineering*, Vol. 209 : pp. 299–305, 1995.
[3] Myatt, J. H. and Blake, W. "Aerodynamics Database Issues for Modeling Close Formation Flight". In *Proceedings of AIAA Guidance, Navigation and Control Conference*, Portland, OR, August 1999.
[4] Gingras, D. "Experimental Investigation of a Multi-Aircraft Formation". In *Proceedings of AIAA Applied Aerodynamics Conference*, 1999.
[5] Gingras, D. and Player, J. "Static and Dynamic Wind Tunnel testing of Air Vehicles in Close Proximity". In *Proceedings of AIAA Atmospheric Flight Mechanics Conference*, Montreal, Quebec, Canada, August 2001.
[6] Nelson, R. C. and Jumper, E. J. "Aircraft Wake vortices and their Effect on Following Aircraft". In *Proceedings of AIAA Atmospheric Flight Mechanics Conference*, Montreal, Quebec, Canada, August 2001.
[7] Proud. A. W., Pachter, M., and D'Azzo, J. J. "Close Formation Flight Control". In *Proceedings of AIAA Guidance, Navigation and Control Conference*, Portland, OR, August 1999.
[8] Blake, W. and Multhopp, D. "Design, Performance and Modeling Considerations for Close Formation Flight". In *Proceedings of AIAA Guidance, Navigation and Control Conference*, Philadalphia, PA, August 1998.
[9] Houghton, E.L. and Brock, A.E. *"Aerodynamics for Engineering Students"*. Arnold Publishing, 1970.
[10] Buzogany, L. E., Pachter, M., and D'Azzo, J. J. "Automated Control of Aircraft in Formation Flight". In *Proceedings of AIAA Guidance, Navigation and Control Conference*, Monterey, CA, August 1993.
[11] Chichka, D. and Speyer, J. "Decentralized Controllers for Unmanned Aerial Vehicle Formation Flight". In *Proceedings of AIAA Guidance, Navigation and Control Conference*, San Diego, CA, July 1996.
[12] D. F. Chichka, J. L. Speyer, and C. G. Park. "Peak-Seeking Control with Application to Formation Flight". In *Proceedings of IEEE Conference on Decision and Control*, December 1999.
[13] J. J. D'Azzo M. Pachter and J.L. Dargan. "Automatic Formation Flight Control". *Journal of Guidance, Control and Dynamics*, 17(6):1380–1383, May 1994.
[14] M. Pachter, J. J. D'Azzo, and A. W. Proud. "Tight Formation Flight Control". *Journal of Guidance, Control and Dynamics*, 24(2):246–254, March-April 2001.
[15] C. Schumacher and R. Kumar. "Adaptive Control of UAVs in Close Formation Flight". In *Proceedings of American Control Conference*, Chicago, IL, June 2000.
[16] F. Giulietti, L. Pollini, and M. Innocenti. "Autonomous Formation Flight". *IEEE Control Systems Magazine*, 20(6):34–44, December 2000.
[17] Capetta, R., Giulietti, F., and Innocenti, M. "WakeCAD: Aerodynamic Interference Calculation Toolbox for Design, Simulation and Control of UAVs". In *Proceedings of AIAA Guidance, Navigation and Control Conference*, Montreal, Quebec, Canada, August 2001.
[18] Giulietti, F., Pollini, L., and Innocenti, M. "SNIPE: Development of an Unmanned Aerial Vehicle at DSEA". In *Proceedings of 15th Bristol International Conference on UAVs*, Bristol, UK, April 2000.
[19] M. R. Anderson and A. C. Robbins. "Formation Flight as a Cooperative Game". In *Proceedings of Aiaa Guidance, Navigation and Control Conference*, Philadelphia, PA, August 1998.
[20] J. Van Tyne and A.J. Berger. "*Fundamentals of Ornithology"*. John Wiley, 1976.
[21] M. Innocenti G.Mancino, M. Garofoli, and M. Napolitano. "Preliminary Analysis of Formation Flight Management". In *Proceedings of Aiaa Guidance, Navigation and Control Conference*, Portland, OR, August 1999.
[22] Takagi, T. and Sugeno, M., "Fuzzy Identification of systems and its applications to modeling and control," *IEEE Transaction on System Man and Cybernetics*, Vol. SMC, No. 15, Jan./Feb. 1985, pp. 116–132.
[23] Lin, C.-F., *Modern navigation, guidance and control processing*, Advanced Navigation Guidance and Control and their Applications, Prentice Hall, 1999.
[24] Giulietti, F., Pollini, L., and Innocenti, M., "Waypoint-based Fuzzy Guidance for Unmanned Aircraft," *Proceedings of* 15*th IFAC Symposium on Automatic Control in Aerospace, Bologna, IT.*, 2001.

[25] Pollini, L., Giulietti, F., and Innocenti, M., “SNIPE: Development of an Unmanned Aerial Vehicle at DSEA - University of Pisa,” *Proceedings of 15th Bristol International Conference on UAVs Conference 2000, Bristol, UK.*, 2000.
[26] Pollini, L., Giulietti, F., and Innocenti, M., “SNIPE: Development of an Unmanned Aerial Vehicle at DSEA - University of Pisa,” *Proceedings of UAV 2000 Conference, Paris, FR.*, 2000.

The Earth Phenomena Observing System: Intelligent Autonomy for Satellite Operations

Dr. Michael Ricard / Mr. Mark Abramson / Dr. David Carter / Dr. Stephan Kolitz
The Charles Stark Draper Laboratory, Inc.
555 Technology Square
Cambridge, MA 02139 USA

Introduction

Earth monitoring systems of the future may include large numbers of inexpensive small satellites, tasked in a coordinated fashion to observe both long term and transient targets. For best performance, a tool which helps operators optimally assign targets to satellites will be required. We present the design of algorithms developed for real-time optimized autonomous planning of large numbers of small single-sensor Earth observation satellites. The algorithms will reduce requirements on the human operators of such a system of satellites, ensure good utilization of system resources, and provide the capability to dynamically respond to temporal terrestrial phenomena. Our initial real-time system model consists of approximately 100 satellites and large number of points of interest on Earth (e.g., hurricanes, volcanoes, and forest fires) with the objective to maximize the total science value of observations over time. Several options for calculating the science value of observations include the following: 1) total observation time, 2) number of observations, and the 3) quality (a function of e.g., sensor type, range, slant angle) of the observations. An integrated approach using integer programming, optimization and astrodynamics is used to calculate optimized observation and sensor tasking plans.

Problem Motivation

"...Thus far, we are only experimenting with long term weather, climate, and natural hazard prediction. The quest for a true predictive capability for Earth system changes requires a flexible and progressive space system architecture that is responsive to our needs based on our current understanding of the system as well as accommodating emerging needs in the coming decades. We need to design and establish a smart, autonomous and flexible constellation [of] Earth observing satellites which can be reconfigured based on the contemporary scientific problems at hand. Such a constellation would exploit a combination of active and passive sensing sensors in ways that we can perhaps imagine today....."

This is a quote from the remarks of former NASA Administrator Daniel S. Goldin "The Frontier of Possibilities" presented at the International Astronautical Federation on October 3, 2000. It clearly describes the underlying rationale for the multi-year project in which we are developing algorithms for resource allocation via autonomous reconfiguration of satellite webs consisting of heterogeneous Earth observation sensor platforms.

The development of these algorithms and a simulation testbed in which they reside, the Earth Phenomena Observing System (EPOS), will reduce requirements on the human operators of satellites, improve system resource utilization, and provide the capability to dynamically respond to temporal terrestrial phenomena. Examples of triggering events are localized transient phenomena that have a significant impact on human life such as volcanic eruptions, weather (hurricanes, tornadoes, etc.), algae plumes, large ocean vortices, ice shelf break-up, seismic activities, oil spills, magnetic anomalies, and search and rescue.

Vision

In the year 2020, automated mission management will be required to help NASA achieve their Sensorweb vision. Draper's vision is to provide functionality for that mission manager, EPOS 2020. Key technology required for the full EPOS 2020 is being developed under our current 2 to 3 year effort.

In 2020, many Earth observing systems will be in place, each with its own ground facilities, communication and data protocols, satellite characteristics and observation schedules. There will be hundreds of Earth observing satellites and relatively few will be identical. In most systems, satellite maneuvering will be reserved for station-keeping only. From EPOS perspective these are "coasting satellites" – EPOS observation schedules must utilize the satellite's existing orbit. Some of the systems have satellites that can be dynamically tasked through in-plane maneuvering. For EPOS purposes these are referred to as "maneuvering satellites." These satellites will be easily upgraded and refueled using technology currently being developed [5]. We hypothesize there will be a coordinating ground station that has: 1) communication with these Earth observing systems, 2) a given set of data/information interchange protocols that provides the mechanism to exchange data with these varied systems, 3) visibility into whatever parameters are needed to model these systems, and 4) the capability to influence the observation schedules of these systems. This is illustrated in Figure 1. When a target becomes known, the coordinating ground station calculates which of the satellites controlled by the Earth observing systems can provide suitable information at the appropriate times and uses that knowledge to send appropriate *viewing requirements* to the individual system's ground stations. The individual ground stations look more closely at the availability and appropriateness of the satellites within its control, especially dealing with the priority of all requests for satellite assets. An inability to satisfy the *viewing requirements* is communicated back to the coordinating ground station.

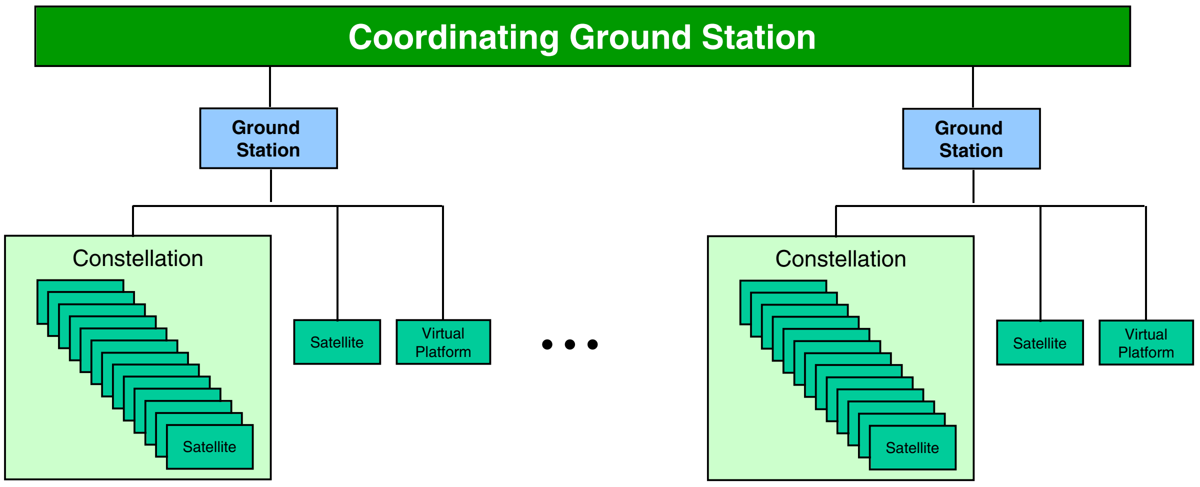


Figure 1: Year 2020 Operation Concept

This concept of operations includes planning and control of constellations and collaborative groups. Constellations are defined as a system employing two or more spacecraft whose orbits, operations, and observations are coordinated to provide global coverage or to improve temporal resolution from an altitude below GEO [4]. Constellations operate within constraints (e.g., keeping relative position in an orbital plane) that a planning and control system must recognize. Collaborative groups are temporary sets of satellites/virtual platforms that have commonality in their objectives; satellites can simultaneously belong to two collaborative groups. These groups "exist" (i.e., are formed) in the coordinating ground facility for the time period during which they are executing a plan. In our concept of operations, a satellite within a constellation can also be a member of a collaborative group as long as the constellation constraints are not violated.

Approach

The implementation of EPOS combines Draper's autonomous system's planning framework [3], which is used to decompose the problem into tractable subproblems or levels. Each level is then solved with a combination of combinatorial optimization combined with astrodynamic modeling.

Planning Framework

The basic building block of Draper's planning and control architecture for autonomous systems [3] is shown in Figure 2. It is an extension of the *sense-think-act* paradigm of intelligence, and is similar to the military's Observe-Orient-Decide-Act loop [2]. The key elements are modules for situation assessment, plan generation, plan implementation, and coordination

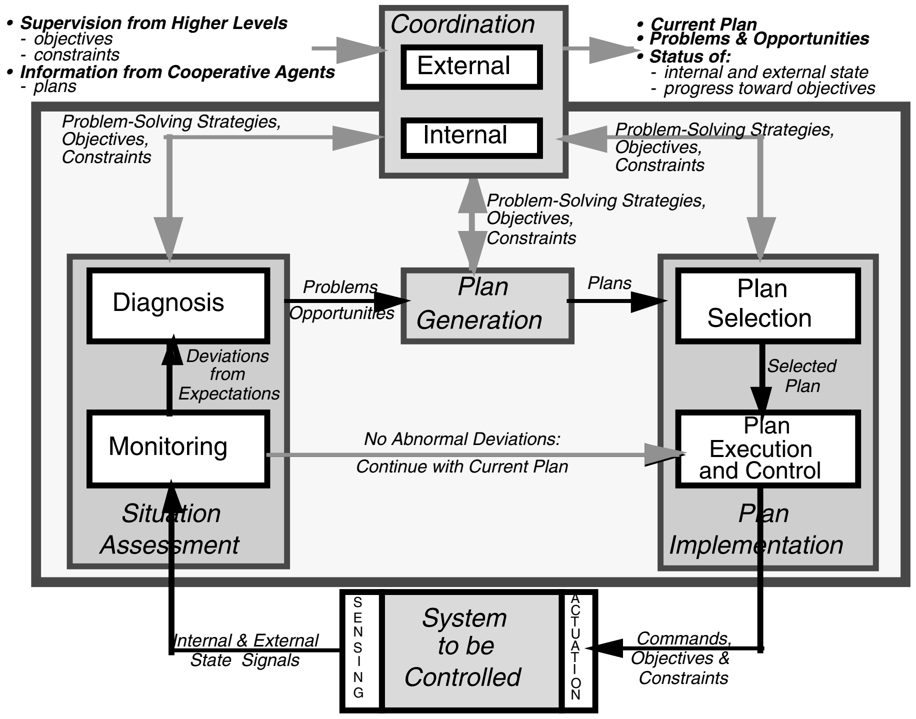

• Supervision from Higher Levels
- objectives
- constraints
• Information from Cooperative Agents
- plans
Coordination
External
Internal
• Current Plan
• Problems & Opportunities
• Status of:
- internal and external state
- progress toward objectives
Problem-Solving Strategies, Objectives, Constraints
Problem-Solving Strategies, Objectives, Constraints
Problem-Solving Strategies, Objectives, Constraints
Diagnosis
Problems
Opportunities
Plan Generation
Plans
Plan Selection
Deviations from Expectations
Selected Plan
Monitoring
No Abnormal Deviations:
Continue with Current Plan
Plan Execution and Control
Situation Assessment
Plan Implementation
SENSING
System to be Controlled
ACTUATION
Internal & External State Signals
Commands, Objectives & Constraints

Figure 2: Planning and Control Architecture Modules

The planning problem addressed in our effort is complex enough to warrant hierarchically decomposing it in order to make it tractable. Replanning takes place at the lowest level possible, without disturbing other plans unless additional resources are needed. Hierarchical decomposition is appropriate for applications in which there is significant stochasticity. In such systems, it is not practical to make detailed plans too far into the future, since the state of the world (e.g., new targets of interest) and the state of the system (e.g., a satellite sensor fault) can change.

In order to make the optimization problem tractable, hierarchical decomposition is used both temporally and functionally. The decomposition is characterized by higher levels that create plans with the greatest temporal scope (longest planning horizon) but with the least detail. At lower levels, the planning horizon becomes shorter (nearer term), but the level of detail of planned activities increases. The less detailed plans at the higher levels coordinate or guide the generation of solutions generated at the lower levels. Indeed, planning actions over extended periods of time at a high level of detail is typically both futile and impractical. Futile because detailed actions planned on the basis of a specific prediction of the future may become obsolete well before they are to be executed due to an inability to accurately predict the future. Impractical because the computational resources required to develop detailed plans over extended periods of time may be prohibitive either in cost or availability or both. The relationship between the levels of the hierarchy and the planning horizon and level of plan detail is shown in Figure 3.

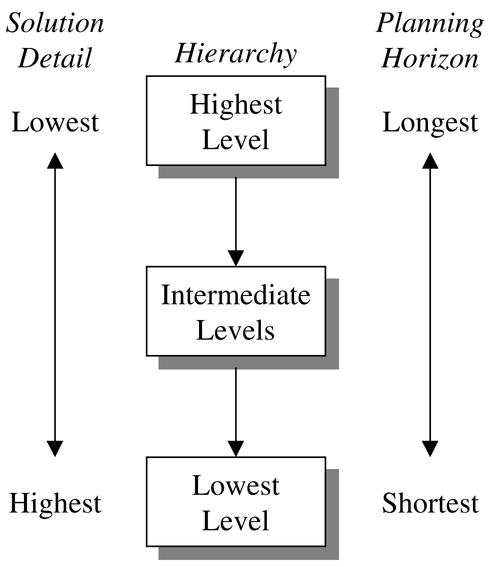


Figure 3: Temporal Decomposition

One way of grouping the types of decisions that need to be made for EPOS is shown in Figure 4. Three decision tiers are present: 1) System, the top tier focuses on decisions that impact the entire EPOS (e.g., which targets to collect data for); 2) Collaborative Group, the middle tier addresses issues that relate to a group of multiple satellites being used to collect data on a target (e.g., which satellites make up this group); and 3) Satellites, the lowest tier addresses decisions relevant to the individual satellites making up each collaborative group (e.g., what burns should be executed to achieve a certain level of required coverage). This decomposition is natural for satellite operations.

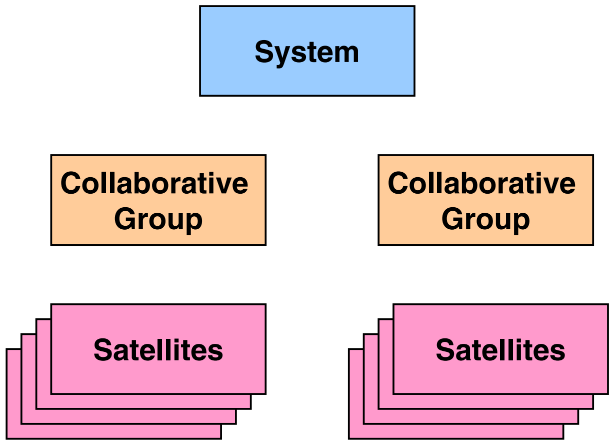


Figure 4: Three Tiered EPOS Hierarchy

Another decomposition, one based around seven key levels of decision-making is describe in Table 3-6. The longer term, EPOS-wide issues are determined at the upper levels, while shorter-term decisions are lower down in the list. This is a decision-centric approach to decomposition compared with the entity-centric approach of the three tiers. The planning and control architecture modules of Figure 2 can be applied within each of these decision levels. Note that the description of the decision levels shown in Figure 5 presents the capabilities of a complete EPOS system, including functionality that has not yet been implemented.

	Mission Manager Levels	Decisions
System Tier	System Level	Select locations on Earth for observation; for each location being observed: what are the candidate satellite platforms?
	Configuration Level	Which satellites need to be refueled? Which satellites need to be launched?
Collaborative Group Tier	Platform Assignment Level	Which satellites are to be actually used for observation?
	Observation Level	For each satellite, when and for what target should the sensors be used?
Satellite Tier	Maneuver Level	For each satellite, when and with what Dv should the maneuvers be made?
	Sensor Level	For each sensor, when and in which direction should it be pointed?
	Data and Communication Level	For each satellite, what data to store when, and what data to communicate when?

Figure 5: Decision Levels

The development of EPOS is an evolving process. Previous versions of EPOS [1] focused on an initial version of the observation level (which included a person-in-the-loop) and an autonomous implementation of the maneuver level. The current version of EPOS, 2.0, is focused on an autonomous implementation of the observation level as well as the sensor planning levels. The optimization algorithms employed by each of those levels rely on data that must be derived through astrodynamic modeling. That modeling is described in the next subsection.

Astrodynamics

Observation planning algorithms require knowledge of the evolution of target illumination and of sensor-to-target relative position. Dynamic models are used to answer the following essential questions for each time t in the planning interval:

- Which targets are in sunlight?
- Which satellites can potentially view a given target, in the sense that the line of sight from satellite to target is not obstructed?
- How must the sensor on such a satellite be oriented to have the target at the center of its field of view?
- Paying attention to pointing angle limits, can the sensor be oriented so that the target is somewhere in its field of view?

Write $r(t)$ for the earth-centered inertial (ECI) coordinates of an earthbound target at time t, and $v(t)$ for a unit vector normal to the earth's surface (local vertical) at $r(t)$. Both $r(t)$ and $v(t)$ are determined from target geodetic latitude, longitude, and time, using an analytic representation for the right ascension of Greenwich.

To determine if the target is in sunlight at time t, we need ECI coordinates $u(t)$ for the direction to the sun. An analytic representation is used. The target is in sunlight when the dot product $v(t)\cdot u(t)$ is positive.

Satellite orbital motion is modeled using the simple J_2 secular theory, but the software could be easily modified to use a more sophisticated orbit propagator. ECI position coordinates, *s(t),* for a satellite are computed from its orbit elements using standard two-body mechanics formulas.

The look direction, from satellite with coordinates *s(t)* to target with coordinates *r(t),* is computed as follows:

$$L_{rs}(t) = \frac{r(t) - s(t)}{\|r(t) - s(t)\|}$$

The satellite can potentially view the target, in the sense that the line of sight is not obstructed by the earth, if the following dot product condition holds:

$$v(t) \cdot L_{rs}(t) < 0$$

Satellite attitude must be modeled in order to answer questions about sensor orientation. Let $A_s(t)$ denote the nominal attitude of the satellite with coordinates *s(t)* at time *t*. The development version of our system assumes sun-nadir steering for each satellite; one spacecraft axis points to nadir and a second axis is normal to the plane spanned by nadir and the sun direction. An operational planning tool would support considerably more flexible specification of satellite attitude profiles.

Attitude $A_s(t)$ is understood to be a coordinate transformation from ECI frame to nominal spacecraft body frame. Sensor orientation is specified as a second coordinate transformation, $\tilde{A}_s(t)$, from nominal spacecraft body frame to a frame fixed in the sensor. The composite transformation, $\tilde{A}_s(t)A_s(t)$, could be realized in either of two ways:

- Steer the spacecraft so that $A_s(t)$ is the transformation from ECI to actual spacecraft body frame. Gimbal the sensor so that $\tilde{A}_s(t)$ is the transformation from spacecraft body frame to sensor frame.
- Steer the spacecraft only, so that $\tilde{A}_s(t)A_s(t)$ is the transformation from ECI to the sensor frame.

In either case, we regard $A_s(t)$ as given and $\tilde{A}_s(t)$ as something to be chosen.

Let ℓ be a unit vector in the sensor look direction, expressed in the sensor frame. The target is at the center of the sensor field of view if the following condition holds:

$$\tilde{A}_s(t)A_s(t)L_{rs}(t) = \ell$$

This condition determines $\tilde{A}_s(t)$ up to rotation about ℓ. It determines the first two angles $x(t)$ and $y(t)$ in a suitable Euler angle sequence, in other words.

The target remains somewhere in the sensor field of view provided that these Euler angles are chosen from intervals $[x(t) - \Delta x, \quad x(t) + \Delta x]$ and $[y(t) - \Delta y, \quad y(t) + \Delta y]$, respectively. Here Δx and Δy are constants of the sensor (field of view half-widths).

Four additional parameters of the sensor, $x_{\min}$, $x_{\max}$, $y_{\min}$ and $y_{\max}$ are used to specify feasible intervals for the two Euler angles. For the sensor to be able to view the target at time *t*, it is necessary that the line of sight be unobstructed (dot product condition given earlier) and that the intervals I_x and I_y be nonempty, where

$$I_x = [x(t) - \Delta x, \quad x(t) + \Delta x] \cap [x_{\min}, \quad x_{\max}]$$
$$I_y = [y(t) - \Delta y, \quad y(t) + \Delta y] \cap [y_{\min}, \quad y_{\max}]$$

Observation Planning

The observation planning level's problem is to find the primary target for each sensor so that the *total science value at time t* (including extra for simultaneous viewing) is maximized provided that:

- no more than one of each type of sensor is tasked for each target[1],
- no more than two (different) sensors are assigned per target
- demand for data storage on a satellite does not exceed available capacity.

The problem is specified through static data that describes the available satellites and their associated ground stations and dynamic data that specifies that targets of interest, which targets can be viewed by a particular satellite and which ground stations can communicate with a satellite at a given time. In the discussion that follows, the terms satellite and sensor are often used interchangeably. The schedule is being done for each *sensor*. Conceptually, one can think of a satellite containing a single sensor. It is of no consequence to the formulation that follows if multiple sensors are contained on a single satellite. To being the derivation of the problem, the following sets must be defined:

$$\begin{aligned}
N &= \text{number of satellites/sensors} \\
M &= \text{number of targets} \\
L &= \text{number of ground stations} \\
S_{EO} &= \text{Set of sensors of type EO} \\
S_{IR} &= \text{Set of sensors of type IR} \\
S_{SAR} &= \text{Set of sensors of type SAR}
\end{aligned}$$

Dynamic Inputs

The dynamic inputs to the observation level optimization specify data that is changed from one pass to the next. The set of targets that a particular sensor can view at time *t* are specified as:

$$o_{ij}^{t} = \begin{cases} 1 \text{ if satellite/sensor } i \text{ can observe target } j \\ 0 \text{ otherwise} \end{cases}$$

The set of ground stations that a satellite can communicate with at time *t* are specified as

$$q_{ig}^{t} = \begin{cases} 1 \text{ if satellite/sensor } i \text{ can communicate with ground station } g \\ 0 \text{ otherwise} \end{cases}$$

The values of o_{ij}^{t} and q_{ij}^{t} are derived using the methods described in the subsection describing astrodynamics.

Whenever one or more sensors view a target, scientific value is gained by the observation. This value is specified quantitatively as:

$$\begin{aligned}
u_{ij} &= \text{value from satellite/sensor } i \text{ observing target } j \\
w_{i_1 i_2 j} &= \text{additional value from satellites/sensors } i_1 \text{ and } i_2 \text{ observing target } j \text{ simultaneously}
\end{aligned}$$

[1] This is a simple model to handle the assumption that two observations by the same type sensor during any time period does not have twice the value of one observation.

Decision Variables

In time period *t*, the decision variables are:

$$x_{ij}^t = \begin{cases} 1 \text{ if satellite/sensor } i \text{ observes target } j \\ 0 \text{ otherwise} \end{cases}$$

$$y_{i_1 i_2 j}^t = \begin{cases} 1 \text{ if satellites/sensors } i_1 \text{ and } i_2 \text{ observe target } j \text{ simultaneously} \\ 0 \text{ otherwise} \end{cases}$$

Static Data

Static data is used to specify the data storage capabilities of the satellites. This data is static in the sense that it is particular to a sensor/satellite only and does not depend on any targets or their visibility status. There is an assumption that ground resources are sufficient for any number of the satellites to be downloading simultaneously. Thus each satellite picks the ground station with which it has the maximum transfer rate.

K_i	=	total capacity of satellite i (number of observation units that can be stored)
k_i^t	=	capacity on satellite i at the start of time period t
a_{EO}	=	data units per observation consumed by an EO sensor
a_{IR}	=	data units per observation consumed by an IR sensor
a_{SAR}	=	data units per observation consumed by an SAR sensor
d_i^t	=	maximum number of data units during time period t that can be downloaded from satellite i when a ground station is visible

The values of k_i^t can then be defined recursively as:

$$k^t{}_i = \begin{cases} k^{t-1}{}_i - \left(\sum_{i \in I_{EO,j}} x^{t-1}{}_{ij} a_{EO} + \sum_{i \in I_{IR,j}} x^{t-1}{}_{ij} a_{IR} + \sum_{i \in I_{SAR,j}} x^{t-1}{}_{ij} a_{SAR} \right) + d^{t-1}{}_i & \text{if } \sum_{g=1\, to\, L} q^{t-1}{}_{ig} > 0 \\ k^{t-1}{}_i - \left(\sum_{i \in I_{EO,j}} x^{t-1}{}_{ij} a_{EO} + \sum_{i \in I_{IR,j}} x^{t-1}{}_{ij} a_{IR} + \sum_{i \in I_{SAR,j}} x^{t-1}{}_{ij} a_{SAR} \right) & \text{if } \sum_{g=1\, to\, L} q^{t-1}{}_{ig} = 0 \end{cases}$$

Formulation

Given the data definitions specified above, the mathematical programming formulation is as follows for each time period *t*.

$$\max \sum_{j} \left(\begin{array}{l} \sum_{i\in S_{EO}} x^t{}_{ij} u_{ij} o^t_{ij} + \sum_{i\in S_{IR}} x^t{}_{ij} u_{ij} o^t_{ij} + \sum_{i\in S_{SAR}} x^t{}_{ij} u_{ij} o^t_{ij} + \\ \sum_{i_1\in S_{EO}, i_2\in S_{IR}} y^t{}_{i_1,i_2,j} w_{i_1,i_2,j} o^t_{i_1} o^t_{i_2} + \sum_{i_1\in S_{EO}, i_2\in S_{SAR}} y^t{}_{i_1,i_2,j} w_{i_1,i_2,j} o^t_{i_1} o^t_{i_2} + \sum_{i_1\in S_{SAR}, i_2\in S_{IR}} y^t{}_{i_1,i_2,j} w_{i_1,i_2,j} o^t_{i_1} o^t_{i_2} \end{array} \right)$$

Subject to

$$\begin{array}{rl}
\sum_{j=1..M} x^t_{ij} \le k^t_i & \text{for } i = 1 \text{ to } N \\
\sum_{j=1 \text{ to } M} x^t{}_{ij} \le 1 & \text{for } i = 1 \text{ to } N \\
\sum_{i\in S_{EO}} x^t{}_{ij} \le 1 & \text{for } j = 1 \text{ to } M \\
\sum_{i\in S_{IR}} x^t{}_{ij} \le 1 & \text{for } j = 1 \text{ to } M \\
\sum_{i\in S_{SAR}} x^t{}_{ij} \le 1 & \text{for } j = 1 \text{ to } M \\
\sum_{i=1 \text{ to } N} x^t_{ij} \le 2 & \text{for } j = 1 \text{ to } M \\
x^t{}_{i_1 j} + x^t{}_{i_2 j} - 2y^t{}_{i_1,i_2,j} \ge 0 & \text{for } i_1 \ne i_2,\ j = 1 \text{ to } M \\
x^t{}_{i_1 j} + x^t{}_{i_2 j} - 2y^t{}_{i_1,i_2,j} \le 1 & \text{for } i_1 \ne i_2,\ j = 1 \text{ to } M \\
x^t{}_{ij} \in \{0,1\} & \text{for } i = 1 \text{ to } N,\ j = 1 \text{ to } M \\
y^t{}_{i_1,i_2,j} \in \{0,1\} & \text{for } i = 1 \text{ to } N,\ j = 1 \text{ to } M
\end{array}$$

The output of the top-level optimization is at each time *t* and for each sensor the primary target it is tasked to look at and whether the satellite downloads observation data or not.

Sensor Planning

The assignments of targets to sensors at each time period are input to the sensor optimization level. The output is a pointing command for each sensor for each time period. The sensors are modeled as being mounted on platform with two orthogonal gimbals. These gimbals are referred to as the *x* and *y* gimbal angles.

Data

To begin the definition of the data required for this optimizer, first define the following sets:

$$R = \text{the set of targets}$$
$$T = \text{the time horizon for the problem}$$

The model of the sensor includes constraints on the movement of each of the gimbals. These constraints include both bounds on the gimbal angles and slew rate constraints. The following data is defined for each $t = 1..T$.

$B(t)$	=	$\{r \in R \mid r$ is assigned to the sensor at time $t\}$
$A_r(t).[lu][xy]$	=	required bounds on the gimbal angles for the sensor to be able to see target $r \in B(t)$ at time t (e.g. $A_r(t).ux$ is upper bound on x direction)
$P.[lu][xy]$	=	gimbal angle limits for the sensor in each direction (rad)
$S.[xy]$	=	slew rate limits for the sensor in each direction (rad/Δt)

The notation *[xy]* is used to represent combinations of the specified variables. For example, *S.[xy]* indicates that both variables *S.x* and *S.y* are defined.

Decision Variables

The primary decision variables are:

$$p(t).[xy] \quad = \quad \text{gimbal angle for the sensor at time } t \text{ for each direction}$$

$$w_r(t) \quad = \quad \begin{cases} 1 & \text{if the sensor can see target } r \in B(t) \text{ at time } t \\ 0 & \text{otherwise} \end{cases}$$

The variables are defined for all $t = 1..T$ and the values of *p(0).[xy]* are specified as input. The $w_r(t)$ variables are obviously a function of the $p(t)$ variables. Specifically, $w_r(t) = 1$ if and only if the following are true:

$$A_r(t).lx \le p(t).x \le A_r(t).ux$$
$$A_r(t).ly \le p(t).y \le A_r(t).uy$$

In support of this, define the following binary variables:

$$b_r(t).lx = \begin{cases} 1 \text{ if } A_r(t).lx \le p(t).x \\ 0 \text{ otherwise} \end{cases} \qquad b_r(t).ux = \begin{cases} 1 \text{ if } p(t).x \le A_r(t).ux \\ 0 \text{ otherwise} \end{cases}$$
$$b_r(t).ly = \begin{cases} 1 \text{ if } A_r(t).ly \le p(t).y \\ 0 \text{ otherwise} \end{cases} \qquad b_r(t).uy = \begin{cases} 1 \text{ if } p(t).y \le A_r(t).uy \\ 0 \text{ otherwise} \end{cases}$$

The relationship between $w_r(t)$ and $b_r(t)$ is then

$$4w_{vr}(t) \le b_{vr}(t).lx + b_{vr}(t).ux + b_{vr}(t).ly + b_{vr}(t).uy$$

The $b_r(t)$ variables are set with the following constraints

$$p(t).x \ge A_r(t).lx - M(1 - b_r(t).lx)$$
$$p(t).x \le A_r(t).ux + M(1 - b_r(t).ux)$$
$$p(t).y \ge A_r(t).ly - M(1 - b_r(t).ly)$$
$$p(t).y \le A_r(t).uy + M(1 - b_r(t).uy)$$

where M is a larger than the absolute value of any $A_r(t)$ entry.

Constraints

The pointing limits for the sensor are specified simply as:

$$P.lx \le p(t).x \le P.ux \qquad \forall t = 1..T$$
$$P.ly \le p(t).y \le P.uy \qquad \forall t = 1..T$$

The slew rates are specified as:

$$\left| p(t).x - p(t-1).x \right| \le S.x \qquad \forall t = 1..T$$
$$\left| p(t).y - p(t-1).y \right| \le S.y \qquad \forall t = 1..T$$

Objective Function

The objective function maximizes the value gained from achieving all of the assignments. This is calculated by multiplying the $w_r(t)$ variables by $V_r(t)$.

The null position of the sensor is pointing at the long-term targets, unless there is value to be gained from an ephemeral target[2], the sensor should try to point in the null position. To address this, a cost will be attributed to moving the sensor from the null position. Specifically, a term will be added to the objective function to minimize the absolute value of the $p(t).[xy]$ variables. Define the non-negative variables $D^{[+-]}(t).[xy]$ as:

$$D^{+}(t).x - D^{-}(t).x = p(t).x$$
$$D^{+}(t).y - D^{-}(t).y = p(t).y$$

The objective function will include a term to minimize

$$\sum_{t=1}^{T}\left(D^{+}(t).x + D^{-}(t).x + D^{+}(t).y + D^{-}(t).y\right)$$

Formulation

Given the data definitions specified above, the mathematical programming formulation is as follows.

$$\max \sum_{t=1}^{T}\left[\sum_{r\in B_v(t)} V_r(t)w_{vr}(t) - \gamma\left(D^{+}(t).x + D^{-}(t).x + D^{+}(t).y + D^{-}(t).y\right)\right]$$

Subject to

$$\begin{aligned}
p(t).x - p(t-1).x &\le S.x && \forall t = 1..T\\
p(t).x - p(t-1).x &\ge -S.x && \forall t = 1..T\\
p(t).y - p(t-1).y &\le S.y && \forall t = 1..T\\
p(t).y - p(t-1).y &\ge -S.y && \forall t = 1..T
\end{aligned}$$

$$\begin{aligned}
p(t).x - Mb_r(t).lx &\ge A_r(t).lx - M && \forall r \in B_v(t), t = 1..T\\
p(t).x + Mb_r(t).ux &\le A_r(t).ux + M && \forall r \in B_v(t), t = 1..T\\
p(t).y - Mb_r(t).ly &\ge A_r(t).ly - M && \forall r \in B_v(t), t = 1..T\\
p(t).y + Mb_r(t).uy &\le A_r(t).uy + M && \forall r \in B_v(t), t = 1..T
\end{aligned}$$

$$4w_r(t) - b_r(t).lx - b_r(t).ux - b_r(t).ly - b_r(t).uy \le 0 \qquad \forall r \in B_v(t), t = 1..T$$

$$\begin{aligned}
D^{+}(t).x - D^{-}(t).x - p(t).x &= 0 && \forall t = 1..T\\
D^{+}(t).y - D^{-}(t).y - p(t).y &= 0 && \forall t = 1..T
\end{aligned}$$

$$\begin{aligned}
p(t).x &\in [P.lx, P.ux]\\
p(t).y &\in [P.ly, P.uy]\\
D^{[+-]}(t).[xy] &\in \Re\\
w_r(t) &\in \{0,1\}\\
b_r(t).[lu][xy] &\in \{0,1\}
\end{aligned}$$

[2] The value from observing an ephemeral target is assumed to be greater than observing a long-term target.

Future Work

The development of the EPOS system has progressed so that both maneuvering and coasting satellites can be optimized to achieve optimal target viewing. A limitation of the current system is that maneuvering satellites treat their sensors as having a fixed attitude while coasting satellites utilize an autonomous sensor optimization level. The next version of EPOS, currently being developed, will merge these two approaches. Specifically, both maneuvering and coasting satellites will be addressed and all satellites will optimize the pointing of each sensor. In effect, the current version of EPOS and the capabilities of the previous version [1] will be merged into a single approach.

Acknowledgments

This material is based upon work supported by the National Aeronautics and Space Administration's AIST program, under Contract #NAS3-00163, through the Glenn Research Center. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Aeronautics and Space Administration.

References

[1] Abramson, M., et al., “The Design and Implementation of Draper’s Earth Phenomena Observing System (EPOS),” in *Proceedings of the AIAA Space 2001 Conference*, Albuquerque, NM, August 2001.

[2] Osborne, W. LTC, United States Army, et al. Information Operations: A New War-Fighting Capability, August 1996. (http://www.au.af.mil/au/2025/-volume3/chap02/vol3ch02.pdf).

[3] Ricard, M. and S. Kolitz, “The ADEPT Framework for Intelligent Autonomy,” in *VKI Lecture Series on Intelligent Systems for Aeronautics*, von Karman Institute, Belgium, May 13-17, 2002.

[4] Ticker, R. L., and J. D. Azzolini, *2000 Survey of Distributed Spacecraft Technologies and Architectures for NASA’s Earth Science Enterprise in the 2010-2025 Timeframe*, NASA/TM-2000-209964, August 2000.

[5] Wall, R., “Boeing Snags Demonstration to Refuel, Upgrade Satellites,” *Aviation Week & Space Technology*, March 18, 2002.

Autonomy Needs and Trends in Deep Space Exploration

Dr. Richard J. Doyle
Manager, Information Technology and Software Systems Division
Leader, Center for Space Mission Information Systems and Software
Jet Propulsion Laboratory
California Institute of Technology
MS 126-221/4800 Oak Grove Drive
Pasadena, California 91109-8099 USA

rdoyle@jpl.nasa.gov
http://it.jpl.nasa.gov, http://csmiss.jpl.nasa.gov, http://cs.jpl.nasa.gov

Abstract: The development of onboard autonomy capability is the key to a set of vastly important strategic technical challenges facing NASA: increased efficiency in the return of quality science products, reduction of mission costs, and the launching of a new era of solar system exploration characterized by sustained presence, in-situ science investigations and missions accomplished via multiple, coordinated space platforms.

Autonomy is a central capability for enabling missions that inherently must be accomplished without the benefit of ongoing ground support. This constraint may arise due to control challenges, e.g., small-body rendezvous and precision landing, or may arise due to mission planning challenges based in the difficulty of modeling the planetary environment coupled with the difficulty or impossibility of communications during critical or extended periods. A sophisticated Mars rover, a comet lander, a Europan under-ice explorer, and a Titan aerobot are examples of missions, some unprecedented, which typify these challenges.

This paper describes the set of NASA missions that aim to utilize autonomy and recent developments in the creation of space platform autonomy capabilities at NASA.

1. THE NASA MISSION CHALLENGES

NASA is embarking on a new phase of space exploration. In the solar system, an initial reconnaissance of all of the planets except Pluto has been accomplished. In the next phase of planetary exploration, the emphasis will be on direct, i.e., in-situ scientific investigation in these remote environments. In the next phase of exploration relating to astrophysics, the emphasis is on new observing instruments – often based on principles of interferometry – to achieve unprecedented resolution in remote observing. A theme that runs through all of these science missions is the search for life.

The development of onboard autonomy capability is on the critical path to addressing a set of vastly important strategic technical challenges arising from the NASA mission set: increased efficiency in the return of quality science products, reduction of mission costs, and the launching of a new era of solar system exploration characterized by sustained presence, in-situ science investigations and missions accomplished via multiple, coordinated space platforms. These new classes of space exploration missions, as a rule, require new capabilities and technologies.

1.1 Mars and Autonomy

Mars is a primary target for future exploration, and certainly has captured the interest of the general public. The set of Mars missions under development differ from previous space exploration in one important aspect: they are being conceived as a collective whole, with the establishment and evolution of infrastructure at Mars as an important sub-goal.

The next rover mission to Mars will be the Mars Exploration Rover (MER) mission, which will place two rovers on the surface of Mars in 2004. This mission will extend the accomplishments of the famous 1997 Mars Pathfinder Sojourner rover mission in several ways. The larger size of the physical rover platforms allows for significantly upgraded payloads of half a dozen sophisticated instruments and tools on each rover. The traverse objectives for the rovers will be 100 meters per day, in contrast to Sojourner, which did not depart the vicinity of the lander. Finally, the nominal mission lasts 90 sols (Martian days), compared to 30 for Mars Pathfinder.

The Mars mission following MER will be the Mars Reconnaissance Orbiter (MRO) mission, in 2005. Although MRO will not contribute new autonomy capabilities, it is an example of how missions in the Mars program are related and support each other. MRO is expected to be a communications relay for future Mars surface missions, and more specifically, it will collect high resolution surface images, and these images will be utilized to plan landing sites for the more sophisticated Mars rover mission to follow, called Mars Smart Lander (MSL), in 2009.

In the words of the MSL project manager, this mission will be "the most software-intensive, autonomy-dependent mission" to date. The emphasis on autonomy capabilities derive directly from the report generated by the study of the project's Science Definition Team (SDT). Scientists are desirous of greater efficiency and reliability in commanding planetary surface rovers to generate science return, following the exciting, but often tedious experience with Sojourner on Mars Pathfinder, and anticipating that improvements on Mars Exploration Rover may be of an incremental nature.

The autonomy capability requirements generated by scientists on the SDT generally fall into three areas: the ability to command the rover, in a single command cycle, to move to a destination beyond the visible horizon; the ability, in a single command cycle, to reliably place an instrument on a designated target; and, for the first time, the ability to perform some limited science activities while traversing from one identified science investigation site to another.

The key phrase in the foregoing is "in a single command cycle." Scientists and mission operators together have experienced frustration at the number of iterations sometimes required to reach a desired location or to successfully place an instrument. When the complicated constraints of light-time delay, in-view communication periods, and solar illumination cycles on Mars are factored in, such inefficiencies translate into serious limitations on overall science return. Conversely, if such actions can be accomplished autonomously and reliably in a single command cycle, overall science return is boosted greatly.

Another autonomy need on MSL is for safe and precise landing. The mission plans to utilize active hazard detection and avoidance capabilities during descent not only to achieve safe landing, but also to ensure that the target science sites are accessible from the landing site.

Looking further in to the future, infrastructure on Mars may include permanent science stations on the surface, propellant production plants, and a network of communications satellites in orbit to extend internet-like capability to Mars, and to enable the coordination of an array of heterogeneous, autonomous explorers: rovers, balloons, airplanes, even subsurface devices. No longer would each mission be conceived and executed in isolation, but through a combination of in situ and multiple platform mission concepts humanity's presence at Mars would continually expand, culminating in the arrival and safe return of the first human explorers. See Figure 1.

Figure 1. Future Mars Missions:
Mars Exploration Rover, Mars Smart Lander, Mars Outposts

1.2 Small Planetary Bodies and Autonomy

Small planetary bodies – comets and asteroids – pose different kinds of challenges to space platforms that would investigate them in situ. Their gravitational environments are more difficult to model, particularly if the shape of the body is well off the spherical, and if it is tumbling rather than rotating. Achieving stable orbit is a complex task, much less landing. The environment of a comet is the most unpredictable: gas and dust and perhaps larger chunks of material can present hazards as well as make it difficult to select and track scientific targets of investigation.

The next mission to grapple with a cometary environment will be the Deep Impact mission, which will rendezvous with Comet Tempel 1 in 2005. The space platform for this mission consists of a flyby spacecraft and a detachable impactor. The impactor is designed to excavate below the surface of the cometary nucleus and reveal primordial material thought to be unchanged from the origin of the solar system. The targeting of the impactor has to be autonomous, because there is insufficient time to return images, analyze them, and send appropriate commands to the impactor. The image processing to guarantee successful impact is complicated by lighting, uncertain surface topography, and the inherent uncertainties of the environment.

The proposed Comet Nucleus Sample Return (CNSR) mission will involve autonomous landing and the return of samples. The unpredictable and volatile cometary environment will amplify the requirements for hazard detection and avoidance capabilities during descent and while on the surface. This mission concept calls for multiple site investigations, i.e., multiple takeoffs and landings in the course of the mission. The engineering and science considerations of autonomy merge in small body missions, as the same phenomena which represent potential engineering hazards (e.g., the formation of jets on comets), are also the phenomena of scientific interest. See Figure 2.

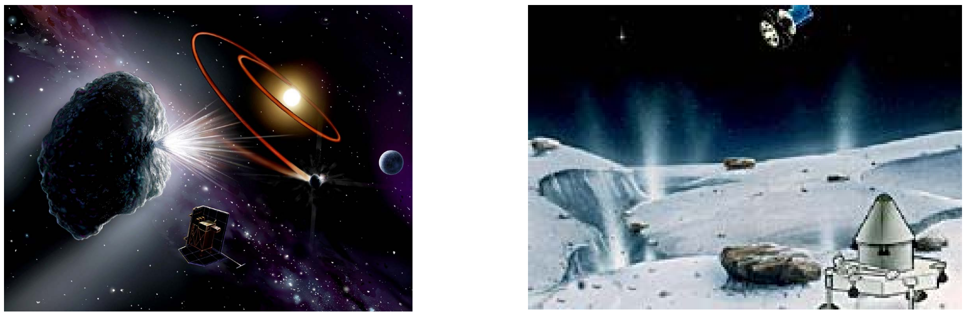


Figure 2. Future Small Body Missions: Deep Impact and Comet Nucleus Sample Return

1.3 Other Planetary Targets and Autonomy

The proposed Titan Organics Explorer would utilize a combination of platform concepts to conduct pre-biological chemistry and atmospheric investigations at Saturn's intriguing satellite, long known to possess an atmosphere, organic materials, and the possibility of a non-water ocean in an entirely different temperature regime. The platform concepts for this mission include an orbiter combined with an aerobot with detachable rover deployables. Aerobots utilize natural thermal cycles to periodically go aloft to sample multiple sites over a wide range of territory. When worthy science sites are found, the in situ investigation capabilities of surface explorers like rovers are then utilized. This type of exploration has a random element however, since landings can be only semi-directed, using direct control only of the vertical dimension, along with knowledge of wind patterns.

Europa is a notable focus for future exploration, second only to Mars as a target of interest within the solar system. The reason, of course, is the possibility that a liquid water ocean may exist beneath its surface, with obvious implications for the search for life. Three mission concepts for Europa exploration have been studies: an orbiter mission that can resolve the question of whether the subsurface ocean exists or not, followed by a lander mission, and ultimately, a cryobot/hydrobot mission. The orbiter and lander would have the additional challenge of survivability in the intense radiation environment at Europa, deeply embedded in the Jovian magnetosphere. If the Europan ocean does indeed exist, the cryobot/hydrobot mission concept involves melting through the ice surface of Europa and releasing an underwater submersible to reach and explore the ocean floor, looking for signs of life. The submersible would require high degrees of autonomy to perform its mission successfully, including onboard algorithms embodying knowledge of possible biosignatures. See Figure 3.

Figure 3. Other Future Planetary Missions: A Titan Explorer, A Europan Submersible

1.4 Astrophysics and Autonomy

Looking beyond the solar system, NASA has a series of next-generation deep sky observing missions planned in its Origins Program, whose end goal is the capability to image Earth-like planets around nearby stars, even to resolve features and perform spectroscopic investigations of such planets. The hallmark mission in this series is known as the Terrestrial Planet Finder, a deep-space-based interferometer consisting of multiple elements. These elements are guided at unprecedented precision to first null the light, via interference effects, coming from the primary star in these distant stellar systems, then collect the precious photons coming from any planetary companions it may possess. The autonomy challenge in this mission involves guidance and control of the interferometer itself during observations, and detection of and compensation for faults and performance degradations in the elements such that the collective capability of the multiple-platform interferometer is maintained. See Figure 4.

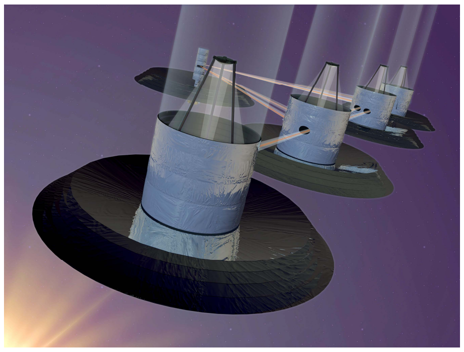


Figure 4. A Future Astrophysics Mission: Terrestrial Planet Finder

2. THE EMERGENCE OF AUTONOMY

Intelligent, highly autonomous space platforms will evolve and deploy in multiple phases. The first phase involves automation of the basic engineering and mission accomplishment functions of the space platform. The relevant capabilities include mission planning and resource management, health management and fault protection, and guidance, navigation and control. Stated differently, these autonomous capabilities will make the space platform *self-commanding* and *self-preserving*. Some of the relevant technologies include Artificial Intelligence (AI)-based planning & scheduling, model-based reasoning, and intelligent agents. In this initial *engineering-directed autonomy* phase, NASA space platforms will achieve onboard automated closed loop control among: planning activities to achieve mission goals, navigating, maneuvering, and deploying instruments and sensors to execute those activities, and detecting and resolving faults to continue the mission without requiring ground support. Also in this phase, the first elements of *science-directed autonomy* will appear. However, the decision-making capacity to determine how mission priorities should change and what new mission goals should be added in the light of intermediate results, discoveries and other events would still reside with scientists and other analysts on the ground.

Work on automating the spacecraft will continue into challenging areas like greater onboard adaptability in responding to events, closed-loop control for small body rendezvous and landing missions, and operation of the multiple free-flying elements of space-based telescopes and interferometers. In the next phase of autonomy development and deployment, a portion of the scientist's awareness will begin to move onboard, i.e., an observing and discovery presence. Knowledge for discriminating and determining what information is important would begin to migrate to the space platform. The relevant capabilities include feature detection and tracking, object recognition, and exploratory data sampling. At this point, the space platform begins to become *self-directing*, and can respond to greater uncertainty within the remote operating context. Ultimately, a significant portion of the information routinely returned from platforms would not simply and strictly match features of stated prior interest, but would be deemed by the onboard software to be "interesting" and worthy of further examination by appropriate science experts on the ground. At this point, limited communications bandwidth would then be utilized in an extremely efficient fashion, and "alerts" from various and far-flung platforms would be anticipated with great interest.

For surveys of NASA autonomy technology activities, see [1,2].

2.1 The Remote Agent

The most notable and successful effort in spacecraft autonomy development at NASA to date has been the Remote Agent, a joint technology development project by NASA Ames Research Center and the Jet

Propulsion Laboratory (JPL) [3]. The Remote Agent Experiment was conducted on the New Millennium Deep Space One (DS1) mission in May 1999, whose primary goal was to flight validate new technologies.

The Remote Agent consists of a Smart Executive [4], a Planning and Scheduling module [5], and a Mode Identification and Reconfiguration (MIR) module [6]. The onboard system receives mission goals as input, which is translated to a set of spacecraft activities free of resource and constraint violations by the Planner/Scheduler. The Smart Executive provides robust, event-driven execution, with runtime monitoring and decision-making. MIR continuously monitors representations of sensor data, identifying current spacecraft modes or states, and when these are fault modes, selects recovery actions. Other functions such as guidance, navigation and control, power management, and science data processing are domain-specific functions that can be layered on top of this basic autonomy architecture, and are developed or modified for each new mission. The Remote Agent was designed to serve as a general spacecraft autonomy architecture.

The demonstration objectives of the Remote Agent Experiment (RAX) on DS1 included nominal operations with goal-oriented commanding, closed-loop plan execution, onboard failure diagnosis and recovery, onboard planning following unrecoverable failures, and system-level fault protection. All of the technology validation objectives for RAX were accomplished. Additional details may be found in [7]. The Remote Agent was a co-winner of the NASA Software of the Year Award in 1999.

2.2 Some Definitions

There is often confusion on the differences between automation and autonomy. Acknowledging that the definitions given here may not be regarded as the final word on the subject, we make the following distinctions:

Automation applies to the creation of functionality (typically via algorithms), which can be fully defined independent of the context in which the functionality will be deployed, or when the context (e.g., the remote environment) can be modeled with sufficient confidence that the required functionality is well understood.

Autonomy, on the other hand, applies to the creation of functionality (typically via reasoning or inference capability), which is designed to be effective when context is important, and when the ability to model context (again, e.g., the remote environment) is limited. Autonomy specifically includes the capability to assess context and to support decisions based on knowledge that will only be available when the functionality is accessed, not when it is created. Knowledge and importance of context is the key consideration for distinguishing the need for automation vs. autonomy.

Handling of context is the central difference, but there is also a way in which automation and autonomy are similar: the functionality being created, whether based in algorithms or reasoning, is in both approaches fixed at deployment time (typically, launch time).

In the remainder of this section, we examine the research and technology investment areas that are contributing to the creation of autonomy capabilities for NASA missions. All of the work described is sponsored either by NASA’s Intelligent Systems Program or NASA’s Mars Technology Program.

2.3 Planning and Execution

Automated planning, scheduling, resource management and execution form, arguably, the core of system-level autonomy. These capabilities, when integrated, provide the basis for a space platform to perform engineering functions in closed-loop fashion onboard.

Planning involves reasoning about activities that will accomplish a desired set of goals. Automated planners work with models of tasks, resources and constraints, determining a set of activities which when executed, moves the space platform from its current state to a state where the desired goals have been achieved. In addition to the basic task of planning (determining the relevant tasks and their ordering), the planning system must also determine the absolute or relative timing of the tasks (scheduling), ensure that no relevant resources are oversubscribed (resource management) and that no flight constraints are violated. The mission plan provided by the planning system is then passed to an execution system, or executive. The executive issues the specific commands contained in the plan, and monitors their execution. Execution violations can be handled in several ways: the executive typically has mechanisms to accomplish local recoveries, global recoveries are the domain of system-level fault protection, and finally, the planning system itself can generate a new plan when it is determined that assumptions in the plan have been violated, e.g., the conditions of the operating environment, or the availability of a resource or capability.

The planner from the Remote Agent, known now as EUROPA, has continued to mature. Other topics, relating to the interaction of planning and execution, are also under investigation [8]. There is work on contingent planning, which is examining the partial expansion of alternate paths of execution in a mission plan [9]. The idea is to anticipate the kinds of failures that may occur during plan execution, and to predetermine recovery actions for those failures, thereby providing both for greater robustness of execution in the plan, and obviating the need to engage full-up, computationally expensive replanning by default when execution failures occur. Another benefit is that ground personnel have the opportunity to validate alternate execution pathways before they may be engaged.

Another successful planning system is ASPEN/CASPER [10]. This planner is based on a continuous planning model where a plan is always being modified, either to optimize it further, or to incorporate the latest information available. This process can begin from a description of an initial state. ASPEN has been deployed on several applications, notably for the Antarctic Mapping Mission, where it achieved an order-of-magnitude reduction in the time required to create schedules for downlink activities, as well as supported "what-if" negotiations between the science and operations teams. CASPER is the high-performance version of ASPEN designed for onboard use, which can achieve plan turnarounds in the tens of seconds.

2.4 Science Data Understanding

Scientists are naturally skeptical of an autonomous capability that purports to be able to interpret raw science data in an informed fashion. The key to progress in this area is working closely with scientists to properly define the scope of any such onboard capability so that it is realistic and value-added. As described above, scientists associated with the Mars Smart Lander mission are interested in a capability to collect science data during traverses and to alert scientists on the ground when certain pre-defined phenomena have been detected.

Science-directed autonomy will typically rely on technologies for pattern and object recognition and machine learning techniques. The system known as QuakeFinder was successful in the use of a change detection capability [11]. QuakeFinder applies a registration technique to before and after images. Rather than insisting on global registration, the technique instead focuses on successful registration of local regions in the images, and as a side effect detects change along the boundaries of those regions. The technique is particularly suited to the detection of linear displacements. QuakeFinder has been successfully applied to the detection of ground movement due to earthquakes, detecting such motion at 1/10 pixel resolution. This system is now being applied to Voyager and Galileo images of Europa, looking for evidence of surface changes that may be consistent with the existence of a subsurface ocean.

Object recognition techniques previously applied successfully to the automated detection of volcanic features in the Magellan SAR image set of Venus are now being applied to the more general problem of crater detection on planetary bodies [12]. The crater detection problem is harder because craters may overlap, be in various stages of degradation, and be observed in diverse lighting conditions. Nonetheless, this crater detection capability has been successfully trained on Mars images and tested on lunar images. Putting this kind of capability onboard would be value-added from a planetary scientist's viewpoint: crater studies are more about determining the number and distribution of craters on a planetary surface, rather than examining images of each individual crater.

Another activity is examining the interaction of onboard science data analysis with mission planning and execution for a Mars rover [13]. The data analysis capability here is a combination of texture analysis and spectral analysis looking to detect mineralogical signatures. The system summarizes and prioritizes science targets based on data collected and analyzed during traverse activities, and generates an update to the mission plan. This new plan can be communicated to scientists on the ground for modification and approval or executed autonomously.

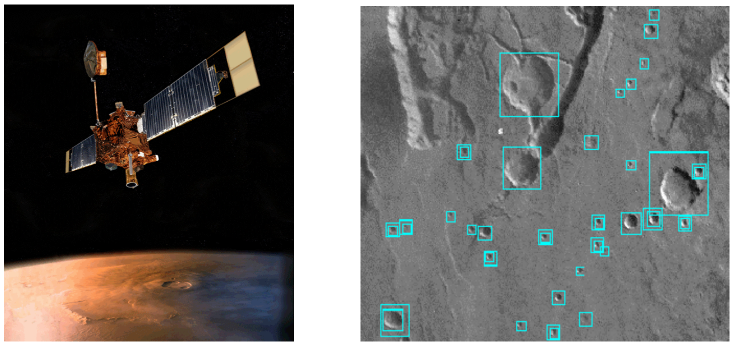


Figure 5. Craters Detected in Viking Data

2.5 Safe and Precise Landing

Many future mission concepts call for the ability to land space platforms on planetary bodies, ranging from small and hard-to-model asteroids and comets, to larger targets with more predictable environments, such as Europa and Mars. Several machine vision-based techniques have matured sufficiently to be the basis for safe and precise landing capabilities [14].

For example, spacecraft motion can be estimated accurately by precisely tracking features during descent. These image-based techniques can estimate motion to 1% accuracy of the total distance traveled. Feature tracking also can be used to accurately estimate position relative to defined landmarks, such as the centers of craters. These feature-based navigation techniques use different methods than, e.g., the crater recognition algorithms described earlier. When the objective is navigation, the task is to select features that can be reliably tracked under different viewing angles and lighting conditions. Features of scientific interest, however, are selected by different criteria, and may or may not line up with features that are useful to support navigation.

The stereo techniques that support surface navigation for rovers can provide critical information during the late stages of descent, when elevation knowledge of the topography directly below can be the key to final guidance to a safe landing spot. These techniques can compute elevation information to greater than one part in a hundred accuracy relative to altitude.

These vision-based techniques are being combined into an integrated hazard detection and avoidance capability for missions such as Mars Smart Lander, to locate and guide the landing spacecraft to a safe zone with local surface roughness of less than ten centimeters and local slopes of no greater than ten degrees. Path planning algorithms also can guarantee that there is accessibility from the landing site to the pre-defined science investigation sites. Benchmarking indicates that all of the required computation can be achieved in real-time and near real-time during the critical and intense entry, descent and landing sequence.

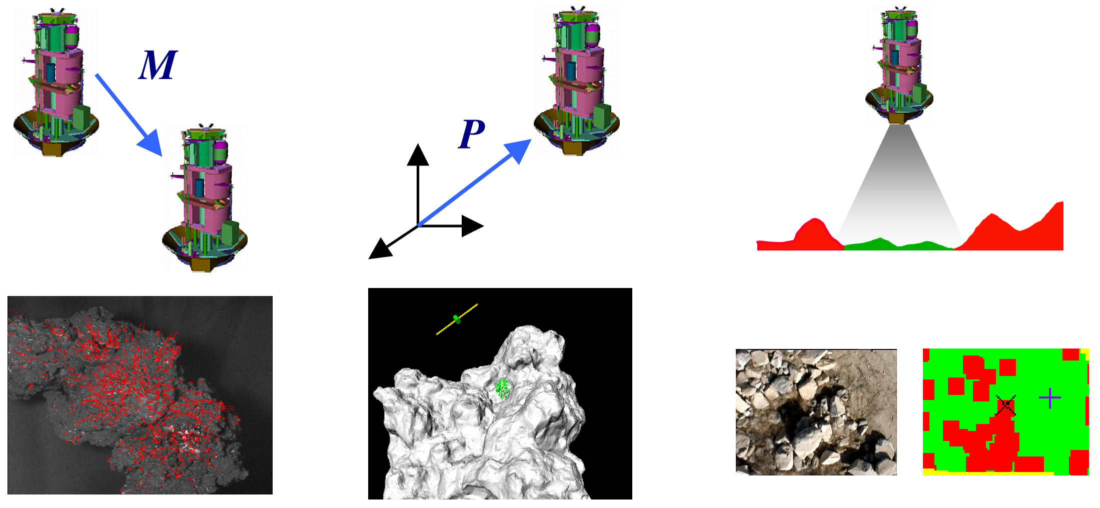


Figure 6. Vision-based Motion Estimation, Position Estimation, Hazard Detection and Avoidance

2.6 Mobility

Mobility is a defining feature of planetary rovers. Future rover missions are being planned now around the ability to go further and through rougher terrain than missions to date have accomplished. The four salient questions of mobility are: "Where am I?", "What is the environment surrounding me?", "Where should I be?", and "How do I get there?" . These are the challenges of, respectively, position estimation, terrain estimation and obstacle detection, goal selection and tracking, and trajectory generation and path planning [15].

Addressing the challenge of position estimation has proven to be most difficult in the barren, rough, natural terrain known to exist on Mars. While current capabilities are able to provide error bounds down to 3% of the distance traveled, newer techniques promise to reduce that margin by an order of magnitude. These techniques include visual odometry for tracking the motion of features in navigation imagery to independently measure displacement of the vehicle, and visual servoing for tracking the motion of a selected visual objective to ensure motion toward that objective. Sensor fusion can play an important role to merge multiple measurement sources through statistical and logical filters, improving an overall position estimate. Related terrain estimation techniques address this challenge by estimating soil properties through visual and contact methods to determine sinkage and slippage of the vehicle. Other techniques match displacement of terrain map features on an intermittent basis to enhance estimates of vehicle position.

Obstacle detection via stereo processing is by now a standard technique, but is limited currently to providing surface elevation estimates only in the immediate vicinity of the rover. Continuing work in this area includes elevation map seaming to merge elevation maps provided by separate sets of stereo image data, and wide baseline stereo for the correlation of imagery taken from separate locations, providing elevation maps out to much greater ranges than is currently possible. Also, multi-resolution mapping techniques correlate surface and orbital imagery.

Mission operators currently perform the tasks of goal selection and tracking. Moving this function onboard the vehicle will require maturation of the position and terrain estimation techniques described above along with maturation of onboard techniques for planning and execution.

Finally, the challenge of trajectory generation and path planning concerns methods for navigating from the current location to the goal location, given the terrain. At this time, it appears that the techniques to be utilized on the MER mission will be sufficient for missions like MSL.

2.7 Distributed Autonomy

Mission concepts involving multiple space platforms are becoming increasingly common. The first examples of so-called spacecraft constellations will likely be in Earth orbit, where fleets of satellites carrying different sensors and instruments may coordinate observations and responses to provide global coverage for the detection of events such as volcanic eruptions or forest fires. Some future space-based observatories will consist of multiple elements, carefully arrayed and maintained to create apertures of unprecedented scale. Finally, heterogeneous assets deployed at Mars will collectively perform science missions, as well as count as the first true example of off-planet infrastructure.

The engineering capabilities of a single space platform do not scale simply to appear as a coordinated capability across multiple platforms. This is most apparent for mission planning, considered by many to be the core capability of system-level autonomy. An important consideration is architecture. Three different architectures for distributed planning are under study [16].

Perhaps the most straightforward is a centralized architecture, where a specially designated space platform performs the planning for all platforms. This is conceptually indistinguishable from mission planning on a single platform but has distinct disadvantages: intense communications to provide timely relevant data from all of the platforms to the central platform, and the need to update the models used by the central planner whenever another platform is added to the configuration.

In a distributed architecture, a central planner only allocates goals to the separate platforms, each of which has a planner of its own to autonomously achieve the goals assigned to them. Less communications are required in this architecture, but planners do appear everywhere, and some models used by the planners still need to be shared, so that the central planner can make informed choices about assigning goals. Whenever another platform is added to the configuration, the central planner requires a minimal model of that platform's capabilities, so that it can continue to assign goals appropriately.

In a market-based architecture, each platform has a model of its own capabilities, and bids to accomplish goals posted in auctions managed by a central platform. This architecture is the most scalable, because no platform requires models of the capabilities of other platforms, which can be added to the configuration at any time. There is global computational overhead however, for all planners examine all goals.

The first example of distributed autonomy in flight will occur in 2004, when the Autonomous Sciencecraft Constellation, a technology experiment on the New Millennium ST-6 mission, will test an integrated capability for onboard mission planning and onboard science data analysis aboard the Air Force's three-satellite constellation known as Techsat-21 [17].

2.8 The Role of Software Architecture

The Remote Agent technology experiment on DS-1 made clear the central role that software engineering must play in deploying autonomy capabilities in flight. In particular, it is highly desirable that the software architecture used in flight provides direct support for autonomy capabilities. The Mission Data System (MDS) is a flight/ground/test software architecture conceived and designed to be such an autonomy-friendly architecture [18]. In this goal- and state-based architecture, goals are defined to be constraints on the values of state variables over specified time intervals. Estimators interpret measurement and command evidence to estimate state. State variables hold state values, annotated with estimates of uncertainty, critical for informed, autonomous decision-making. Controllers issue commands, striving to achieve goals. Explicit models express specific relations among states, commands and measurements. Goals and states are the atomic concepts of autonomy, and MDS thereby provides a head start on implementing autonomy capabilities.

The MDS approach offers other important benefits, such as providing a common language of discourse for systems engineers and software engineers, who otherwise interact imperfectly at best across a pile of textual requirements. MDS also utilizes a components-based software architecture, and designs out certain classes of software defects at the architectural level, such as certain race conditions, along with units or dimensionality errors.

The Mars Smart Lander mission has adopted the MDS software architecture for these reasons, including its system engineering approach of state analysis.

3. SUMMARY: LOOKING FARTHER INTO THE FUTURE

The preceding on the emergence of autonomy included descriptions of the initial phases of development and deployment of autonomy capabilities for space platforms: Engineering-directed autonomy will place many traditional spacecraft functions such as mission planning, execution, resource management, guidance and control, navigation and fault protection onboard in closed-loop fashion. Science-directed autonomy will allow scientifically interesting phenomena to be detected via onboard capabilities, supported by onboard mission replanning.

Beyond these initial phases, we can project a phase where space platforms become the analogues of web nodes, with direct interaction enabled among space platforms, the science community, and the general public. Interested users may “register” with autonomous spacecraft to learn about breaking results.

The next phase may involve self-organizing constellations of space platforms consisting of heterogeneous assets performing joint, coordinated execution of mission objectives, with self-calibration and adaptation enabled.

A phase beyond that may be characterized by long-term survivability of ten years or more, even with zero ground support, achieved by onboard contingency handling and self-repair, including functional redundancy achieved via software, particularly planning capability.

We’re not done yet. The next phase may involve onboard reprogramming and discovery, with the spacecraft authorized to modify science objectives and the mission plan based on onboard learning and discovery capabilities.

And in a final phase, with spacecraft and mission evolution enabled, space platforms may not only repair themselves, but may also refine and improve their functionality, as a more efficient form of mission redesign and reprogramming.

As with any vision, the most remote projections get a bit harder to track and to immediately credit. But we can be confident that autonomy is here to stay as a central capability for achieving future NASA missions. Autonomy is multidisciplinary in nature and must be the product of the inputs of computer scientists, spacecraft engineers, mission designers, ground system engineers, mission operators, scientists, software engineers and systems engineers.

REFERENCES

[1] Richard J.Doyle, “Spacecraft Autonomy and the Missions of Exploration,” Guest Editor’s Introduction, Special Issue on Autonomous Space Vehicles, *IEEE Intelligent Systems*, September/October 1998.

[2] Daniel E. Cooke and Butler Hine, “Virtual Collaborations with the Real: NASA’s New Era in Space Exploration,” *IEEE Intelligent Systems*, March/April 2002.

[3] Douglas Bernard, Gregory A. Dorais et al, “Design of the Remote Agent Experiment for Spacecraft Autonomy”, in *Proceedings of the IEEE Aeronautics Conference*, Aspen, CO, March 1998.

[4] Barney Pell, Erann Gat et al, “Plan Execution for Autonomous Spacecraft,” *15th International Joint Conference on Artificial Intelligence*, Nagoya, Japan, August 1997.

[5] Nicola Muscettola, Benjamin D. Smith et al, “On-board Planning for the New Millennium Deep Space One Spacecraft,” *Proceedings of the 1997 IEEE Aerospace Conference*, Aspen, CO, February 1997.

[6] Brian C. Williams and Pandu Nayak, “A Model-based Approach to Reactive Self-Configuring Systems,” *13th National Conference on Artificial Intelligence*, Portland, OR, August 1996.

[7] Douglas C. Bernard, Gregory A. Dorais et al, “Spacecraft Autonomy Flight Experience: The DS1 Remote Agent Experiment,” *Proceedings of the AIAA Space Technology Conference and Exposition*, Albuquerque, NM. September 1999.

[8] Nicola Muscettola, Gregory A. Dorais, et al, “IDEA: Unifying Planning and Execution for Real Time Autonomy,” *Proceedings of the i-SAIRAS 2001*, Montreal, Canada, June 2001.

[9] John L. Bresina, and Richard Washington, “ Robustness via Run-time Adaptation of Contingent Plans” *Proceedings of the AAAI Spring Symposium on Robust Autonomy*, Stanford, CA, March 2001

[10] Russell Knight, Gregg Rabideau, et al, “CASPER: Space Exploration Through Continuous Planning,” *IEEE Intelligent Systems*, September/October 2001.

[11] Paul Stolorz and Peter Cheeseman, “Onboard Science Data Analysis: Opportunities, Benefits, and Effects on Mission Design,” Special Issue on Autonomous Space Vehicles, *IEEE Intelligent Systems*, September/October 1998.

[12] Michael C. Burl, Tim Stough et al, “Automated Detection of Craters and Other Geological Features,” *Proceedings of the i-SAIRAS 2001*, Montreal, Canada, June 2001.

[13] Tara A. Estlin, Tobias P. Mann et al, “An Integrated System for Multi-Rover Scientific Exploration,” *Proceedings of 16th National Conference on Artificial Intelligence*, Orlando, FL, July 1999.

[14] Yang Cheng, Andrew Johnson, et al, “Passive Imaging-based Hazard Avoidance for Spacecraft Safe Landing,” *Proceedings of the i-SAIRAS 2001*, Montreal, Canada, June 2001.

[15] James Cutts, Samad Hayati et al, “The Mars Technology Program,” *Proceedings of the i-SAIRAS 2001*, Montreal, Canada, June 2001.

[16] A. Barrett, G. Rabideau et al, “Coordinated Continual Planning Methods for Cooperating Rovers,” *Proceedings of the IEEE Aerospace Conference*, Big Sky, MT, March 2001.

[17] Steve Chien, Rob Sherwood et al, “The Techsat-21 Autonomous Sciencecraft Constellation,” *Proceedings of the i-SAIRAS 2001*, Montreal, Canada, June 2001.

[18] Daniel Dvorak, Robert Rasmussen and Thomas Starbird, “State Knowledge Representation in the Mission Data System,” *Proceedings of the IEEE Aerospace Conference*, Big Sky, MT, March 2002.

A Multi-Agent Approach for Complex System Design

Irène Degirmenciyan-Cartault
Dassault Aviation, 78, quai Marcel Dassault Cedex 300 92552 St Cloud Cedex France
irene.degirmenciyan@dassault-aviation.fr

Abstract: Multi-agent systems that arose from research in Distributed Artificial Intelligence are now considered as a new paradigm to design and model complex systems. The design of complex systems seems to be more intuitive with cognitive agents because the designer works at a high level of abstraction where the agent is an important granularity entity with aspects of calculation, reasoning and control which make it quasi autonomous. After a brief overview of the required notions to understand the multi-agent systems' domain, we describe the JACK agent-oriented environment on top of which we are developing the SCALA environment that provides a multiagent-based methodology and tool for the design of complex systems. In SCALA, the global behaviour (*i.e.* its goal) of the system is modelled through a functional approach based on the definition of a graph of dependencies between the basic behaviours (*i.e.* tasks to accomplish to achieve a goal). The definition of this graph provides the necessary knowledge to manage cooperation between the agents and to plan reactively their activities when new events occur and when they have to reorganise themselves.

1. Introduction

Multi-agent systems that arose from research in Distributed Artificial Intelligence are now considered as a new paradigm to design and model complex systems. Those systems are used for several well-known reasons: their ability to run in unpredictable or less predictable environments, to react to unknown events, to reason, to cooperate, to learn and to be pro-active. The design of complex systems seems to be more intuitive with cognitive agents because the designer works at a high level of abstraction where the agent is an important granularity entity with aspects of calculation, reasoning and control which make it quasi autonomous. Then the distribution of the control on the agents gives the system more robustness, efficiency and a better reactivity thanks to sophisticated coordination mechanisms. Agent-oriented programming addresses the need for software systems to exhibit rational, human-like behaviour in their respective problem domains. Traditional software systems make it difficult to model rational behaviour, and often programs written in these systems experience limitations, especially when attempting to operate in real time environment.

The aim of this paper is not to present the entire domain of research on multi-agent systems, but to point out characteristics that seem to be relevant for the design of complex systems. So, in section 2 we overview the required major notions to understand the multi-agent systems' domain. In section 3 we describe an example of the JACK agent-oriented environment on top of which we develop the SCALA project that provides a multiagent-based methodology and a tool for the design of complex system (section 4). Then, we expose in section 5 the perspectives of this work to extend it to time constraints.

2. Multi-agent Systems

In this part, we introduce the notion of **agent** and the advantages to adopt such a technology of programming.

2.1. Definitions

Although there is no consensus on the agent definition, we can assume that an agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its designed objective [WOO 95].

The notion of autonomy means that agents have control both over their own internal state, and over their behaviour. They act without the intervention of humans or other systems.
To define the notion of **intelligent agent**, the notion of flexibility is added. Intelligent agents can be characterised by their ability to carry on flexible autonomous action [JEN 98], *i.e.* they present:

- Reactivity: intelligent agents are able to perceive their environment, and respond in a timely fashion to changes that occur in it to satisfy their design objectives;
- Pro-activeness: intelligent agents are able to exhibit goal-directed behaviour by taking the initiative in order to satisfy their design objectives;
- Social ability: are capable of interacting with other agents (and possibly humans) in order to satisfy their design objectives.

In a dynamic and uncertain environment, these characteristics are not easy to obtain. The agents have to continually respond to their environment to reactively take into account its changes, while maintaining their goals. The difficulty for an agent is to deal simultaneously with these two aspects. The agents thus have to attempt to achieve their goals, but not in a blindly fashion. They continually have to perceive the environment and be ready to quickly react to new situations. The carried on goal can be questioned, or some changes in the environment can impact the way to achieve it, etc.
The last important point is social ability. Agents have to negotiate or cooperate with others to achieve shared goal. This social ability includes the possibility to define organisational structure where agents hold roles and obey to social rules. The multi-agent approach specially focuses on these aspects.

A **multi-agent system** is a distributed system composed by several agents interacting with each other's. A multi-agent can be characterised by the following points as reminded in [BRIOT 01]:

- Each agent has limited information or capabilities;
- There is no global control of the multi-agent system;
- The data are decentralised;
- The calculation is asynchronous.

Multi-agent systems have the traditional advantages of the distributed and concurrent problem solving, such as modularity, computation time (parallelism) and reliability (due to the redundancy of resources and capabilities). Multi-agent systems are also interested in developing sophisticated interaction mechanisms, such as cooperation, coordination and negotiation.

2.2. Agents versus objects

When one considers only the relative properties of agents and objects, one can have difficulties to see the added value of agents. Indeed objects are computational entities that encapsulate some states, are able to perform actions (methods), and communicate with other objects through message passing. Yet there are significant differences between agents and objects [WOO 99]:

The principle of encapsulation is the idea that objects can have control over their own states, *i.e.* they exhibit a certain autonomy over their states. But an object does not exhibit control over its behaviour. A method *m* of an object is executed by this object when another object invokes it. The object that performs the action (method *m*) has no control over this invocation. The decision lies with the object that invokes the method. In the agent case, we do not think of invocation, but in requesting actions. The decision lies with the agent that receives the request. The agents have the control over the decision to execute an action or not.

Although one can integrate flexible (reactive, pro-active, social) autonomous behaviour in object-oriented programs, this kind of behaviour is not inherent to the object philosophy, and this means to implement agents' behaviour in an object-oriented language.

Another important distinction is that agents have each their own thread of control, whereas there is only one in an object-oriented program. Even if we take into account the concept of concurrency that arrives in object-oriented languages, such as the multi-threading in Java, it does not capture the idea of autonomous entities. A multi-agent system is inherently multi-threaded.

3. JACK: An agent infrastructure [HOW 01]

A host of environments support the development of multi-agent applications each presenting different characteristics. An effort of standardisation is led by the FIPA (Foundation for Intelligent and Physical Agent).

In this section, we are describing one of them, the agent-oriented language JACK software, on which lies our work presented in section 4. JACK is an agent-oriented development environment build on top of Java. It addresses the need for software systems to exhibit rational, human-like behaviour in their respective problem domains. The agents used in JACK are intelligent agents. They model reasoning behaviour according to the theoretical Belief Desire Intention (BDI) model of Artificial Intelligence developed by Rao and Georgeff [RAO 95]. More precisely, following the BDI model the JACK agents are autonomous entities that have goals to achieve thanks to events to which they are sensitive. To determine the way to achieve a goal, the agents have plans. Each plan describes the way the agent has to react to cope with a given situation. Consequently, an agent attempt to achieve a goal (its desire) using the relevant plan (its intention) that it will have determined in analysing its knowledge (its beliefs) on the external environment.

So, a JACK agent can exhibit reasoning behaviour under both pro-active (goal directed) and reactive (event driven) stimuli. Each agent has:

- A set of beliefs about the world;
- A set of events that it will respond to;
- A set of goals that it may arise to achieve;
- A set of plans that describe how it can handle the goals or events that may arise.

Let us now detail JACK specific classes [JAR 01]:

The Agent class

The Agent class embodies all the functionality associated with a JACK intelligent agent. It allows to define the behaviour of an agent, its capabilities, the type of messages and events to which it is sensitive and the plans it uses to achieve its goals. Each JACK agent is associated to an independent execution process (a Java thread).

In general, the definition of this class needs to include the following conceptual statements:

- *Knowledge Bases* which the agent can use and refer to
- *Events* (both internal and external) that the agent is prepared to handle
- *Plans* that the agent can execute
- *Events* the agent can post **internally** (to be handled by other plans)
- *Events* the agent can send **externally** to other agents.

The Database Class

The *Database* class implements the main data storage device that agents use. Each database class describes a set of *beliefs* that the agent can have. It represents these beliefs in a *first order*, *tuple-based relational* database system. The logical consistency of the belief this database contains is automatically maintained. Hence, for example, if an agent adds a belief that contradicts a belief it already has, the database class detects this and automatically ejects the old belief. The database class is not the only way that an agent can represent information. Agents can also include ordinary members and other data storage structures that have been implemented in Java.

The Event class

Events motivate an agent to take action. There are a number of event types in JACK, each with different uses. These different event types help model.

- Internal stimuli; essentially events that an agent sends to itself. These internal events are integral to the ongoing execution of an agent and the reasoning that it undertakes.
- External stimuli, such as messages from other agents, or percepts that an agents receives from its environment.
- *Motivations* that the agent may have, such as goals that the agent is committed to achieving.

Events are the origin of all activity within an agent-oriented system. Whenever an event occurs, an agent initiates a task to handle it. This task can be thought of as a thread of activity within the agent. This task causes the agent to choose between the plans it has available, executing a selected plan or plan set (depending on the event processing model chosen) until it succeeds or fails. If plan execution succeeds, then the event that initiated it is said to have succeeded. If plan execution fails, on the other hand, there are two options. Under normal event handling the event is said to have failed after the first instance of plan failure. Under *BDI* event handling, a number of plans can be selected for execution and these are attempted in turn, in order to try to achieve successful plan execution. If the set of chosen plans is exhausted, then the event is said to have failed. There are a number of event classes in the JACK Agent Language, each representing different types of motivation to act.

One of the important aspects of the BDI reasoning model at a conceptual level is that it models *goal-directed behaviour* in agents, rather than plan-directed behaviour. That is, an agent *commits* to the desired outcome, not the method chosen to achieve it. When using the BDI reasoning model, an agent does not simply react to incoming information, but sets itself a goal that it then tries to achieve. Rather than distracting an agent from its goal, incoming events are added to an agents knowledge base and can then subtly influence its behaviour.

The key difference between normal events and BDI events is how an agent selects plans for execution. With normal events, the agent selects the first applicable plan instance for a given event and executes that plan instance only. The handling of BDI events is more complex and powerful. An agent can assemble a plan set for a given event, apply sophisticated heuristics for plan choice and act intelligently on plan failure. At least one of the following characteristics applies to each type of BDI event under the BDI model:

- *Meta-level reasoning* : it allows precise control over how an agent chooses a plan for execution from the set of applicable plans. Whenever there is more than one applicable plan instance for a given BDI event, a *special* event is posted within the agent. By choosing to handle this event an agent can implement meta-level reasoning. If the meta-level reasoning plan fails and does not select a plan for execution, then the default plan selection method is invoked.
- *Reconsidering alternative plans on plan failure* : if a course of action (plan) fails, an agent can try a number of other courses of action by attempting any number of applicable plans to achieve the goal that has been set.
- *Re-calculating the applicable plan set* : after that the previous selected plan has failed, an agent may select an alternative plan: it either keeps track of the plan instances that were initially applicable and select another member of this set; or it re-computes which plan instances are applicable and select one from the new set, excluding plan instances that have already failed.

Additionally it is possible to further control BDI behaviour by setting *behaviour attributes*.

The Plan class

The Plan class describes a sequence of actions that an agent can take when an event occurs. Whenever an event is posted and the agent adopts a task to handle it, the first thing that the agent does is to try to find a plan to handle the event. Plans are similar to methods and functions from more conventional programming languages. Each plan is capable of handling a single event. An agent may further discriminate between plans that declare they handle an event by determining whether a plan is *relevant* by using specific JACK methods that can be assimilated to a filter.

The Capability Class

The *capability* concept is a means of structuring reasoning elements of agents into clusters that implement selected reasoning capabilities. This technique simplifies agent system design, allows code reuse and encapsulation of agent functionality. Capabilities represent functional aspects of an agent that can be plugged in as required. This *capability as component* approach allows an agent system architect to build up a library of capabilities over time. These components can then be used to add selected functionality to an agent. Events, databases, plans, Java code and other capabilities can all be combined to make a capability.

4. SCALA: A methodology and a tool to design complex systems

At Dassault Aviation, we work on a project named SCALA (Cooperative System of Software Autonomous Agent) developed on top of the JACK agent-oriented software presented in section 3. SCALA aims at showing the interest of the multi-agent approach for modelling and designing complex systems, where several entities have to cooperate to achieve joint goals. SCALA provides a tool and a methodology that enable the designer to build at a high-level of abstraction the behaviour of a multi-agent systems.

The global behaviour of the system (*i.e.* its goals) is modelled through a functional approach based on the definition of the graph of dependencies between basic behaviours, *i.e.* the tasks to be accomplished by the agents of the system.

4.1. The objectives

The major objectives carried out in the SCALA project are:

- To provide a tool to prototype multi-agent systems (MAS) in proposing a methodology of design based on the functional requirements;
- To make easier the modelling and the design of such systems thanks to a high level abstraction language;
- To simulate different types of organisation and communication protocols between agents;
- To constitute and provide libraries of reusable mechanisms and protocols;
- To propose tools to support the design of the system and to monitor the behaviour of the system (individual and collective behaviour).

In next section, we are focusing on the methodology proposed in SCALA to assist the design of complex systems.

4.2. A methodology to develop complex systems

This methodology is based on a functional approach enabling designers to elaborate and model the whole behaviour of the multi-agent system. This methodology is composed of 8 steps:

1. Definition of the graph of functional dependencies;
2. Definition of the sub-graphs (and the goals associated);
3. Identification of the expected relevant events;
4. Description of the elementary behaviours or tasks to be accomplished by the agents;
5. Definition of the agents;
6. Definition of the group of agents;
7. Choice of the social organisations;
8. Choice of the cooperation protocols.

Now, we are going to detail the different points of this methodology and briefly describe the functioning of SCALA through a given type of application: the simulation of air combat missions. This presentation allows us to point out several limitations that we are attempting to overcome in the second part of the presented study.

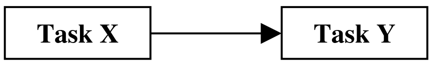
4.3. The graph of functional dependencies

The graph of dependencies enables the designer to model at a high level of abstraction the behaviour of the multi-agent system. This graph is constituted by one or more distinct graphs composed of tasks or basic behaviours that have to be accomplished by the agents of the system, and the constraints between them. The definition of the graph provides the necessary knowledge to manage the cooperation between the agents. Different types of the control either centralised or distributed, can be built depending on which agent(s) own(s) this knowledge. At this stage, tasks are not allocated yet to the agents. The assignment is made dynamically (see § 4.11) according to the current situation and the available resources (agents are assimilated to resources through their skills). Thus we give the agents a certain freedom of action that makes the system more reactive and allows to avoid some failures such as the use of non-available resources. We have a proscriptive approach that constrains the behaviours rather a directed one. For example, according to the constraints on a task, an agent can attempt to ask another agent for help or performs another task which does not requires the help of another agent .
The graph of dependencies is in fact, at a given level of abstraction, a decomposition of the global behaviour of the system, close to a decomposition of a problem into sub-tasks, and including the different possible alternatives for each sub-goal. In that way, the SCALA agents' behaviour is task-oriented.
Furthermore, we will see, in § 4.7 that each task is associated to several JACK plans, that specify the different methods to accomplish a same task. By default, *i.e.* without specific associated constraints, each task of the graph has to be achieved by a single agent.

Let us now detail the constraints between tasks. They are defined by links or connectors expressing notions of synchronism (at the beginning or at the end of several tasks), exclusion (the execution of one task inhibits others'), refinement (several methods can be invoked to execute a task) and abstraction (a task is composed of others). Another type of constraint is about the number of agents required to perform a same task.

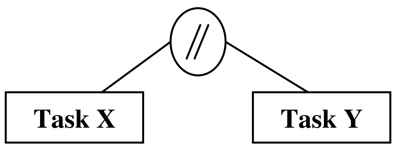
The link of precedence:

This link shows that Task Y cannot be executed before Task X. A task may have several links of precedence and may also be the predecessor of several tasks.

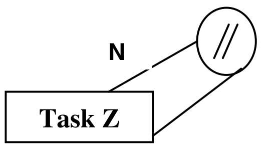


The connector of temporal synchronisation //:

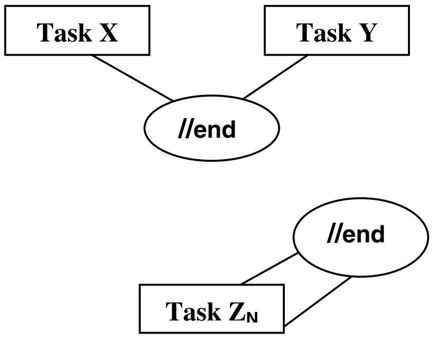
This connector implies that Task X and Task Y begin synchronously.



It is also possible to define that N agents have to execute the same task concurrently. It is the case for the following example where N agents have to begin Task Z synchronously.



The connector of temporal synchronisation //end:

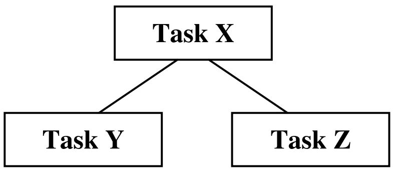


This connector implies a synchronisation of termination. Task X and Y must end at the same instant, or N agents have to end their task at the same instant.

Remark: To realise this connector, the graph must bring the notion of delays on tasks. But it could be applied easily in case of interruptible tasks.

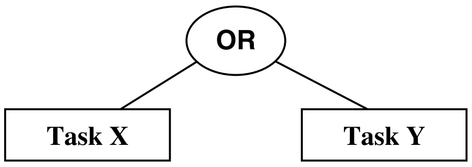
The link of decomposition:

The decomposition may be considered as a refinement.



Task X (abstracted Task) is composed of two sub-tasks Y and Z. The execution of Y and Z is necessary to validate Task X. The execution of Y and Z is possible only when the predecessors of X have been realised.

The connector OR:

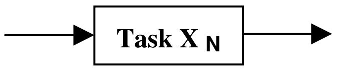


This connector represents the concept of exclusion between tasks. The execution of one of the tasks will prevent the execution of the other linked tasks.

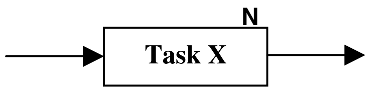
In this example, the activation of Task X will inhibit Task Y one. The choice of the branch is context dependant.

The connector T_N:

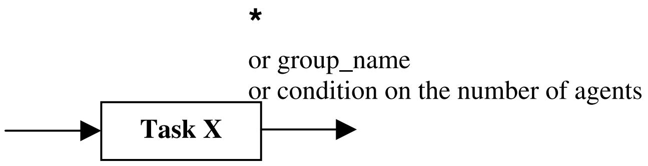
This connector implies that the task must be executed N times but with no constraints on the executors. So a single, or many agents may execute N times Task X.



Constraints of allocation:



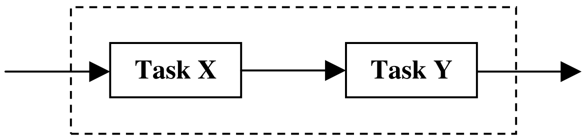
This connector means Task X must be done by N agents but with no constraints of synchronisation.



When all the agents of the group have to execute this task, the N is replaced by a star (*). This permits to leave the constraint on the number of agents.
But one can also precise if a group has to do this task: *group_name* or you can put conditions on the number of agent that has to do the task: *≥2 (this condition means that at least two agents of the group have to do the task).

Series:

A series of tasks must be executed by the same agent or the same group of agents.
In this example, task X and Y must be executed by a same agent.



Those all connectors can be merged to express more sophisticated constraints between tasks.

4.4. The definition of the sub-graphs (goals)

The sub-graphs represent partial graphs of the graph of functional dependencies. Each sub-graph is associated to a goal that replies to a particular event. For example, "**To intercept a threat**" is one of the possible goals to be achieved by the patrol if the event "**New contact on the radar**" occurs. This goal is associated to a sub-graph (a set of tasks and connectors) depicting the functional possible procedures the agents have to carry out to achieve it.

4.5. The expected relevant events

The agents of the system have sometimes to reorganise themselves to achieve goals. The goals are generally fulfilled by one agent or a group of agents. The agents take into account new events that can be interactively provided by the designer during the simulation (for example the on-line creation of a new threat) coming either from environment changes, or from messages between agents. A new event implies that the agents have to cope with a new sub-graph. All the difficulty lies in the management of the new goals. Each event is characterised by a degree of priority which allows the agent to select the most urgent to treat. The algorithm concerned is developed in fig. 6. To model the system, the designer specifies the different events to which the agents are sensitive. The designer has to specify the "event/goal" pairs.

Now, let us present our scenario to illustrate the methodology of SCALA and model the behaviour of the designed multi-agent system.

4.6. An interception scenario

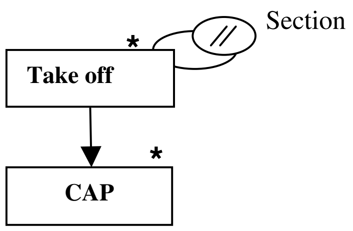


fig. 1: Sub-graph "CAP"

In the scenario, four aircraft fly as a division on a close-air support mission. A division is composed of two sections, each containing two aircraft. Close-air support missions involve substantial communication and coordination to ensure the success of the mission. We are interested to a high level of coordination, *i.e.* on the goal to achieve and actions to perform. Command and control are provided by an airborne radar plane (in the friend area) and a number of forward air controllers (AC), friend forces on the enemy ground. We are following this scenario over the steps of SCALA's methodology.

The division's pre-briefed goal is **"Bomb Target 1"**. The aircraft take off from the carrier and rendezvous at a pre-briefed location to join into formation. The rendezvous is necessary to coordinate group flight because we assume that only two planes (a section) can launch simultaneously. According to the formalism defined in SCALA, this can be modelled by the sub-graph (fig.1). The star (*) means that all the sections of the group defined by the designer have to take off. This implies no conditions on the number of aircraft involved in the mission (it is specified via the agents definition) but only the (functional) way they have to take off. This generic approach of specifying behaviour

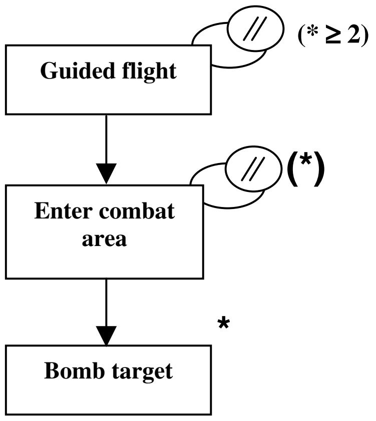


fig. 2: Sub-graph "Bomb Target"

ensure the re-usability of the graphs. The second task "**CAP**" (Control Air Patrol) expresses, in the same way, that all the planes have to join at a pre-briefed point.

While flying their route, the lead of the division checks in with the air controllers any changes in routing and the permission to enter the combat area. Once the AC verifies the mission, visually acquires the planes, and determines they can bomb the target without endangering friendly forces, the AC gives the final permission to drop their ordnance. Then, they exit the area and fly back on their egress route. The constraints (//) and (*** ≥ 2**) express that the task "**Guided flight**" has to be accomplished simultaneously by a group composed of at least two aircraft. This task is monitored by the airborne radar plane. All the agents of the related group must synchronously begin the task "Enter combat area". Then, all the agents, who carry a bomb, have to drop it (not necessarily at the same time). We do not have to precise on this graph that the last task is to return to the base because it is the task by default: it is a "home state" automatically executed. The global plan is the concatenation of these two sub-graphs (fig. 1 & fig. 2). In this step of the modelling, the designer has to define the "event/goal" pairs: "**Order to bomb the target / Bomb the target**". We further expose a complete scenario with new events occurring during the simulation.

The next step is to define the local behaviours, *i.e.* the different tasks.

4.7. Definition of the tasks

The tasks have some notable properties that we are describing here.

A set of methods

Each task is associated to a set of methods. A task is in fact a sub-goal that can be achieved by different manners. For instance, several tactics lead to the same result, but their use is context-dependent and an agent have to decide which of them apply according to its skills or to the availability of other agents in the case of a team-tactics. (*Remark*: this functionality is directly available in JACK with the meta reasoning of the agents based on sophisticated heuristics for the selection of the relevant plans).

Interruptible tasks

The tasks can be interruptible or not interruptible. The interruptible tasks have the capability to stop their execution and then to resume it or to definitely abandon it.
For example, a task can be interrupted when an event implies the interruption of the current method (same remark than below). In the related scenario, the "**Guided Flight**" task is interrupted to let the agents enter in the combat area when they arrive close to this zone and when they had received the permission from the AC. Furthermore, the execution of a task is submitted to constraints clustered in pre and post conditions.

Pre and Post conditions

Those notions are linked to the perception of the agent or its knowledge. The pre-conditions specify the conditions to execute a task but also the way. In fact, several manners can exist to achieve a task. Those pre-conditions can be assimilated to a filter for the choice of the relevant method.
The post-conditions describe the changes of the resources after the execution of the task. For example, if a missile is launched, the number of missiles must decrease in the knowledge base of the concerned agent.
According to our scenario, the pre-conditions on the task "**Enter combat area**" are to get the permission from the AC. For the task "**Bomb the target**", the planes have to wait for the permission and to be sure of the target to bomb.

Number of agents

It is the necessary number of agents required to execute the task. This information is very important to initiate cooperation between agents.

Skills and level of specialisation

Here are defined the necessary skills and level of specialisation required to execute the task. These characteristics also enable the cooperation between agents, while considered as complementary resources for the system.

4.8. A recursive groups definition

The groups in SCALA are defined via the following way:
- The type of the group,
- The members (type: agent or group, and number),
- The roles composing the group,
- Group Goal: the current global goal of the group,
- The sub-groups: the groups inside the group,
- The meta group: the group which it belongs,
- The type of organisation,
- The type of cooperation

- **name**: Group2,
- **type**: Section,
- **members**: Two Aircraft,
- **roles**: 1 Leader, 1 Wingman,
- **Group Goal**: To destroy the bandit
- **Sub-groups:** none,
- **Meta group**: Group 1,
- **Organisation**: Hierarchical,
- **Cooperation**: Delegation

- **name**: Group1,
- **type**: Division,
- **members**: 2 Sections,
- **roles**: 1 escort, 1 Bomber,
- **Sub-groups:** 2 Group2,
- **Group goal:** To destroy the bandit
- **Meta group**: none,
- **Organisation**: Hierarchical,
- **Cooperation**: Delegation

fig. 3: A recursive groups definition

This notion of group enables the designer to create sophisticated mechanisms of cooperation between agents belonging to a group and even between groups. An example of teams modelling for tactical aircraft simulation can be found in [TID 98].
In our scenario, the division is composed of two sections. And recursively, the section is composed of two aircraft. The organisation is hierarchical and the cooperation is based on delegation. One can assume the leaders of the two groups take joint decision by consulting together, the organisation of Group 1 then becomes a community, but the cooperation mechanism makes essentially intervene the leaders.

4.9. The definition of the agents

In SCALA, a generic structure of agents is defined and specialised by the designer for a given application. It is decomposed in two levels:
The agent itself:
- Skills and the relevant level of specialisation,
- Resources: physical resources of the agent,
- Events: to which the agent is sensitive,
- Current goal.

The agent in its group:
- Group(s): the group(s) it belongs,
- Current role(s): the role(s) it holds in each group(s),
- Possible role(s): the other role(s) it can eventually take in each group,
- Communication Protocol: the protocols of cooperation with the other members of the group.

4.10. The social organisations and the communication protocols

The organisations define the internal relations inside the groups. Given an organisation, the agents can have different roles depending on the situation. These roles can be dynamically assigned. For the moment, we distinguish two types of organisations: hierarchical and community. The organisation can also evolve (under certain conditions) during the simulation depending of the current topology of the graph and of the predefined mechanisms (a hierarchical structure is fit to treat a task naturally decomposed into sub-tasks).
The communication protocols depend on the type of the organisations. For instance, in a hierarchical organisation, the agent which holds a manager role can delegate a task or a set of tasks to another agent or to a group via a simple point-to-point message sending. Whereas in a community, a network is built between several agents to dynamically exchange their tasks or collaborate for their execution. An example of such mechanism is the *Contract Net protocol,* which have been implemented in SCALA.

4.11. The simulation process "event to plans"

Each new relevant event that occurs during the simulation triggers a new instance of the related sub-graphs. The sub-graphs tasks are dynamically distributed to some agents according to their skills, roles and resources. The distribution process (fig. 4) depends on the organisation type of the group to which belong the sensitive agents. The sub-graphs (a multi-agent plan) are split into mono agent plans (lists of partially ordered tasks) as a network of dependencies between the activities of agents. Those dependencies are a representation of the constraints of the graph. Thus, the agents coordinate their activities while taking into account these dependencies and environment changes.

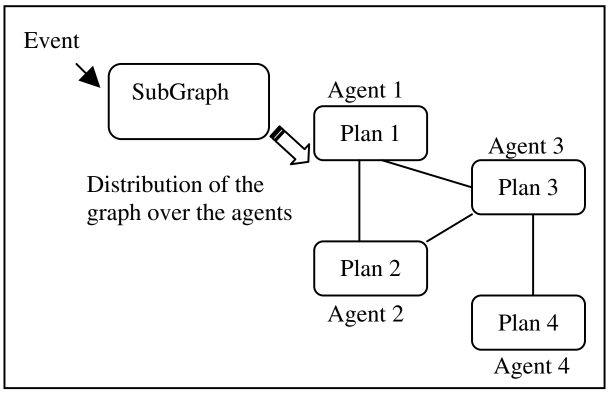


fig.4: Network of dependencies

The SCALA Grapher

Numerous applications can be modelled. To define these applications, we developed a tool to easily design the graphs of dependencies that are associated to an event name, and also build a library of reusable behaviours (the Event/goals pairs). SCALA automatically interprets them and generate the code of the simulation. Only the real contents of the tasks have to be implemented for each application, the mechanisms providing the strategies of cooperation lie on the information entered by the designer through the steps of the methodology. In SCALA, the reasoning part (including coordination between agents) is entirely independent of the basic behaviours.

The activities' scheduler

Another tool to monitor the mission is a scheduler diagram that represents the activities of the agents. On this diagram can be seen the multi-agent behaviour: the evolution of the simulation, the coordination between agents, the arrival of new events.

5. Perspectives

This first approach in SCALA encounters certain limits: the lack of temporal aspects necessary to meet the simulation requirements. Those limits could be critical factors. In addition, we are interested to explore reactive planning in order to manage the environment dynamic (arrival of new goals or events). Consequently, SCALA is extended to take into account time constraints when new events occur. It is also a critical factor when agents have to hold temporal objectives. Furthermore, it is obvious that the behaviour of a group of agent will depend on the available time they have to react.

One of the domains of application of SCALA is tactical aircraft simulation. This domain is characterised by a highly dynamic (unpredictable) and uncertain environment, that implies for the agents to plan reactively when new events occur and sometimes to reorganise themselves. But, the agents have also to respect time constraints in the execution of their tasks and so, the reorganisation and coordination of their activities have to take into account this constraints. Thus the work in progress attempts to extend the graph of dependencies by the notion of time in the tasks, and time constraints on the goals of the agents. These constraints can be assimilated to temporal objectives that are critical factors to achieve successfully the mission. Our approach is real-time execution driven.

6. Conclusion

Through the experience of the use of JACK and the development of SCALA, we have seen that the agent-oriented programming fits to the development of complex system evolving in dynamic environments. The high level of abstraction of the agent approach makes easier the modelling of complex systems, especially the modelling of distributed applications, where the entities (the agents) have to perform tasks autonomously and to interact with others. The designer can focus on sophisticated mechanisms such as coordination and cooperation between the components of the system without getting tangled up in the development of communications for messages sending, or other low-level developments such as multi-threads, etc.

Furthermore this approach allows to easily separating reasoning from actions, and so improves the reusability of the implemented behaviours. In the case of aerial missions simulations, the advantage of such programming is to make reasoning and physical models completely independent. It makes simpler the addition of new behaviours and introduces flexibility in the system. New events can easily be taken into account and managed by the system. These aspects are very interesting for designing systems submerged in a real world (in simulation or in reality), when flexibility, reactivity and adaptation are crucial (responses of new events). Interactivity with human users of the system is also facilitated.

SCALA project proposes a functional approach to design complex systems and provides a tool to rapidly setting up simulations. The designer has to model the global behaviour of the system as a graph supporting functional constraints. The connectors represented in the graph of dependencies of SCALA enable an easier coordination and organisation of the agents driven by their activities. Another important contribution in the design of complex systems is the fact that different behaviours can be easily obtained by only modifying the links and connectors in the graph of dependencies of SCALA and simulated without new compilations. New behaviours are thus specified at a high-level of modelling and then directly interpreted by SCALA to generate the code of the related simulation.

An interesting application of our work is the domain of tactical aircraft simulation, to rapidly prototype tactical behaviours of well-known aircraft or new ones.

Acknowledgements

The author would like to thank Frédéric MARC who works on the related project in the frameworks on his PhD. on Computer Science at LIPN (Laboratory of Computer Science of PARIS 13) and Dassault-Aviation.

References

[BRIOT 01] BRIOT J.-P., DEMAZEAU Y., "Principes et architectures des systèmes multi-agent", Edited by Hermes, pp. 17-70, 2001.

[HOW 01] HOWDEN N., RONNQUIST R., HODGSON A. and LUCAS A., "JACK Intelligent Agents^{TM} Summary of an Agent Infrastructure", *5th Conference on Autonomous Agent*, 2001.

[JAR 01] JARVIS J., MAISANO P, "Jack Intelligent Agents^{TM} User Guide", Release 3.1, Agent Oriented Software Pty. Ltd, Mars 2001.

[JEN 98] JENNINGS N. R.,WOOLDRIDGE M., SYCARA K., "A roadmap of agent research and development", *Int. Journal of Autonomous Agents and Multi-Agent Systems*, vol. 1, n°1, pp. 7-38, 1998.

[RAO 95] RAO A. S., GEORGEFF M. P., "Modeling rational agents within a BDI architecture", *In* J. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, pp. 473-484. Morgan Kaufman Publishers, San Mateo, 1991.

[TID 98] TIDHAR G., HEINZE C., SELVESTREL M., "Flying Together: Modelling Air Mission Teams", *Applied intelligence*, vol. 8, pp. 195-218, 1998.

[WOO 95] WOOLDRIDGE M., JENNINGS N. R., "Intelligent Agents: Theory and practise", *The Knowledge Engineering Review*, 10(2):115-152, 1995.

[WOO 99] WOOLDRIDGE M., "Intelligent Agents", In Multiagent System: A Modern Approach to Distributed Artificial Intelligence, Edited by WEISS G., pp. 27-77, 1999.

Reactive Planning in Air Combat Simulation

Irène Degirmenciyan-Cartault
Dassault Aviation, 78, quai Marcel Dassault Cedex 300 92552 St Cloud Cedex France

irene.degirmenciyan@dassault-aviation.fr

Abstract: In real-world environments such in air combat missions the agents continuously receive perceptual inputs from the environment that is highly dynamic (unpredictable) and uncertain. The aircraft often have to reorganise themselves and to make decisions under time constraints. Two modes of control are thus necessary: planning and reaction. By planning we mean both building a course of action before execution, and reaction as dynamic replanning which interleaves planning and execution. Furthermore, in air combat simulation the plans adopted by the agents in response to external events are known in advance and are not generated by the agents as in other domains. To be reactive, the agents have to choose dynamically the appropriate plans and to coordinate their actions. We present how we take into account new events thanks to dynamic allocation of tasks by means of the graphs of dependencies between agents' activities. Our current work aims to extend this model of tasks and goals by integrating time notions in the selection of action plans and coordinate mechanisms. Operations on plans under time constraints are also examined in order to enable the simultaneous management of several events.

1 Introduction

In real-world environments such as air combat missions the agents continuously receive perceptual inputs from the environment that is highly dynamic (unpredictable) and uncertain. The aircraft often have to reorganise themselves and to take decisions under time constraints. Two modes of control are thus necessary: planning and reaction. By planning we will understand both building a course of action before execution, and reaction as dynamic replanning which interleaves planning and execution. Furthermore, in air combat simulation the plans adopted by the agents in response to external events are known in advance and are not generated by the agents as in other domains. To be reactive, the agents have to choose dynamically the appropriate plans and to coordinate their actions.

The purpose of this paper is to point out the difficulties to coordinate the behaviour of several agents in applications such as air combat simulation. A host of approaches to dealing with reactive planning or multi-agent planning exists, but in the context we are interested, we should simultaneously cope with these two aspects in order to treat the coordination of the agents' activities under time constraints facing to the arrival of new events.

After having outlined the interest of using intelligent agents in the domain of warfare simulations (section 2) and characterised the air combat mission and the requirements in terms of modelling coordination and reactive planning in the distributed context of air combat simulation (section 3). We then focus on reactive and real-time planning (section 4 and 5) and multi-agent planning (section 6). In next sections, we present the work led at Dassault Aviation, and particularly the reactive planning process implemented in the SCALA environment (Cooperative System of Software Autonomous Agent) through simple air missions scenarios. We show how SCALA deals with new events by dynamic allocation of tasks to the agents via the use of graphs of dependencies between their activities, and we also define plan operators enabling a more pre-emptive time management.

2 Thinking with agents

Intelligent software agents have been successfully used for modelling human decision making. In particular, intelligent agents have already shown their suitability for operational analysis of aerial warfare simulators

[STE 96], or man-in-loop simulation as computer generated forces [ILR 97]. Some studies are going so far as to the concept of interchanging humans and agents [HEI 01], or to simulating human performance while taking into account factors such as experience, attention, workload and stress [LLO 97].
Thus, the use of intelligent agents focused on the modelling of human reasoning. The **BDI** model of rational agency have provided the basic paradigm of much of the research in this domain [RAO 95]. The power of this model is the ability to describe folk-psychological notions of **belief, desire and intention**, which helps to describe some aspects human decision making. Furthermore, the appropriate level of abstraction of this model makes it well understood by the analysts and decision makers who are exactly their users. In particular, intelligent agents have widely proven their utility in the modelling of tactical decision process of pilots and fighter-controllers by easily involving the operational air force personnel in the design and development of these kinds of simulations [HEI 98].

3 Air combat mission characteristics

We assume that aircraft and pilots are modelled by intelligent agents. Modelling an air combat mission implies the following points:

- The agents continuously receive perceptual inputs from the environment that constitutes their beliefs of the world (establishment of the situation awareness).
- The environment is highly dynamic (unpredictable) and uncertain, and the aircraft often have to reorganise themselves (dissolution and formation of new teams). For example, reorganisation is performed when a failure occurs or when an aircraft is destroyed (the others have to reconfigure themselves and reassign their roles).
- The agents have to take decisions under time constraints.
- The plans adopted by the agents in response to external events or to accomplish goals are supplied in advance (generally at the briefing before the mission) and are not generated by the agents as in other domains. The agents have to select an applicable plan in their directory of plans under several conditions (this choice is context dependent).
- The aircraft in teams (sections, divisions, packages) have joint goals to achieve (intercept a bandit), and joint plans (team tactics).
- Each member of a team may respect the constraints imposed by the type of organisational structure, and its role within it. The functional responsibilities adopted by the members depend on the goal assigned to the team. A same aircraft can be leader (organisational role) and escorter (functional role).

In a such context, the problems addressed are the coordination and the synchronisation of agents activities under time constraints. So, we will firstly bring reactive and real-time planning to our attention, and secondly, we will focus on multi-agent planning.

4 Reactive planning

As we saw below, air combat simulation involves coordination, and reactive planning in a multi-agent context. Let us compare conventional planners and reactive planners [ASH 00]:

Conventional planners are characterised by their ability to establish a deterministic sequence of actions for getting from a given initial state to a given goal state. They successfully can be applied to problems that are completely defined in a formal way. The original conventional planners attempt to find a path to reach the goal state; they did not really plan, and can also be considered as search algorithms. They do not exhibit recovery notions and can fail if some uncertainties exist in the initial hypothesis or whether the current state of world changes during the planning process.
By contrast, reactive planners construct plans which are able to respond to a large number of world states while working with some type of monitoring of plan execution that keeps track of the world states at all times to respond accordingly.

The notions of reaction and reactive planning have to be jointly considered. Reactive planning takes place before plan execution, although reaction takes place at execution time. These two types of control have to work together to anticipate and avoid critical situations (pro-active behaviour).

A very important notion in reactive planning is the notion of contingency: "Contingency is any state of the world entered by the executing agent while following a plan; which state should not have occurred as a result of executing the plan up to that point" [ASH 00]. The uncertainty characteristic of real-world domains involves the apparition of contingencies during the plan execution. D.W. Ash and V.G. Dabija classified contingencies in three types related to the limited resources that an agent can use, and according to the action taken at planning time to prepare for their occurrence at execution time:

- Contingencies for which the planner builds complete conditional branches, from the contingency state to the goal state, in the main plan (contingencies with high likelihood of occurrence and requiring elaborate plans to treat them).
- Contingencies for which the agent prepares reactive responses; these may be combined into reactive plans which are integrated in the complete plan (to stabilise the situation in a short time), after what a more extensive planning can be necessary at execution time.
- Contingencies ignored by the agent at planning time, either their treatments can be left for dynamic replanning during execution (contingencies with low likelihood of occurrence, and without short-term disastrous consequences), because they are less important than the below categories and do not require replanning at all, and in last case, when the agent can’t take any action to solve the problem.

It seems obvious that an agent, with limited resources, considered in a real world cannot handle all of the contingencies at planning time (concept of "universal plan"). Fortunately, many of the contingencies can be ignored at planning time (for example those which have a low likelihood of occurrence). The problem for the agent is thus to decide which contingencies require reactive responses and those which can be ignored at planning time.

Two control modes are thus necessary: planning and reaction [HAY 93]. By planning we mean both building a course of action before execution, and dynamic replanning which interleave planning and execution. The most important disadvantage of the planning approach is the inflexibility of the planned behaviour. The agent is constrained to act according to the states of the world strictly specified in the plan and leading it straight to his goal failing at the first not planned disrupting event. The reactive approach is more flexible: the agent acts according to a set of perception-action rules which allow it to respond to a larger set of run-times conditions in short-time. Thus, he does not build a complete solution to the final goal, but can quickly stabilise the situation. Therefore, this model of control provides a less carefully in-depth analysis of the current state and the related action consequences.

These two models complement each other, and have to be implemented together, that is what is called reactive planning. In real-world dynamic environments, where short-term responses are necessary, agents need frameworks for action selection. Such frameworks, detailed in [ASH 00], ensure the choice of the best set of events (contingencies) and reactions to be stored at planning time before execution, according to the perception capabilities (sensors), and the reaction execution mechanisms of the agent. Furthermore, planning and reaction techniques can be used to plan the agent’s sensing activities.

5 Real-time planning

In mission-critical system it is imperative to ensure that the proposed solutions have a correct temporal behaviour before they are deployed, and that these solutions have been elaborated under time constraints. The Maruti operating system [LEV 89] [LEV 90] provides tools to build verifiable real-time system services including hard real-time, distributed operation, and fault-tolerance, which are crucial in mission-critical

systems. In this work, the time-driven approach is used, where the control of the resources is done by the operating system scheduler based on the time elapsed in an operation and on absolute time value. This work has shown that a time-driven design is simpler and more easily verifiable.
This kind of system can support the development of dynamic reaction system, which may guarantee performance characteristics, suitable for mission-critical applications.

An example of such a time-driven system applied to aircraft simulation is the Cooperative Intelligent Real-Time Control Architecture: CIRCA.
CIRCA [MUS 93] is an architecture that aims at monitoring environment changes and agent knowledge modifications in order to pre-empt possible failures in the system. It blends the real-time and reasoning requirements by executing them on two separate components [ASH 00]:

- AIS: the artificial intelligence subsystem, which performs high-level reasoning about tasks and develops low-level control plans.
- RTS: the real-time subsystem, which ensures a predictable behaviour for the guaranteed execution of these (mission-critical) control plans.

A *Control plan* is a cyclic schedule of simple test-action pairs (TAP). A TAP contains temporal data such as worst-case timing data on: how long it takes to test the preconditions (TEST-TIME), and how long to execute actions (ACTION-TIME). In addition, the TAP is associated with a maximum TAP period (assigned during planning), which represents the longest time interval allowed between invocations of the TAP that can still guarantee to avoid failure.

The automatically derived reactive control plan is guaranteed to meet the domain's deadlines and achieve the system's goals. The architecture makes a fundamental distinction between activities with respect to the types of goals they are interested to achieve:

- *Control-level goals*, which are guaranteed to meet domain deadlines, via the predictable execution of the RTS. They are frequently related to system safety, and correspond to hard deadlines derived from physical relationships between agents, and the environment (*i.e.* collision avoidance must be achieved before their deadlines). The priority is always given to those goals.
- *Task-level goals*, which are executed in a less predictable manner, are achieved on a best-effort basis. The system tries to achieve them when possible, but if time pressure or other resource limitations make this impossible, the system is still considered successful. The deadlines are negotiable by the agents.

CIRCA operates by simultaneously planning new control plans in the AIS that cooperates with the TAP scheduler, while the RTS executes existing Control plans. The RTS can also influence the AIS by giving it feedback about changes in the world. In that case, the AIS has to replan a set of actions and a control plan.

Distributed versions of CIRCA have been developed [MUS 98] [KRE 01]. The pre-emption process is distributed and coordinated. In fact, each agent has the possibility to avoid the possible failure of another agent: if an agent A determines that an agent B is tracked by a missile, it will tell agent B to activate its electronic counter-measures to defeat the missile. This means that each agent has special TAPs to take care of event that could arise to others such as the Detection of a missile.

Other works are focused on the process of planning [ATK 96]. Those works aims at reducing the planning time by typing the different states that could occur in the environment. Actually, they try to determine in which case the system has to plan a complete control plan or execute an existing Control plan. The main topic of this work is the time that the system has to react to the events. If the transition to failure comes “fast”, then it executes an existing plan. Otherwise, if the transition is “slow”, it will plan a new Control plan. The time allocated to the planning process dependens on the time before the transition occurs.

CIRCA essentially deals with the safety of the system and not with the achievement of high-level goals. The safety is ensured thanks to a real-time monitoring of the environment, where any changes implies the planning of a new control plan with real-time constraints.

6 Multi-agent planning

An important aspect encountered in Air-Combat simulation is the need for coordinated behaviour modelling. The coordination issue is central in multi-agent systems since the agents that constitute a multi-agent system have to coordinate their activities. A way to enhance agents' cooperation is to give them capabilities that enable to forecast and plan cooperatively their actions. Several works deal with multi-agent planning as a coordination strategy [GEO 83] [DUR 88]. Generally, these works address the post planning. First, the agents generate individual plans and then attempt to merge them in a global plan (the multi-agent plan) where the execution of the local plans is compatible. This post planning process lies on the detection of sub-goals and conflicting situations in order to remove them.

One of the major properties of multi-agent planning is the distribution that distinguishes it from traditional (centralised) planning. Although this notion is inherent to multi-agent systems, it becomes ambiguous when it qualifies planning. Indeed, we have to question on what is really distributed. Distribution may concern either the planning process itself or the resulting plans (or even both) [DUR 99]. So, different types of multi-agent planning can be considered [ELF 01]:

Centralised planning and distributed plans

The planning process is centralised in a specific agent (the coordinator) and the resulting plan is distributed over the agents. The plan is partially ordered and the parallel actions can be concurrently executed by several agents. The coordinator can be designated after a negotiation cycle between the agents. It is the case of the Air Traffic Control application [CAM 83] where the aim of the agents is to build coherent flying plans.
This type of planning makes conflict easy to solve and the convergence to a global solution, but it suffers from the traditional disadvantages of centralised control, such as the lack of robustness or the communication bottleneck.

Besides, this approach is not very reactive. Every occurring problem during the execution of a plan has to be passed to the coordinator who may decide to activate some replanning operations. So, the coordinator should maintain a continuously updated representation of execution states in order to solve the conflicts.

Distributed planning and centralised plan

We assume here that the problem to be solved (*i.e.* the tasks to be achieved) is decomposed in sub-tasks and each of them is planned by an agent. The agents have to interact with each other to synchronise and merge the local plans in order to constitute a unique multi-agent plan.

This approach is also costly in communications and the achievement of a global solution is not guarantied.

Distributed planning and distributed plans

This approach of distributed planning is undoubtedly the most challenging because it does not assume that global plan exists somewhere in the system, and yet the distributed plans are compatible, i.e. their executions do not cause conflicts between agents [DUR 99].

The agents plan their actions and execute them concurrently and autonomously in a shared environment. Events due to the concurrency of actions can occur during execution or plan generation. So, this type of planning requires rich and powerful formalisms that can express parallelism, concurrency, hierarchical goals, etc.

Incremental planning is one of the approaches proposed in the domain. It is assumed that a set of plans is already coherent and that the planning consists in integrating a new plan to be coordinated with others.
In [VMA 92], a taxonomy of the relations between plans is developed and a communication framework allows plan exchanges and negotiations between agents to take into account these relations when a new plan is inserted. This work concerns the simultaneous coordination of two agents.

El Fallah-Seghrouchni and S. Haddad in [ELF 96a] proposes a distributed planning algorithm that simultaneously coordinate several agents. The formal model is based on the partial order of the actions. The plans' coordination consists of adding synchronisation links between actions. The algorithm solves the positive interactions by using the pre and post-conditions, and the negative interactions by synchronising the actions. The model guaranties the feasibility of the multi-agent plan for all total order whatever is the total order extending the partial order. This formalism of plan representation is extended in [ELF 95] [ELF 96b] to enable the algorithm to solve more complex situations. It lies on an extension of the Petri-net formalism to the Recursive Petri-net formalism. It includes the characteristics of recursivity, dynamism, and interleaving of planning and execution. It allows the representation and the reasoning on simultaneous actions and continuous processes (concurrent actions, choice between alternatives, synchronisation, etc.).
This planning model allows the representation of abstract action during the generation of plans. The refinement of this type of action is dynamic (during the execution) and context dependent which allows the dynamic choice of the manner of executing an action. Furthermore, this model guaranties the consistency of the generated plans with the help of efficient algorithms.

A more detailed presentation of multi-agent planning is beyond the scope of this paper. The interested reader could refer to [ELF 01] and [DUR 99].

Let us now present the studies carried out at Dassault Aviation that attempts to introduce reactivity and time constraints in the coordination of the activities of agents. The application considered is the air combat simulation. For the moment, the aim of our work does not go so far as to integrate real-time constraints in the reorganisation or planning process, but we are attempting to introduce time constraints in the choice of plans and in the choice of tasks assignment to agents, and further, in the goals management too, *i.e.* how the goals have to be taken into account to satisfy the temporal objectives fixed in order to successfully accomplish the mission.

In the next sections, we are describing the reactive planning process implemented in the SCALA environment (Cooperative System of Software Autonomous Agent) through air missions' scenarios. Firstly, we are demonstrating how SCALA handles new events by dynamically assigning the necessary tasks that treat them, and secondly we are defining operators on plans enabling a more pre-emptive time constrained management.

7 The reactive planning process in SCALA

7.1 A dynamic assignment of tasks

In SCALA, the designer models the behaviour of the multi-agent system at a high level of abstraction via a graph of dependencies. This graph is constituted by one or more distinct graphs composed of tasks or basic behaviours that have to be accomplished by the agents of the system, and the constraints between them. These are defined by links or connectors expressing notions of synchronism (on the beginning or on the end of several tasks), exclusion (the execution of one task inhibits others'), refinement (several methods can be invoked to execute a same task) and abstraction (a task is composed of others). Another type of constraint is about the number of agents having to perform a given task. At this stage, tasks are not yet allocated to the agents.

The definition of the graph of dependencies provides the knowledge required for managing the cooperation between the agents. The need of collaboration for the execution of some tasks implies dynamic planning during simulation through specified cooperation mechanisms. This on-line process interleaves planning and execution just like in reactive planning. For example, to fire a missile, the designation of the target can be done by an aircraft, and the fire of the missile by another aircraft. Depending on the available resources, the same aircraft also can accomplish these two actions. This coordination can be either implicit, the agents have joint goals and each of them attempt to accomplish the relevant tasks, or explicit, for example, an agent can ask another agents to help him to accomplish a task because this task requires the synchronisation of their actions.

The assignment of the tasks to the agents is thus made dynamically according to the current situation and the available resources (agents are also assimilated to resources through their skills). So, we give the agents a certain freedom of action that gets more reactivity for the system and allows to avoid some failures such as the use of non-available resources.

7.2 The simulation of an interception mission

In this section, we are describing an interception scenario. Let us assume that an enemy wants to ingress our territory with a "**Bomb target**" goal (fig. 1).
The mission of our aircraft is to protect the territory by intercepting the threats. We assume that a four aircraft division is on alert on a CAP waiting for events such as "**Contact on the radar**". A possible sub-graph associated to this event is modelled in fig. 2. The first task that responds to this event is "**Guided flight**" to achieve the interception. This task has to be executed by at least two agents of the entire division (**$* \geq 2$**). The "Guided Flight" is an interruptible task (grey colour), which implies it can be abandoned as soon as the next task "**Acquisition**" is possible. When the agents have obtained the permission to engage the bandit (pre condition of "Engagement" satisfied), the best-placed agent does it.
SCALA enables the on-line addition of new agents (arrival of a new bandit in the theatre). Let us assume now that the designer creates a new bandit following the same type of graph than the first bandit before this one had been treated. The reception of this new event by the Division implies it has now two bandits to treat. A new goal associated with a new instance of the same graph is generated.

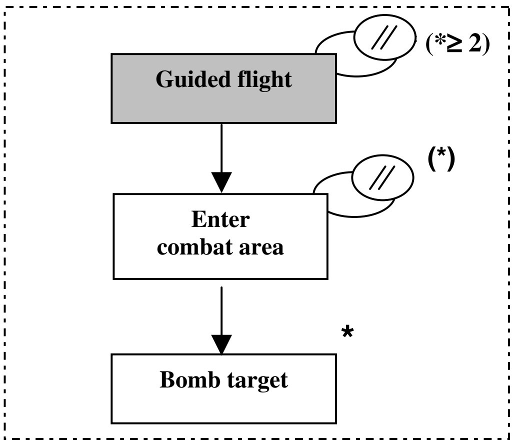


fig. 1: Sub-graph "Bomb Target"

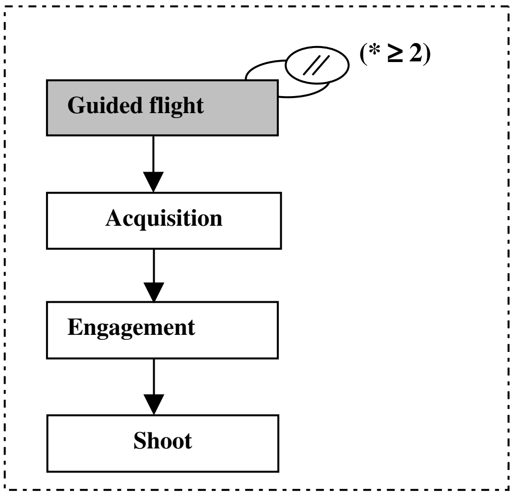


fig. 2: Interception Sub-Graph

In our domain, plans are predefined, so we do not have to generate new plans to respond simultaneously to several events, but only to manage their combination. In the next section, some of our operators on plans will be defined.

We favour that goals must be treated concurrently if the agents have enough resources. The agents are considered as resources (skills and role), and physical resources of the agents, such as missiles are also taken into account. In the other case, the priority of urgency of the goals has to be deal with. The complete heuristic of goals' management is explained beside (fig. 3).

In our scenario, we assume the division has enough resources to treat simultaneously the two threats. The division thus split into two groups: Section1 assigned to the first target, Section2 assigned to the second target. This decision is made by the leader of the division according to the structural organisation specified by the designer.
The difficulty here is to re-organise the aircraft in the best way to take into account the new goal. This reorganisation depends on many parameters (context, physical and temporal resources). The problem is to extract the best required criteria to ensure that each new group (reconfiguration inside the groups is possible) has the minimal resources to achieve its assigned goal. This re-organisation is decisive for the continuation of the mission and requires a time-constrained pre-planning algorithm on a high-level of decision.

```
Priority(goal G1, goal G2)
Begin
        If enough Resources
                Treat G2 and G1 concurrently;
        End If
        If  not enough Resources
                If P2 > P1
                        Treat G2 then G1;
                End If
                If P2 ≤ P1
                        Treat G1 then G2;
                End If
        End If
End
```

fig. 3: Goals management algorithm

Then, Bandit2 ripostes (he replies also to an event such as “Contact on the radar” by a sub-graph “Counterattack”) and we assume to simplify the scenario that he is immediately shot by Section2. This Section examines if there is some bandits left. That is why they consider Bandit1 as their new target and decide to follow him. The two sections have now the same goal. Finally, Section1 shoots Bandit1, and they can return home because there are no more bandits in the friend territory. The mission is completed.

This scenario is depicted in fig. 4, 5 and 6 on a visualisation tool enabling to situate the agents and show their behaviour. SCALA provides other tools, independent of the application, to analyse the behaviour of the system (dynamic information on the states of the agents and on their activities).

SCALA has been developed on top of the JACK agent-oriented language [HOW 01], itself built on top of Java.

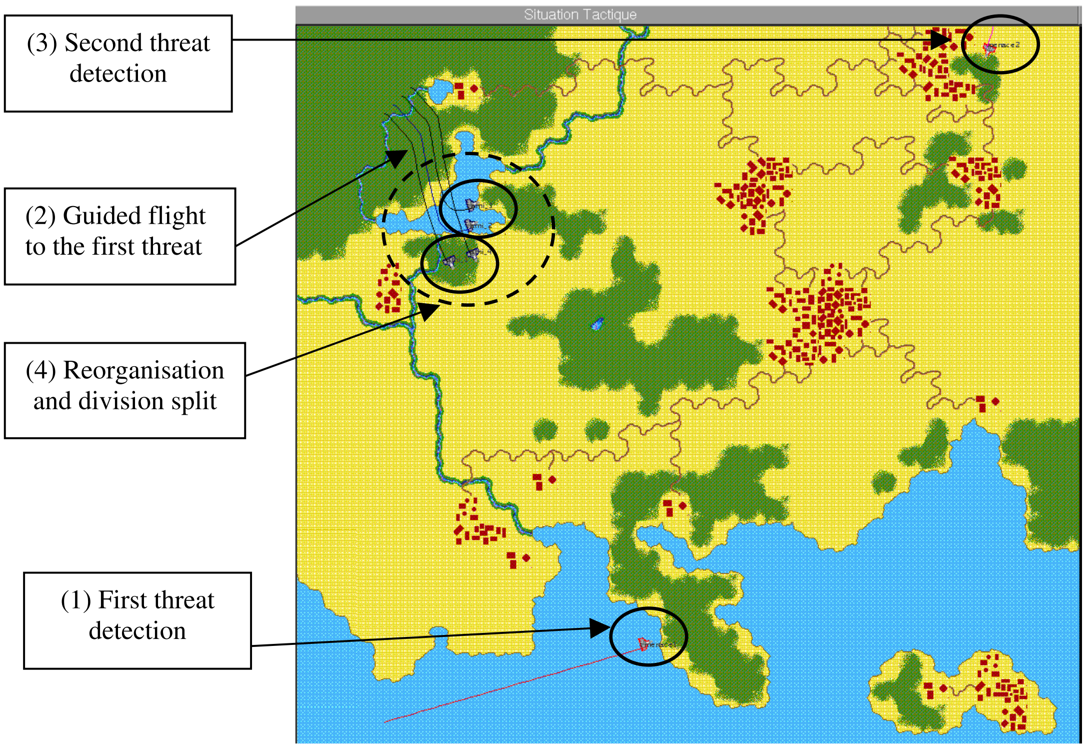

Situation Tactique
(3) Second threat detection
(2) Guided flight to the first threat
(4) Reorganisation and division split
(1) First threat detection

fig. 4: An air combat scenario developed in SCALA (1)

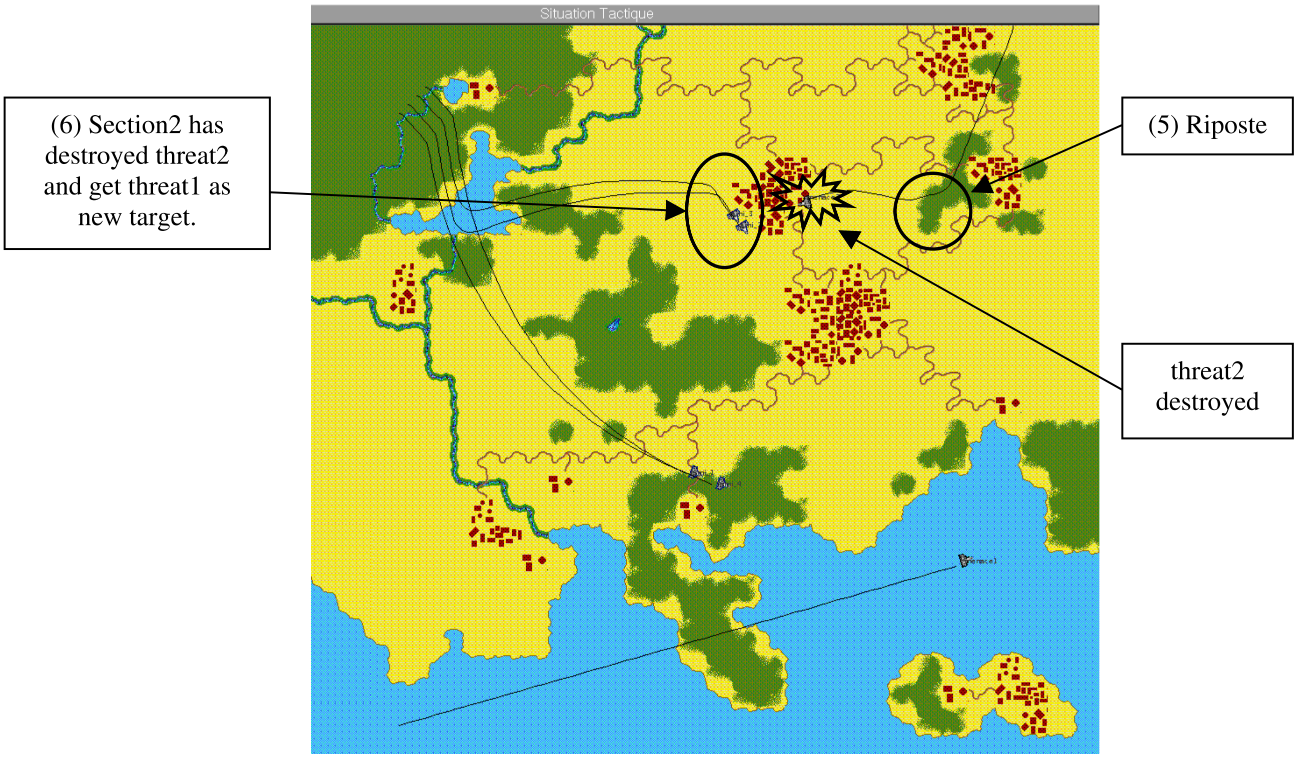

Situation Tactique
(6) Section2 has destroyed threat2 and get threat1 as new target.
(5) Riposte
threat2 destroyed

fig. 5: An air combat scenario developed in SCALA (2)

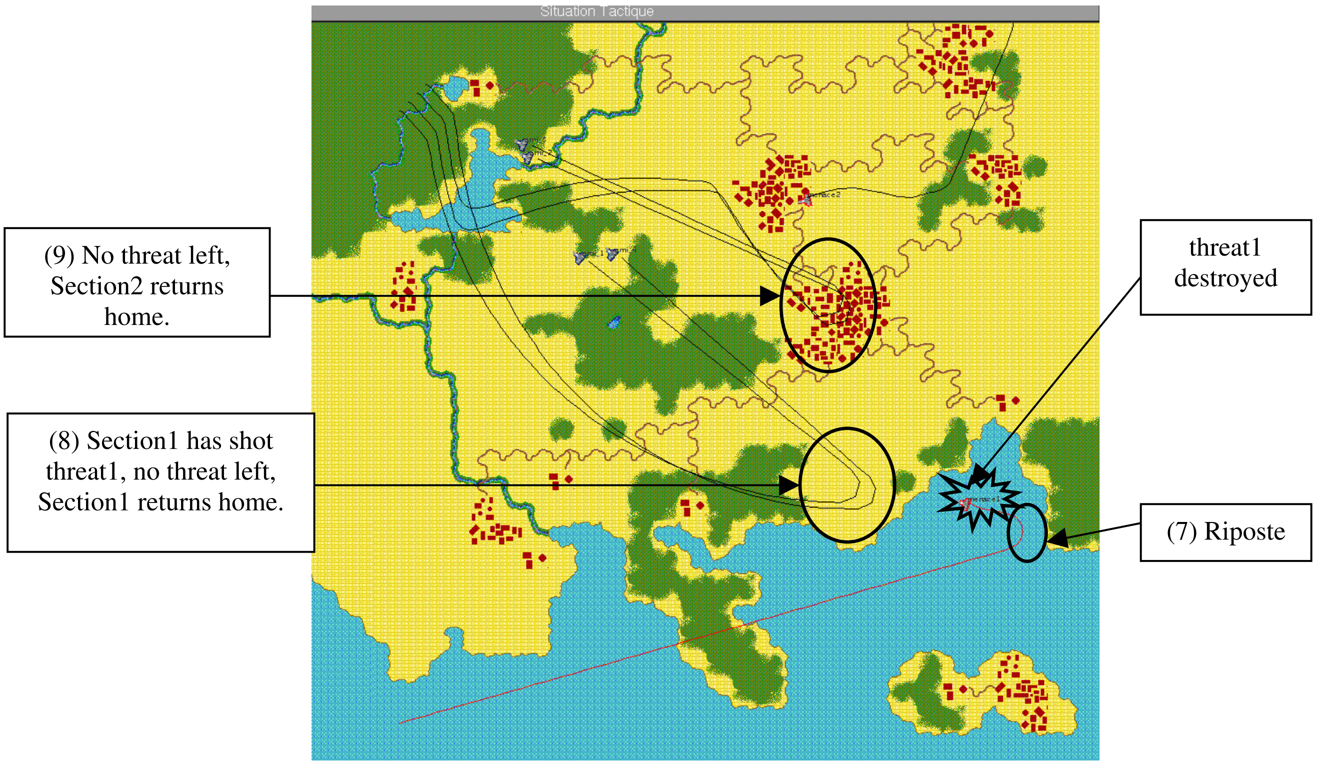

Situation Tactique
(9) No threat left, Section2 returns home.
threat1 destroyed
(8) Section1 has shot threat1, no threat left, Section1 returns home.
(7) Riposte

fig. 6: An air combat scenario developed in SCALA (3)

Several works have been developed in multi-agent simulations like SWARMM [HEI 98] [MIL 96] based on dMARS [DIN 97] or TacAir-Soar [ROS 94].

SWARMM is a detailed multi-agent simulation of fighter combat designed for analysing the impact of upgrades to modifications and development of the tactical employment of aircraft. It is capable of simulating the physics of air missions and the pilot's reasoning process involved in such missions. In SWARMM, the behaviour of the agents is based on the choice of relevant pre-specified plans supported by the meta-level reasoning of the agent-oriented software dMARS. So, the process of replanning is entirely based on the direct choice of behaviours through predefined plans, *i.e.* all the behaviours, even team tactics are implemented in specific plans [TID 98]. The essential difference between the graph of dependencies of SCALA and SWARMM plans, is that the choice of behaviours in SCALA is less determined, or more exactly let at the last moment. The graphs are a kind of factorisation of plans, and represent a class of behaviours responding to a same goal. It is the coordination mechanisms that instantiate at the last moment the graph, which becomes a plan. This approach allows making lighter the definition of the plans and their pre-conditions. In SCALA, a first level choice is made for the relevant graph, and a second level choice is made by the agents themselves through the heuristics of coordination.

The TacAir-Soar system also combines reactive and goal-driven reasoning. It contains a large set of rules that fire as soon as their conditions are met, without search or conflict resolution. The choice of the actions to be performed entirely lies on the interpretation of the environment observations [JON 94]. Another part of this research effort deals with a deliberative planning component that separate planning from normal execution by projecting future possible states and searching through them which courses of action are appropriate. The challenge of this research is to integrate deliberative planning with dynamic reasoning.

7.3 Time constraints

Our current work in collaboration with the LIPN (Laboratory of Computer Science of Paris 13) consists to extend the SCALA framework with temporal aspects in order to satisfy time constraints at the execution level. Let us now introducing the expected improvements of SCALA.

A part of our current research effort is to equip the graph of dependencies with the notion of time. The graph or more precisely the sub-graphs can be constrained in terms of temporal objectives. In fact, the global time constraint on a graph is the resultant of time constraints on its tasks. The main idea is that time constraints can be added to the graph, giving a limited duration of the plan execution. This notion is crucial in the context of a real-time environment where the execution time allocated to agents to accomplish tasks is critical. Our approach is real-time execution driven.

Time constraints are defined as intervals of time to be associated with the tasks as in [VMA 92]. We place our work in the framework of real-time execution that is to say that the agents have temporal objectives on the execution of their tasks. We attempt to develop an anytime approach for task execution such as the "contract algorithms" [ZIL 96]. We think that temporal objectives should be an important criteria in the process of planning because the behaviour of a group of agents depends on the time it has to react. So, the available time must be considered as a resource and the choice of the relevant graph must take into account both physical and temporal resources.

7.4 Operations on plans

The choice of the relevant operations is time dependant. We take the hypothesis that we will always try to treat all the goals concurrently if there are enough resources in the system. The following section is devoted to describe the operators on plans and finally depicting several behaviours.
Before choosing what type of operation an agent is allowed to do, he verifies if he can maintain the global goal. In fact, an agent can have different behaviours: to keep his current plan and to cope with the new allocated tasks or to distribute his current plan (i.e. set of tasks) over the other agents. In the last case, we consider that the agent does not maintain the global goal himself but this goal is maintained by the others. In

an extreme case, the agent cannot maintain the global goal, *i.e.* he cannot keep his initial set of tasks and simultaneously manage the new tasks.

Global goal is maintained

We have developed several operators to manipulate the plans: concatenation, parallelisation, and insertion (merging). These operators are invoked when the global goal is maintained:

- *Concatenation*: This operator sequences two plans. The interest of a simple concatenation is limited. In fact, it would be interested, in a general manner, to re-examine the two plans, by adding or deleting tasks such as redundant tasks. With such an operator, the dependencies are not updated.
- *Parallelisation*: The two plans are performed concurrently. To use this operator, possible conflicts about resources have to be checked. In this case, the dependencies are not updated because they do not interfere with the global goal. So, no updating with other agents is necessary.
- *Insertion*: The insertion of plan consists in adding a new set of tasks into the current plan. This implies an updating of the dependencies with the other agents. In fact, even if the global goal is not changed in terms of objectives, the constraints on those objectives (such as the delays of execution) could be modified.

Global goal is not maintained

When the global goal is not maintained, the agent has two possible behaviours. Actually, its behaviour depends if it evolves alone or with others. In the first case, if it cannot cope with the new goal, also it removes it and starts again from a *home state*. But in the second case, it distributes its current set of tasks over the other agents of its group.

7.5 Some examples of behaviours

The presented scenario involves two "Friend" planes (Friend1 and Friend2) composing a patrol and firstly, a single bandit B1. The threat is following a flight plan and the patrol has a goal that is "To intercept the threat". As in the first scenario, the patrol fulfils a classic plan to intercept this menace. The patrol succeeded when the threat is destroyed. In this scenario, we create a new bandit B2 following a flight plan too. This involves the arrival of a new goal for the patrol that takes it into account.
Several alternatives should be applicable to cope with the arrival of a new event (fig. 7):

1. To keep the entire patrol and to insert the goal if its priority is higher than the current one (the insertion operator is used)
2. To keep the entire patrol and to concatenate the goal if its priority is lower or equivalent than the current one (the concatenation operator is used)
3. To keep entire the patrol and to abandon the goal an to start again from a home state if there is no available resources or if the friends are outnumbered (the goal is not maintained)
4. To split the patrol and to simultaneously treat the two bandits if there are enough available resources to treat them concurrently (the parallelism operator is used).

Actually, it is obvious that treating the two goals concurrently takes less time than sequentially. But this point could also be tactical since a patrol is generally stronger than an isolated aircraft. As a consequence, if there is enough available time for the patrol to achieve sequentially the goals, it could sometimes be better than splitting it. We assume that those operational aspects are under the responsibility of the designer and he has to specify them in the graph of dependencies.

Work in progress tackles the issue of validation of the generated plans after the operations. Our approach is based on timed automaton. They present the advantages to model some of our connectors as the refinement and the abstraction [ALU 94] [BER 97] based on different semantics of time, considering the action as instantaneous or with duration. In this approach, the multi-agent plan is modelled as a synchronised network of timed automata where each agent plan as a timed automaton. The main interest is to control plans execution while taking into account the action duration and the arrival of new events.

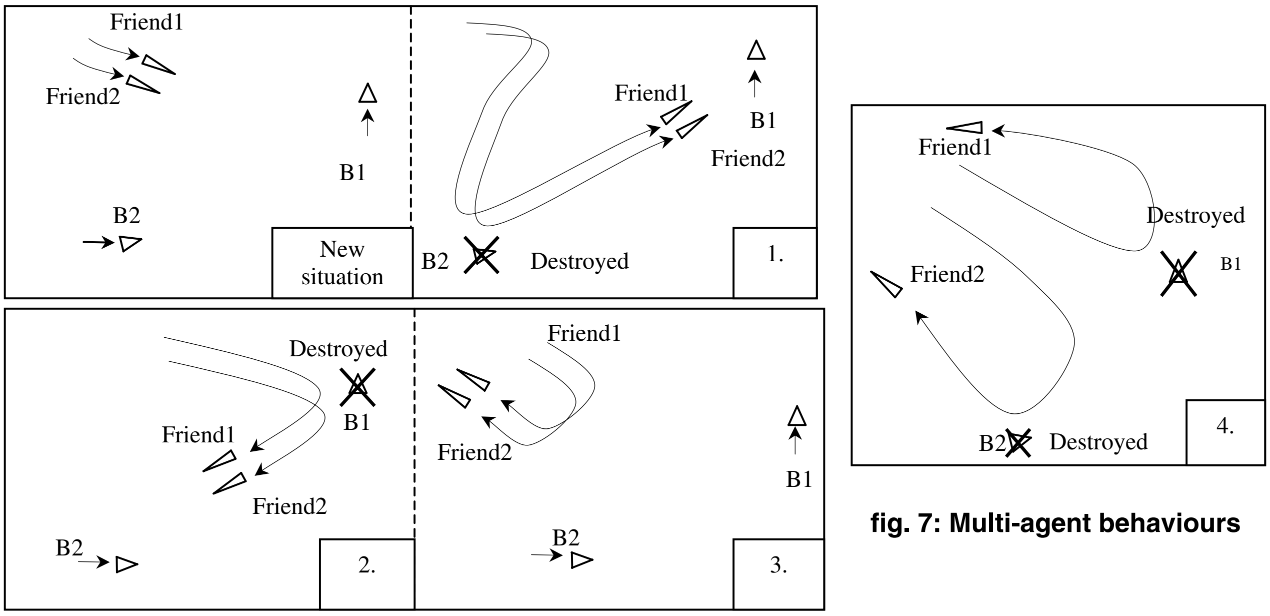


fig. 7: Multi-agent behaviours

8 Conclusion

In real-world environments such as air combat mission the agents continuously receive perceptual inputs from the environment that is highly dynamic (unpredictable) and uncertain. The aircraft often have to reorganise themselves and to take decisions under time constraints. In this paper we focused on the multi-agent technology that provides an interesting framework to design human-like behaviours and to model collective behaviours and also on reactive planning that is appropriated to deal with the related domain which requires both reactive and pro-active behaviours.

The SCALA project based on the multi-agent paradigm proposes a functional approach to design complex systems and provides tools to rapidly setting up simulations. The designer is supported by tools enabling the modelling of high-level behaviours through the graphs of dependencies that contain the necessary information to coordinate agents' activities. Indeed in air combat simulations the plans adopted by the agents in response to external events are known *a priori* and are not generated by the agents as in other domains, but the agents have to dynamically choose the relevant plans and then to coordinate their actions.

Our current work led in collaboration with the LIPN (Laboratory of Computer Science of PARIS 13) focuses on the extension of the SCALA graphs to temporal aspects, that have to be taken into account in the agents' plans and in the coordination mechanisms. Indeed the agents in such applications have to respect duration on their tasks or on the achievement of their joint goals. Operations on plans under time constraints are also examined to enable the simultaneous management of events. These operators enable the agents to cooperate through the concatenation, the insertion or the parallelisation of their plans. The formalism that we are exploring is close to the timed automata and should allow to easily model some of our connectors such as the refinement and abstraction connectors and to validate the generated plans [ALU 94] [BER 97].

Acknowledgements

The author would like to thank Amal EL FALLAH-SEGHROUCHNI, researcher at the LIPN (Laboratory of Computer Science of PARIS 13) for her collaboration on this work with Dassault Aviation and discussion. She also would like Frederic MARC who works on this project in the framework of his Ph.D. on Computer Science at LIPN and Dassault Aviation.

Reference

[ALU 94] ALUR R., DILL D. L., "A Theory of Timed Automata", *Theoretical Computer Science* Vol. 126, p. 183-235, 1994.

[ASH 00] ASH D. W., DABIJA V. G., ."Planning for Real-Time Event Response Management", Edited by Prentice Hall PTR, 2000.

[ATK 96] ATKINS E. M., DURFEE E. H., SHIN K. G., "Detecting and Reacting to Unplanned-for World States", 1997.

[BER 97] BERARD B., GASTIN P. and PETIT A., "Refinement and abstraction for timed languages", Research Report LSV-97-3, Apr. 1997.

[CAM 83] CAMMARATA S., McARTHUR D., STEEB R., "Strategies of Cooperation in Distributed Problem Solving", In *Proceeding of the 8th International Joint Conference On Artificial Intelligence,* pp. 767-770, Karlsruhe, Deutschland, 1983.

[DIN 97] D'INVERNO M., KINNY D., LUCK M., WOOLDRIDGE M., "A formal specification of dMars", In M. Wooldridge and A. Rao, editors, Fourth International *Workshop on Theories, architectures and Languages*, 1997.

[DUR 88] DURFEE E. D., ."Coordination of Distributed Problem Solvers", Kluwer Academic Press, Boston 1988.

[DUR 99] DURFEE E. D., ."Distributed Planning ", In *Multi-agent Systems: A modern Approach to Distributed Artificial Intelligence*, MIT Press, pp. 139-149, 1999.

[ELF 95] EL FALLAH-SEGHROUCHNI A., HADDAD S., "A Formal Model for Coordinating Plans in Multi-Agents Systems", In *Proceeding of Intelligent Agents Workshop,* Augusta Technology Ltd, Brooks University, Oxford, Grande-Bretagne, 1995.

[ELF 96a]. EL FALLAH-SEGHROUCHNI A., HADDAD S., "A Coordination Algorithme for Multi-Agents Planning", In *Proceeding of MAAMAW'96,* Springer-Verlag, LNAI 1038, Eindhoven, Pays-Bas, 1996

[ELF 96b]. EL FALLAH-SEGHROUCHNI A., HADDAD S., "A Recursive Model for Distributed Planning", In *Proceeding of ICMAS'96 (International Conference of Multi-Agnets Systems),* AAAI Press, Kyoto, Japan, 1996

[ELF 01] EL FALLAH-SEGHROUCHNI A., "Modèles de coordination d'agents cognitifs", In *Principes et architectures des systèmes multi-agent*, Edited by Hermes, pp. 139-176, 2001.

[GEO 83] GEORGEFF M. P., ."Communication and Interaction in Multi-Agents s of ", In *Proceeding of the Third National Conference on Artificial Intelligence (AAAI-83)*, pp. 125-129, July 83.

[HAY 93] HAYES-ROTH B., "Opportunistic Control of Action in Intelligent Agents", *IEEE Transactions on Systems, Man and Cybernetics*, Special Issues on Planning, Scheduling, and Control: An AI-based Integrated System View, March 1993.

[HEI 98] HEINZE C., SMITH B., CROSS M., "Thinking Quickly: Agents for Modelling Air Warfare", In Proceedings of Australian Joint conference on Artificial Intelligence, AJCAI 98, Brisbane, Australia, 1998.

[HEI 01] HEINZE C., GOSS S., JOSEFSSON T., BENNETT K., WAUGH S., LLOYD I., MURRAY G., OLDFIELD J., "Interchanging Agents and Humans in Military Simulation", In AAAI Workshop on the Innovative Application on AI, 2001

[HOW 01] HOWDEN N., RONNQUIST R., HODGSON A. and LUCAS A., "JACK Intelligent Agents™ Summary of an Agent Infrastructure", In *Proceedings of the fifth Conference on Autonomous Agent*, 2001.

[ILR 97] McILROY D., HEINZE C., APPLA D., BUSETTA P, TIDHAR G., RAO A., "Towards Credible Computer-Generated Forces", In *Proceedings of The International Simulation Technology and Training conference*, (SimTecT 97), 1997.

[JON 94] JONES R. M., LAIRD J. E, "Multiple information sources and multiple participants: Managing situational awareness in an autonomous agent", In *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, May 1994.

[KRE 01] KREBSBASCH K. D., MUSLINER D. J., "Multi-Agent Mission Coordination via Negotiation*", Working Notes of the AAAI Fall Symposium on Negotiation Methods for Autonomous Cooperative Systems*, 2001.

[LEV 89] LEVI S. T., TRIPATHI S. K., CARSON S. D., AGRAWALA A. K., "The MARUTI Hard Real-Time Operating System", ACM Operating System Review, Vol. 23, N°3, June 1989.

[LEV 90] LEVI S. T., AGRAWALA A. K., "Real-Time System Design", McGraw Hill Publishing Co., New-York, 1990.

[LLO 97] LLOYD I., "Simulating Human Characteristics for Operational Studies", DSTO Research Report DSTO-RR-0098, 1997.

[MIL 96] McILROY D., HEINZE C., "Air Combat Tactics in the Smart Whole AiR Mission Model", In *Proceedings of the First International Simulation Technology and Training conference*, (SimTecT 96), Melbourne, Australia, 1996.

[MUS 93] MUSLINER D. J., "CIRCA: The Cooperative Intelligent Real-Time Control Architecture" *Ph. D. Dissertation*, University of Michigan, Ann Arbor. Available as University of Maryland Computer Science Technical Report CS-TR-3157.

[MUS 98] Issues in Distributed Planning for Real-Time Control D. J. Musliner, K. D. Krebsbasch, M. J. S. Pelican, R. P. Goldman, M. S. Boddy, In *Working Notes of the AAAI Fall Symposium on Distributed Continual Planning*, Oct. 1998

[RAO 95] RAO A. S., GEORGEFF M. P., "Modeling rational agents within a BDI architecture", In J. Allen, R. Fikes, and E. Sandewall, editors, In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, pp. 473-484. Morgan Kaufman Publishers, San Mateo, 1991.

[ROS 94] ROSENBLOOM P. S., JOHNSON W. L., JONES R. M., KOSS F., LAIRD J. E, LEHMAN J. F., RUBINOFF R., SCHAMB K. B., TAMBE M., "Intelligent automated agent for tactical air simulation: A progress report", In *Proceedings of the fourth Conference on Computer Generated Forces and Behavioral Representation*, May 1994

[STE 96] STEUART S., MURRAY G., TIDHAR G., RAO A., "Air Mission Modelling - An Application for Articicial Intelligence", In *Proceedings of The First International Simulation Technology and Training conference*, (SimTecT 96), Melbourne, Australia, 1996.

[TID 98] TIDHAR G., HEINZE C., SELVESTREL M., "Flying Together: Modelling Air Mission Teams", *Applied intelligence*, vol. 8, pp. 195-218, 1998.

[VMA 92] VON MARTIAL F., "Coordinating Plans of Autonomous Agents", Springer-Verlag, LNAI, 1992.

[ZIL 96] ZILBERSTEIN S., "Using Anytime Algorithms in Intelligent systems", In *Proceedings of the National Conference on Artificial Intelligence (AAAI),* 1996.

The TechSat-21 Autonomous Sciencecraft Experiment

Steve Chien, Rob Sherwood, and Richard Doyle
Jet Propulsion Laboratory
California Institute of Technology
MS 126-221/4800 Oak Grove Drive
Pasadena, California 91109-8099 USA

chien@jplnasa.gov, rsherwood@jpl.nasa.gov, rdoyle@jpl.nasa.gov
http://asc.jpl.nasa.gov

Abstract

The Autonomous Sciencecraft Experiment flight demonstration (ASE) will fly onboard the US Air Force's TechSat-21 constellation, an unclassified mission scheduled for launch in 2004. ASE will use onboard science analysis, replanning, robust execution, and formation flying to radically increase science return by enabling intelligent downlink selection and autonomous retargeting. Demonstration of these capabilities in a flight environment will open up tremendous new opportunities in planetary science, space physics, and earth science that would be unreachable without this technology.

1 Introduction

There is an increasing desire in many organizations, including NASA and the US Department of Defense, to use constellations or fleets of autonomous spacecraft working together to accomplish complex mission objectives. The Air Force Research Laboratory (AFRL) has initiated the TechSat-21 program to serve as a proof of concept mission for a new for space mission paradigm. This paradigm seeks to reduce costs and increase system robustness and maintainability by distributing functionality over several micro-satellites flying in formation. The distributed functionality includes processing, command and control, communications, and payload functions. A chief objective is for the system of micro-satellites to in effect function as a "virtual" satellite, which can be controlled and tasked as if it were a single satellite.

TechSat-21 is scheduled for a late 2004 launch and will fly three satellites in a near circular orbit at an altitude of 600 km (See Figure 1.) The primary mission is one year in length with the possibility for an extended mission of one or more additional years. During the mission lifetime the cluster of satellites will fly in various configurations with relative separation distances of approximately 100 meters to 5 km. One of the objectives of TechSat-21 is to assess the utility of the space-based, sparse-array aperture formed by the satellite cluster. For TechSat-21, the sparse array will be used to synthesize a large radar antenna. Three modes of radar sensing are planned: synthetic aperture radar (SAR) imaging, moving target indication (MTI), and geo-location.

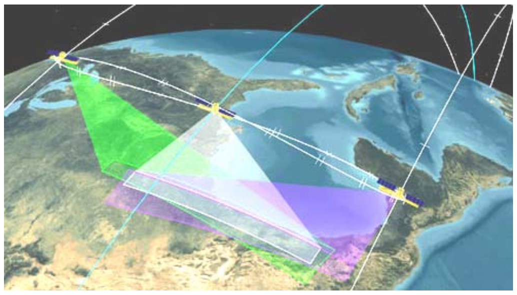


Figure 1 – TechSat-21 Configuration

The principal processor onboard each of the three TechSat-21 spacecraft is a BAE Radiation hardened 175 MIPS, 133MHz PowerPC 750 running the OSE 4.3 operating system from Enea Systems. OSE was chosen because it is inherently message-passing-based and particularly suitable for distributed applications. Each satellite will have 256 Kbytes of EEPROM for boot loads and 128 Mbytes of SDRAM. Communications will be through a Compact PCI bus. For storage of payload data and some large flight software components eight disk drives per spacecraft will be used.

The ASC onboard flight software includes several autonomy software components:

- *Onboard science algorithms* that will analyze the image data, generate derived science products, and detect trigger conditions such as science events, "interesting" features, and changes relative to previous observations.
- *Robust execution management software* using the Spacecraft Command Language (SCL) [7] package to enable event-driven processing and low-level autonomy.
- The Continuous Activity Planning, Scheduling, and Replanning (CASPER) [4] *planner* that will replan activities, including downlink, based on science observations in the previous orbit cycles.
- The ObjectAgent and TeamAgent *cluster management software* that will enable the three TechSat-21 spacecraft to autonomously perform maneuvers and high-precision formation flying to form a single virtual instrument.

The onboard science algorithms will analyze the images to extract static features and detect changes relative to previous observations. Prototype software has already been demonstrated on X-band radar data (from Space Shuttle missions) to automatically identify regions of interest including: regions where change has occurred, such as flooding, ice melt, and lava flows, as well as regions containing recognizable features of interest such as craters and volcanoes. Such onboard science will enable retargeting and search, e.g., shifting the radar aim-point on the next orbit cycle to identify and capture the full extent of a flood. Onboard science analysis would also enable capture of short-lived science phenomena at fine time-scales without overwhelming onboard caching or downlink capacities. Future examples can include: eruption of volcanoes on Io, formation of jets on comets, and phase transitions in ring systems. Generation of derived science products (e.g., boundary descriptions, catalogs) and change-based triggering will also reduce data volumes to a manageable level for extended duration missions that study long-term phenomena such as atmospheric changes on Jupiter and flexing and cracking of the ice crust on Europa.

The onboard planner (CASPER) will generate mission operations plans from goals provided by the onboard science analysis module. The model-based planning algorithms will enable rapid response to a wide range of operations scenarios based on a deep model of spacecraft constraints, including faster recovery from spacecraft anomalies. The onboard planner will accept as inputs both science and engineering goals and ensure high-level goal-oriented behavior for the constellation.

The robust execution system (SCL) accepts the CASPER-derived plan as an input and expands the plan into low-level commands. SCL monitors the execution of the plan and has the flexibility and knowledge to perform event-driven commanding to enable local improvements in execution as well as local responses to anomalies. The ObjectAgent and TeamAgent cluster management software manages the maneuver planning and execution for the constellation. This software accepts high-level constellation formation goals from CASPER and plans and executes these formations to support science (e.g. radar imaging) and engineering (e.g. downlink) activities.

One of the ASE demonstration scenarios involves monitoring of lava flows in Hawaii. SIR-C radar data have been used in ground-based analysis to study this phenomenon. The ASE concept would be applied as follows:

(1) Initially, ASE has a list of science targets to monitor.
(2) As part of normal operations, CASPER generates a plan to monitor the targets on this list by periodically imaging them with the radar.
(3) During such a plan, a spacecraft images the volcano with its radar.
(4) Onboard, a reflectivity image is formed.
(5) The *Onboard Science Software* compares the new image with a previous image and detects that the lava field has changed due to new flow. Based on this change the science software generates a goal to acquire a new high-resolution image of an area centered on the new flow.
(6) The addition of this goal to the current goal set triggers CASPER to modify the current operations plan to include numerous new activities in order to enable the new science observation. During this process CASPER interacts with ObjectAgent to plan required slews and/or maneuvers.
(7) SCL executes this plan in conjunction with ObjectAgent and TeamAgent, which execute maneuvers planned by CASPER and requested at run-time by SCL.
(8) Based on the science priority, imagery of identified "new flow" areas are downlinked. This science priority could have been determined at the original event detection or based on subsequent onboard science analysis of the new image.

As demonstrated by this scenario, onboard science processing and spacecraft autonomy enable focusing of mission resources onto science events so that the most interesting science data are downlinked. In this case, a large number of high priority science targets can be monitored and only the most interesting science data (during times of change and focused on the areas of change) need be downlinked.

The ASE concept has been selected for flight on the TechSat-21 mission and the necessary software is currently being matured and brought into flight readiness. Key TechSat-21 design reviews occur in Spring 2001 to Spring 2002, with final delivery of the spacecraft and software in September 2003. The nominal launch date is September 2004. The NASA New Millennium Space Technology 6 Project has selected the ASE concept for flight demonstration.

2 Onboard Science

There are two components of the onboard science software, the *image formation module* and the *onboard science algorithms*. The image formation module forms a (possibly reduced resolution) SAR image onboard the spacecraft from the raw phase history (demodulated I and Q returns). In the ASE mission concept, we only need to form a few images per orbit cycle (in contrast to a global mapping mission such as Magellan); hence, the necessary processing can be carried out onboard. Our baseline calculations estimate that forming a 15 km diameter spot size (dependent upon grazing angle) 10-meter by 10-meter resolution image can be formed in at best 18 seconds (with full processor utilization). A 2-meter resolution would require approximately 45 minutes. Both these timescales are considerably less than the 90-minute orbit decision cycle for downlink.

Once the image has been formed, the *onboard science algorithms* can then analyze the SAR image(s) to create derived science products and detect trigger conditions, such as changes relative to a previous orbit cycle. For example, fresh lava and old lava have very different backscatter properties; thus, new lava flows can be easily detected and localized. Likewise, water has very different backscatter characteristics than soil, enabling the detection of flooding.

We are currently investigating several methods of converting images into derived science products. The derived products will in effect be summarizations that are significantly more compact than the raw image (or phase history) data. Intensity and texture-based segmentation (in common use for ground-based processing) will be evaluated for effectiveness in generating terrain boundary descriptions and region summarizations (e.g., a flooded region will be described by an average radar cross-section and a polyline outlining its boundary). Statistical pattern recognition techniques [1] [3] will be used to identify specific types of features such as volcanoes, lakes, and iceberg fragments. Figure 2 shows results from a prototype lava cone recognition algorithm under development for ASC flight. Output from such a module could be used to downlink higher resolution data around items of interest or by downlinking a summary catalog of the interesting features. Recently developed discovery techniques [2] will also be applied to identify "interesting" regions that differ from their local background, leading to a compact description of an image in terms of subimage patches and locations.

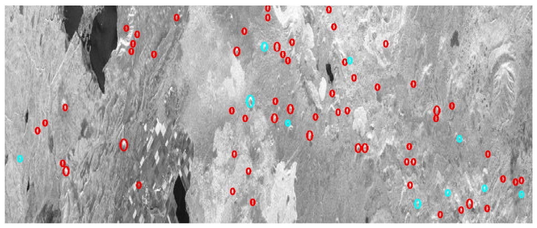


Figure 2 - Automatically identified lava cones in X-SAR image of Lava Beds National Monument, CA

In addition to calculations based on a single image, the onboard science analysis software will include change detection algorithms that compare images of the same region taken at different times. The change detection capability is particularly relevant for capture of short-term events at fine time-scale resolutions without overwhelming onboard caching systems, and also for compressing long-term "monitoring" observations in which changes are infrequent. For space science missions, example applications include tracking atmospheric changes on Jupiter, Neptune, or Triton (from optical image data), tracking ice plate movement on Europa, monitoring known (and identifying new) volcanoes on Io, capturing fine time-scale events such as jet formation on comets or phase transitions in ring systems, and detecting new cratering on planets and moons.

To detect change, we will test for statistically significant differences in derived descriptors such as region sizes, locations, boundaries, and histograms, as well as in the raw pixel data. The latter case is complicated by the need to ensure that two images are approximately co-registered. The orbit repeatability and small absolute positional uncertainty of the TechSat-21 group will help insure approximate co-registration. Also, since the magnitude of change necessary to initiate a trigger event can be specified as a parameter, some degree of robustness to image misalignment will be built-in. For change detection, radar observations have the advantage that the illumination, target, and receiver geometry remains basically the same from pass to pass. (In optical imagery, irrelevant change caused by sun position complicates the change detection problem). Figure 3 contains successive X-SAR radar images indicating lava flow on Kilauea Volcano, Big Island, Hawaii. The changes in the highlighted areas of the image are indicative of lava flow that occurred in between images. This is the type of change detection that our algorithms will perform onboard TechSat-21.

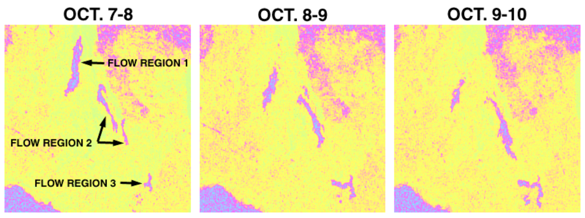


Figure 3 - Hawaii Lava Flows

All of the algorithms described scale linearly in the number of image pixels. Hence image resolution can be selected appropriately to ensure that computational and memory requirements fit within the onboard processing capabilities. For example, a previous study of the recognition algorithm in [1] indicates complexity on the order of 250 operations per pixel to reliably detect a particular type of Venus small shield volcano in the Magellan SAR data. Using this figure as a baseline, we could process approximately 10^5 pixels per second on the PowerPC 750 flight processor.

Detection of the image- and change-based triggers described here will enable a range of automated spacecraft reactions. On the conservative end of the spectrum, triggers can be used to prioritize data for downlink. For example, regions in which change was detected may be down linked first (with TechSat-21, it will take a full four days to downlink the entire onboard cache of the three spacecraft). Early access to the "interesting data" would be especially valuable to the project scientist, potentially enabling a request for modification of the original observation plan.

A slightly more aggressive use of the trigger information involves actually "discarding data". For example, if nothing significant has changed in a region, that region may be excluded from the downlink. Although a scientist would never like to discard data, the realities of a finite onboard cache and constrained downlink bandwidth will sometimes force a discard to satisfy a primary mission objective. For example, if the science goal is to capture the fine temporal details of jet formation on a comet, the onboard cache will quickly overflow unless older data that doesn't contain the desired event is discarded or degraded to lower resolution.

A third, more aggressive, but potentially extremely rewarding use of the trigger information that we will demonstrate onboard TechSat-21 is to autonomously retarget observations. For example, if an image indicates flooding in a region, subsequent orbits will employ the planner to close the control and decision loop onboard and use a modified radar aim-point in an attempt to capture the full scope of the flooding. Similarly, since many geological features are spatially clustered (e.g., volcano fields, hydrothermal vents), detection of some features will be used to seed a broad area search (e.g., using the three spacecraft radars in a coordinated effort to look in the surrounding area for additional instances).

3 Robust Execution

TechSat-21 will fly The Spacecraft Command Language (SCL) [7] to provide robust execution. SCL is a software package that integrates procedural programming with a real-time, forward-chaining, rule-based system. A publish/subscribe software bus allows the distribution of notification and request messages to integrate SCL with other onboard software. This enables either loose or tight coupling between SCL and other flight software as appropriate. Dynamic messages are supported to allow for future growth as ever-smarter software agents are added to the constellation in different satellites.

The SCL "smart" executive supports the command and control function. Users can define scripts in an English-like manner. Compiled on the ground, those scripts can be dynamically loaded onboard and executed at an absolute or delta time. Ground-based absolute time script scheduling is equivalent to the traditional procedural approach to spacecraft operations based on time. In the ASE concept, SCL scripts will also be planned and scheduled by the CASPER onboard planner. The science processing agents, cluster management software, and SCL work in a cooperative manner to generate new goals for CASPER. These goals are sent within the messaging system.

Spacecraft telemetry from all satellites is gathered onboard and fed into the integrated expert system. Significant change in the data will trigger user-defined rules. Those rules can be used for fault detection, isolation and recovery (FDIR). In that case, rules can be used to execute recovery scripts. Another application of rules is for mission constraint checking to prevent operator errors or, more simply, to provide for command pre-processing.

SCL is a mature software product, and has successfully flown on Clementine I, and ROMPS. SCL has also been used in a wide range of ground-based control and operations contexts. As such it represents a good basis for integrating the multiple ASE autonomy functions: onboard science, planning, and constellation management.

4 Onboard Mission Planning

Traditionally, the majority of planning and scheduling research has focused on a batch formulation of the problem. In this approach, when addressing an ongoing planning problem, time is divided up into a number of planning horizons, each of which lasts for a significant period of time. When one nears the end of the current horizon, one projects what the state will be at the end of the execution of the current plan. See Figure 4. The planner is invoked with a new set of goals for the new horizon and the expected initial state for the new horizon; the planner generates a plan for the new horizon. As an example of this approach, the Remote Agent Experiment operated in this fashion [8].

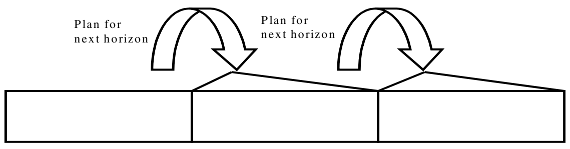


Figure 4: Traditional Batch Plan then Execute Cycle

This approach has a number of drawbacks. In the batch oriented mode, typically planning is considered an off-line process, which requires considerable computational effort, and there is a significant delay from the time the planner is invoked to the time that the planner produces a new plan. If a negative event occurs (e.g., a plan failure), the response time until a new plan is generated may be significant. During this period the system being controlled must be operated appropriately without planner guidance. If a positive event occurs (e.g., a fortuitous opportunity, such as activities finishing early), again the response time may be significant. If the opportunity is short lived, the system must be able to take advantage of such opportunities without a new plan (because of the delay in generating a new plan). Finally, because the planning process may need to be initiated significantly before the end of the current planning horizon, it may be difficult to project what the state will be when the current plan execution is complete. If the projection is wrong the plan may have difficulty.

To achieve a higher level of responsiveness in a *dynamic planning* situation, we utilize a *continuous planning* approach and have implemented a system called CASPER (for Continuous Activity Scheduling Planning Execution and Replanning) [4]. Rather than considering planning a batch process in which a planner is presented with goals and an initial state, the planner has a current goal set, a plan, a current state, and a model of the expected future state. At any time an incremental update to the goals, current state, or planning horizon (at much smaller time increments than batch planning)[1] may update the current state of the plan and thereby invoke the planner process. Such an update may be an unexpected event or simply time progressing forward.

[1] For the spacecraft control domain we are envisioning an update rate on the order of tens of seconds real time.

The planner is then responsible for maintaining a consistent, satisfying plan with the most current information. This current plan and projection is the planner's estimation as to what it expects to happen in the world if things go as expected. However, since things rarely go exactly as expected, the planner stands ready to continually modify the plan. From the point of view of the planner, in each cycle the following occurs:

- Changes to the goals and the initial state are first posted to the plan
- Effects of these changes are propagated through the current plan projections (including conflict identification)
- Plan repair algorithms [5] are invoked to remove conflicts and make the plan appropriate for the current state and goals

This approach is shown in Figure 5. At each step, the plan is created by using incremental replanning from:

- The portion of the old plan for the current planning horizon;
- The change (Δ) in the goals relevant for the new planning horizon;
- The change (Δ) in the state; and
- The new (extended) planning horizon

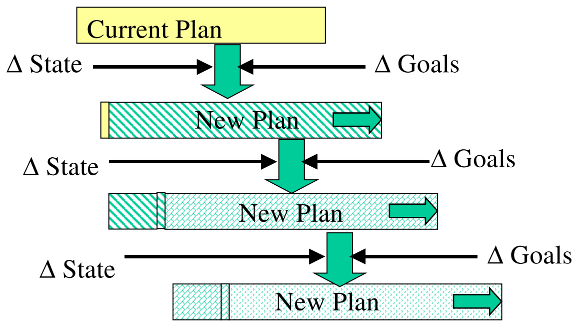


Figure 5 - Continuous Planning Incremental Plan Extension

In the ASE concept, CASPER is responsible for long-term mission planning in response to both science goals derived onboard as well as anomalies. In this role, CASPER must plan and schedule activities to achieve science and engineering goals while respecting resource and other spacecraft and constellation operations constraints. For example, when change is detected in an image, CASPER plans a response. If it is appropriate to take a more detailed image of the change area, CASPER will modify the operations plan to include the necessary activities to re-image. If this includes changing the formation of the constellation, the cluster manager will be consulted. Other required activities, such as calibration of the radar, acquisition of the image, and subsequent science processing are all planned by CASPER.

5 Cluster Management

Several new missions being proposed by NASA call for constellations or fleets of autonomous spacecraft working together to accomplish complex mission objectives. Some of the many advantages of using distributed satellite systems include greater performance, lower cost, and improved fault tolerance, ability to reconfigure, and ability to upgrade. However, coordinating the activities of all the satellites in a constellation is a challenging task. Princeton Satellite Systems (PSS) is developing the ObjectAgent (OA) and TeamAgent (TA) systems to create an easy to use agent-based software architecture for autonomous, distributed systems.

In ObjectAgent, control systems are decomposed into agents, each of which is a multi-threaded process. Agents are used to implement all of the software-based functionality and communicate via a flexible messaging architecture. Each message has a content field written in natural language that is used to identify the purpose of the message and its contents. Agents may be loaded at any time and can be configured when launched, which simplifies the process of updating flight software and removes the complexity associated with software patches. Agents will automatically seek out other agents who can provide them with the inputs they need. Decision-making and fault detection and recovery capabilities are also built-in at all levels. ObjectAgent also provides a graphical user interface (GUI)-based development environment for the design and simulation of multi-agent systems. This design environment simplifies the agent creation process and provides a common interface to a number of advanced control and estimation techniques.

The TeamAgent system applies ObjectAgent to the problem of controlling multiple cooperative satellites. TeamAgent enables agent-based multi-satellite systems to fulfill complex mission objectives by autonomously making high- and low-level decisions based on the information available to any and/or all agents in the satellite system. The required spacecraft functions for the multiple spacecraft missions have been identified and the use of software agents and multi-agent based organizations to satisfy these functions have been demonstrated [6]. Simulations of multi-agent systems for multiple satellites have been developed using TeamAgent to illustrate collision avoidance and reconfiguration for a four-satellite constellation. Agents were used to monitor for collisions, reconfigure the fleet, optimize fuel usage across the cluster during reconfiguration, and develop a fuel-optimal maneuver for reconfiguration.

ObjectAgent/TeamAgent is scheduled to fly on the Air Force's TechSat-21 mission involving three satellites flying in formation and acting as a "virtual" satellite. ObjectAgent/TeamAgent will be used to build two elements of the flight software, the Cluster Manager and the Spacecraft Manager. The Cluster Manager is designed to perform relative control of the satellites in the cluster. This control task includes relative station keeping, estimation of the cluster center-of-mass, and estimation of the relative positions of each satellite. This also includes cluster-level commanding, health summarization, and fault detection as relating to the attitude and orbit of the spacecraft and constellation. The Spacecraft Manager performs many of the functions that would normally reside on the ground including spacecraft-level fault detection. The Spacecraft Manager manages at a level above the spacecraft flight software, while the Cluster Manager manages the three Spacecraft Managers.

These ObjectAgent and TeamAgent functions encompass both plan time and execution time capabilities. At plan time, CASPER consults OA and TA on the feasibility and resource requirements to perform formation changes, maneuvers, and slews. At execution time, formation changes, maneuvers, and slews planned by CASPER and requested by SCL are performed by OA and TA. In this execution time function, OA and TA perform closed loop control in their use of lower-level attitude and control software to achieve the desired goals.

Current efforts involve transitioning this demonstrated software from workstations to the TechSat-21 flight software environment (OSE Operating system on PowerPC 750s). The OSE operating system is a message passing OS designed for distributed architectures and thus facilitates the distributed agent-based architecture.

6 Related Work and Conclusions

In 1999, the Remote Agent experiment (RAX) [11] executed for several days onboard the NASA Deep Space One mission. RAX demonstrated a batch onboard planning capability but did not demonstrate onboard science. RAX also included the Livingstone mode identification and fault recovery software [14].

PROBA [13] is a European Space Agency (ESA) mission that will be demonstrating onboard autonomy. PROBA will be launching in 2001.

The Three Corner Sat (3CS) University Nanosat mission will be using the CASPER onboard planning software integrated with the SCL ground and flight execution software [6]. The 3CS mission is scheduled for launch in late 2002. 3CS will use onboard science data validation, replanning, robust execution, and multiple model-based anomaly detection. The 3CS mission is considerably less complex than TechSat-21 but still represents an important step in the integration and flight demonstration of onboard autonomy software.

ASE will fly on the TechSat-21 mission and demonstrate an integrated autonomous mission using onboard science analysis, replanning, robust execution, model-based estimation and control, and formation flying. ASE will perform intelligent science data selection that will lead to a reduction in data downlink. In addition, ASE will increase science return through autonomous retargeting. Demonstration of these capabilities in onboard the TechSat-21 constellation mission will enable radically different missions with significant onboard decision-making leading to the pursuit of novel science opportunities. The paradigm shift toward highly autonomous spacecraft will enable future NASA missions to achieve significantly greater science returns with reduced risk and cost.

Acknowledgement

Portions of this work were performed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

We also wish to acknowledge the contributions of the Air Force Research Laboratory, Interface and Control Systems, Princeton Satellite Systems, the Massachusetts Institute of Technology, Arizona State University and the University of Arizona in conceiving and supporting the Autonomous Sciencecraft Experiment.

References

[1] M.C. Burl, L. Asker, P. Smyth, U. Fayyad, P. Perona, J. Aubele, and L. Crumpler, "Learning to Recognize Volcanoes on Venus," *Machine Learning Journal,* April 1998.

[2] M.C. Burl and D. Lucchetti, "Autonomous Visual Discovery", *SPIE Aerosense Conference on Data Mining and Knowledge Discovery,* (Orlando, FL), April 2000.

[3] M.C. Burl, W.J. Merline, E.B. Bierhaus, W. Colwell, C.R. Chapman, "Automated Detection of Craters and Other Geological Features", *Intl Symp Artificial Intelligence Robotics & Automation in Space*, Montreal, Canada, June 2001

[4] S. Chien, G. Rabideau, R. Knight, R. Sherwood, B. Engelhardt, D. Mutz, T. Estlin, B. Smith, F. Fisher, T. Barrett, G. Stebbins, D. Tran, "ASPEN - Automating Space Mission Operations using Automated Planning and Scheduling," *SpaceOps*, Toulouse, France, June 2000.

[5] S. Chien, R. Knight, A. Stechert, R. Sherwood, and G. Rabideau, "Using Iterative Repair to Improve Responsiveness of Planning and Scheduling," *Proc Fifth International Conference on Artificial Intelligence Planning and Scheduling,* Breckenridge, CO, April 2000.

[6] S. Chien, B. Engelhardt, R. Knight, G. Rabideau, R. Sherwood, E. Hansen, A. Ortiviz, C. Wilklow, S. Wichman " Onboard Autonomy on the Three Corner Sat Mission," *Intl Symposium on Artificial Intelligence Robotics and Automation in Space*, Montreal, Canada, June 2001

[7] Interface & Control Systems, Spacecraft Command Language, http://www.sclrules.com.

[8] A. Jonsson, P. Morris, N. Muscettola, K. Rajan, and B. Smith, "Planning in Interplanetary Space: Theory and Practice," *Procs Fifth International Conference on Artificial Intelligence Planning Systems, Breckenridge*, CO, April 2000.

[9] J. Kurien and P.P. Nayak, "Back to the Future for Consistency-based Trajectory Tracking," *Proc. National Conference on Artificial Intelligence*, Austin, TX, 2000.

[10] G. Rabideau, R. Knight, S. Chien, A. Fukunaga, A. Govindjee, "Iterative Repair Planning for Spacecraft Operations in the ASPEN System," *International Symposium on Artificial Intelligence Robotics and Automation in Space*, Noordwijk, The Netherlands, June 1999.

[11] rax.arc.nasa.gov, Remote Agent Experiment Home Page.

[12] D. Surka, J. Mueller, M. Brito, B. Urea, M Campbell, “Agent-based Control of Multiple Satellite Formation Flying,” *Intl Symposium on Artificial Intelligence Robotics and Automation in Space*, Montreal, Canada, June 2001

[13] The PROBA Onboard Autonomy Platform, http://www.estec.esa.nl/proba/

[14] B.C. Williams and P.P. Nayak, “A Model-based Approach to Reactive Self-Configuring Systems,” *Proceedings of the National Conference on Artificial Intelligence*, Portland, Oregon, 1996.

Genetic Algorithms In Aerospace Design: Substantial Progress, Tremendous Potential

Murray B. Anderson
Sverdrup Technology Inc./TEAS Group
308 West D. Avenue
Building 260
Eglin Air Force Base, FL 32542, USA

Executive Summary

This paper was written to provide an overview of the potential of using genetic algorithms (GA's) for aerospace design or to support aerospace design efforts. Genetic algorithms can design individual vehicle components, like the external aerodynamic shape, the internal supporting structure, the propulsion system, the autopilot, or just about any other system component that has requirements that can be mathematically represented and modeled. Genetic algorithms can also be used to support design studies through their ability to find optima for complicated multi-dimensional optimization topologies. This paper, which overviews multiple recent aerospace design efforts using genetic algorithms, was an invited paper presented at a joint NATO/Von-Karmen-Institute Workshop on Intelligent System held in Brussels, Belgium during May of 2002.

Abstract

Since the mid to late 1980's, Genetic Algorithms have been increasingly applied to aerospace problems, producing some exciting results. Almost every discipline in aerospace, from Guidance, Navigation, and Control to Propulsion and Structures, has yielded itself to the power of genetic algorithms. The shear volume of applications that GA's have been applied to has grown tremendously over the past decade. This growth can, to a large degree, be attributed to a growth in computer speed coupled with a growth in the understanding that advanced computing techniques can often produce designs vastly superior to those possible by trial and error or by experience alone. As the design and analysis space grows more non-linear for advanced systems, engineers and their managers are beginning to realize the potential of using a non-linear design and analysis tool to help produce better products.

Publications Analysis by Discipline

A quick survey of papers published in recent years that have used genetic algorithms as the key design/analysis tool is quite revealing. Figure 1 shows how the publication distribution has varied through the years for AIAA conference papers. It is interesting to note the categories that have received the most attention in recent years: multidisciplinary design optimization, aerodynamics, propulsion, and structures/structural-dynamics. Also interesting is the category called GA's. These publications have applied GA's to aerospace problems, plus they investigate different modes of GA behavior based on population sizing, crossover techniques, mutation levels, etc., to study how best to apply GA's to particular problems.

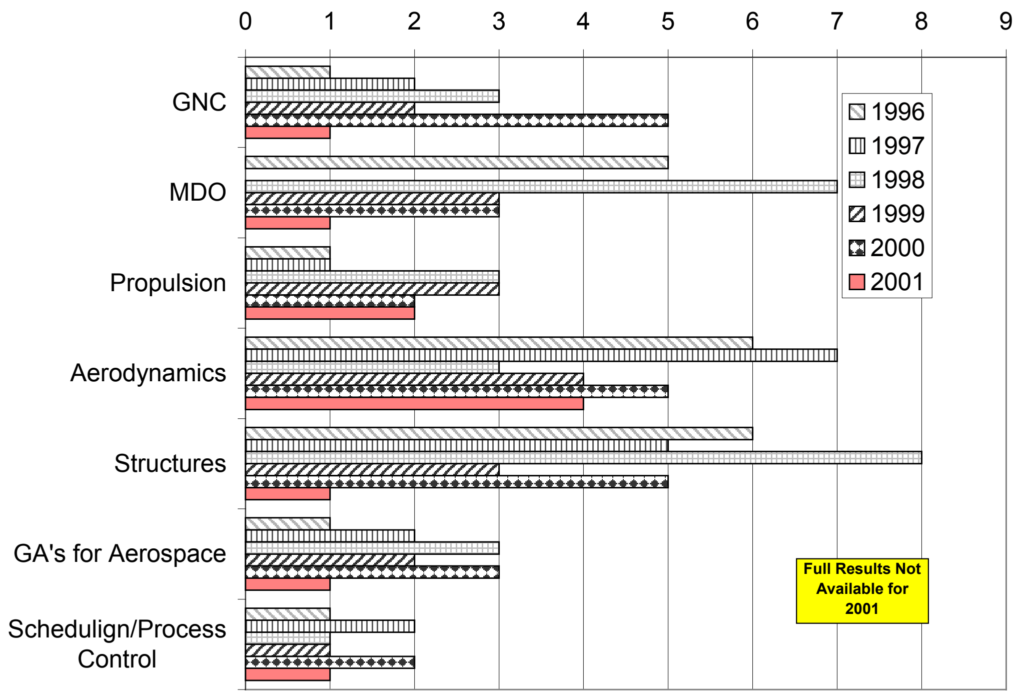


Figure 1. Trends in AIAA Conference Paper Publications

In terms of total numbers of publications within AIAA conferences, Figure 2 shows an interesting trend. Even number years have drawn considerably more publications than odd-number years. The reason for this trend is not clear. The year 1998 produced the most GA-Aerospace publications with 34. There is no clear growth trend in the total number of GA-related conference papers for AIAA alone during the 6-year time period shown in the figure. As the raw numbers show, there are not enough AIAA papers to support an independent GA Conference each year. Right now the GA applications are spread out among up to 5 or 6 different AIAA conferences each year, making it challenging to keep track of all the GA applications. By looking at the trends, there probably are enough papers to support a bi-annual AIAA GA conference similar to the International Conference on Genetic Algorithms, but such a conference would require significant coordination and might actually hinder the spread of GA's. By being presented as a "new" application at the individual discipline-specific conferences, GA's are given a good exposure across the breadth of the aerospace community.

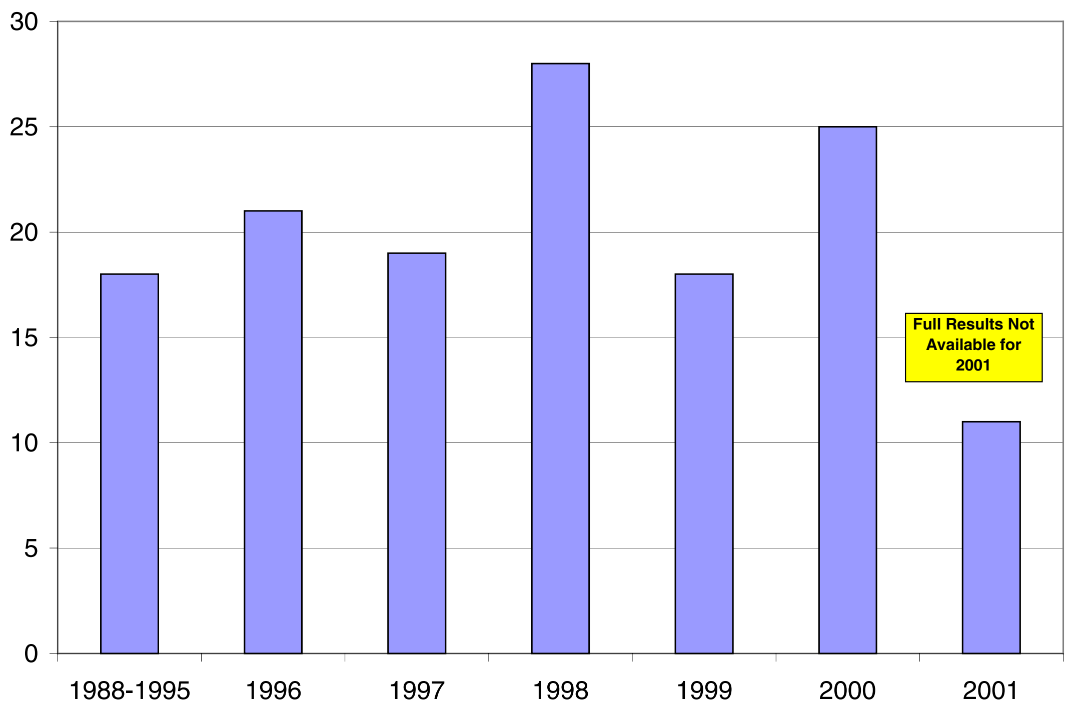


Figure 2. Total AIAA Conference Paper Publications

When a more broad inclusion of publications is considered, including aerospace-related GA publications in journals from ASME, AHS, or IEEE for example, it is interesting that GNC, aerodynamics, and structures/structural dynamics are still very dominant, but MDO is not (see Figure 3). Scheduling/Process-Control emerges as a very important area for GA applications. MEMS, communications, heat transfer, damage control, and acoustics all emerge during certain years, but do not have a steady publication record in general.

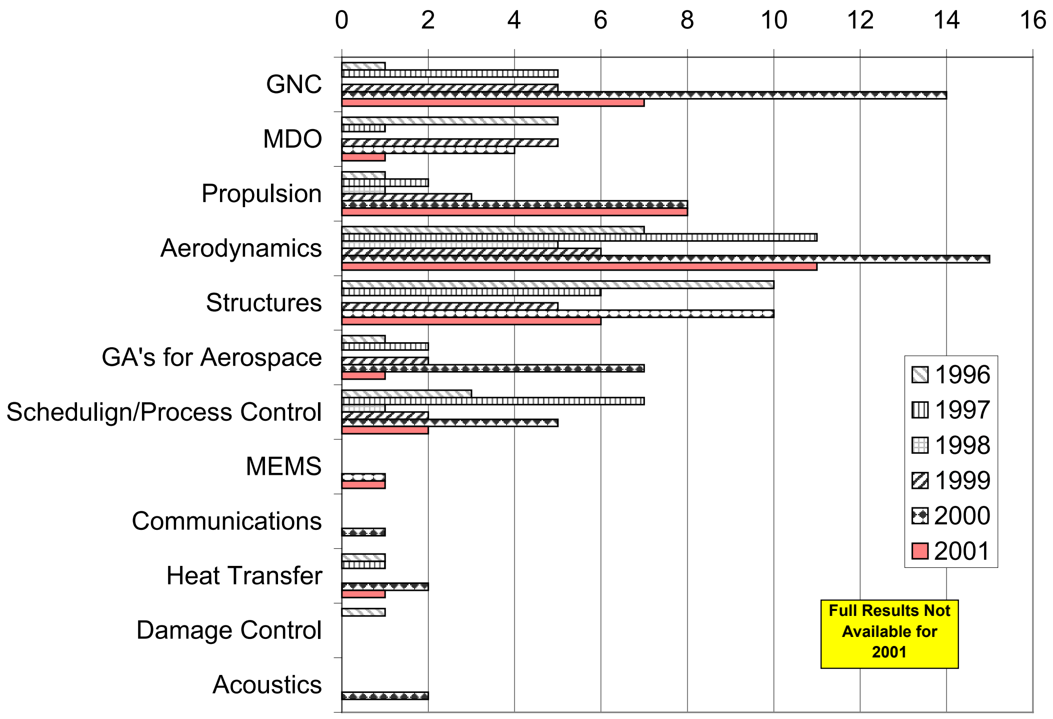


Figure 3. Non-AIAA Publications with Aerospace Applications

Figure 4 shows how AIAA contributes to the total number of GA-aerospace publications each year. In general, AIAA is generally a significant proportion of the total publications, but some years AIAA is less than ½ the total publications. This fact makes it difficult to stay abreast of all the GA publications that occur each year. There is no "one place" to look for GA-aerospace applications. The year 2000 was a significant year for GA-aerospace application publications, where the total number neared 100. If the year 2001 is omitted from consideration, there would seem to be a fairly consistent upward trend in the number of GA-aerospace publications. It is very likely that all of 2001's publications are not yet available on the internet, so the number of publications for 2001 could actually be higher than what is shown in the figure.

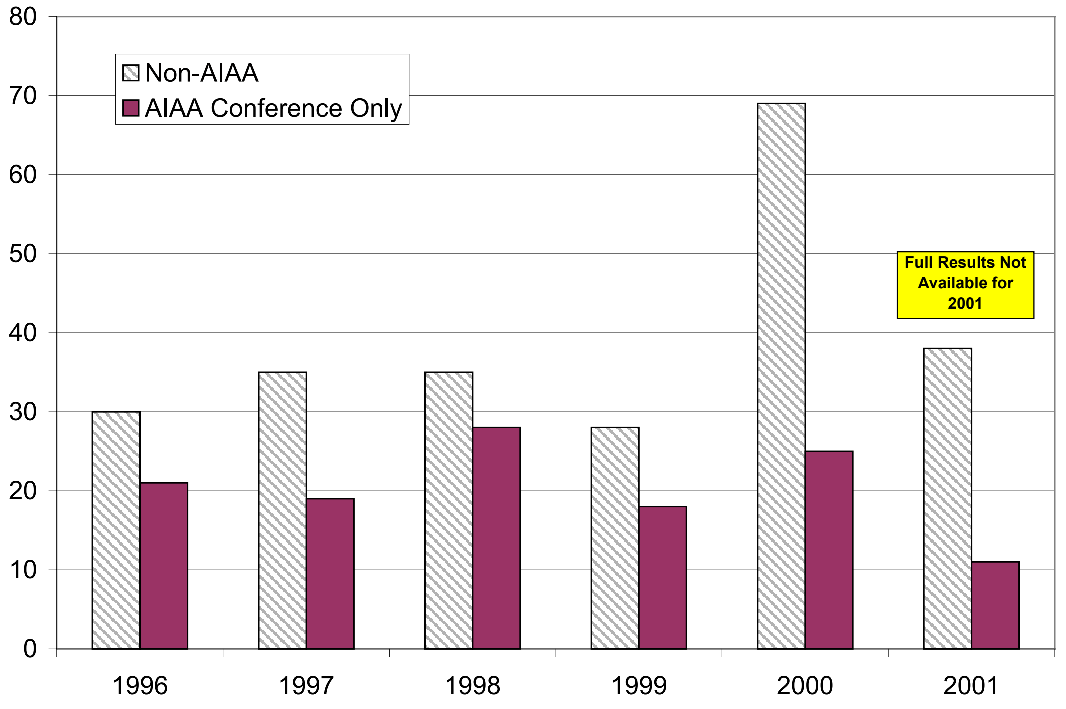


Figure 4. Comparison of Total AIAA and Non-AIAA Publications

With all these publications, it is impossible to list or even discuss each one. But in order to give a sampling of the types of design studies encompassed in all these publications, a few papers in the most prominent technical areas are highlighted to show what is possible when a GA is used to help engineering design and analysis.

Genetic Algorithms in Guidance, Navigation, and Control

Guidance, Navigation, and Control (GNC) was one of the first technical areas in aerospace engineering exploited by GA's. For a time, David Goldberg[1], author of the most popular text on GA's, was at the University of Alabama in the Aerospace Engineering Department. K. Krishnakumar, the organizer of this NATO/VKI workshop was also a professor there at the same time. Together, they published some of the first papers on using GA's for Aerospace applications. GNC was one of their first application areas. One 1990 conference paper, later a Journal of Guidance, Control, and Dynamics paper[2], showed how a GA can design a lateral autopilot and a windshear controller. In the words of the authors:

> *"The results show that a variety of aerospace control system optimization problems can be addressed using genetic algorithms with no special problem-dependent modifications."*

Certainly the authors were correct that a variety of control problems can be solved with GA's. But whether the authors could have forseen the growth of GA applications beyond control systems is unknown. Besides the autopilot/controller applications envisioned by the authors within the GNC field, GA's have also been used for mechanical control system component studies. For example, Rogers[3] has used a GA to help select the right actuator for a given problem.

Air data systems also play a significant roll in vehicle control. One of the earliest GA applications to air data systems was by Deshpande, Kumar, Seywald and Siemers[4] and later by Anderson and Lawrence and Lopez[5]. In each of these papers a GA was used to design a working array of flush-orifice air data ports to provide accurate estimates of vehicle flight conditions.

Norris and Crossley[6] recently used a genetic algorithm to find gains to control a standard pedagogical two-disk torsional spring system. The approach used a very simple two-loop proportional-integral control system with velocity feedback. The two-loop controller had three gains that needed to be determined as a function of variable spring stiffness. The objective functions were defined such that good gain values produce little error between the commanded and achieved disk rotation angles. Since two separate disks were being commanded in twist, a pareto genetic algorithm was used to try to minimize the rotational errors in both disks simultaneously. At the conclusion of 80 generations, the resulting family (i.e. population of 80 members) of three controller gains were hybrid performers that would work reasonably well for both disks. Two points worth noting from this study was that the crossover probability was rather low (50 percent) and the population size was three times the number of bits required to represent the three gains (12 bits per gain, 36 bits total).

Aerospace control system publications have also led to GA’s being applied in other fields. For example, another interesting genetic algorithm controller application was recently presented by McGookin[7] to control the steering of oil tankers to multiple waypoints in a narrow channel. In McGookin's work the control system consisted of a two-loop autopilot and a sliding mode controller (SMC). The four parameters being optimized consisted of the 1st and 2nd heading loop poles (frequency plane), a heading switch gain, and a so-called heading boundary layer thickness. The goal (e.g. objective function) of the study was to find values for these four parameters such that the oil tanker passed within an acceptable distance of each waypoint while minimizing rudder movement. Minimizing rudder movement saves fuel and time. The results shown indicate that the genetic algorithm found excellent values of the four "gains" in 100 generations. Rudder movement was minimal and each waypoint was reached with very little error. In terms of the genetic algorithm operation in this study, it is interesting that McGookin used a 5 percent mutation rate, which is at least an order of magnitude higher than conventional values of this parameter. Since there is no agreement among genetic algorithm researchers about value ranges for crossover and mutation, it is interesting to note what values other researchers use for their applications.

Another study by Martin[8] addressed the issue of autopilot gain scheduling by using genetic algorithms to replace the ad hoc design process typical in linear gain scheduling with a genetically-fit hyperplane-surface strategy. The genetic algorithm was basically used to optimally design the gain schedule. Of course a gain scheduling approach that might work for one scenario could be inadequate without adaptation. Adaptation strategies have been pursued by Karr and Harper[9] using genetic algorithms to augment fuzzy logic controllers. Coupling genetic algorithms with neural networks appears to offer improvement to adaptive control that neither approach has independently. The fuzzy controller (neural network) uses a "rule-of-thumb" strategy to control a chemical system, but the chemical system is periodically changed, thereby invalidating some of the rules-of-thumb. As the system changes, a learning algorithm (the genetic algorithm in this case) tests new rules-of-thumb so that the fuzzy controller can continue to control the chemical system. For autonomous systems[10] this type of approach has obvious advantages assuming the genetic algorithm can keep up with the rate of change of the system. Karr[11] later expanded this work to control the rendezvous of two spacecraft. Another excellent work in adaptive fuzzy logic controller design using genetic algorithms was done by Homaifar and McCormick[12] to control a simple electronic cart. The genetic algorithm designed both the rules-of-thumb and the membership functions for the system in an automated process that did not require human input. Other publications worth mentioning include Sweriduk, Menon, and Steinberg[13], Mondoloni[14], and Perhinschi[15].

Genetic Algorithms in Aerodynamics

In recent years researchers have applied gradient-based optimization schemes to aerodynamic design[16,17,18,19], but these methods are subject to several undesirable restrictions. First, these methods require knowledge of the aerodynamic derivatives for each parameter, or combination of parameters for higher order coupling. These derivatives may not be easily determined and must be continually recalculated as the design moves through the design space since few aerodynamic design problems have linear derivatives, especially in multiple dimensions. Second, gradient-based optimizers must start with a specified set of initial parameters, which can bias future solutions toward a local optimum in the viscinity of the starting point. Third, gradient-based methods cannot be applied to problems where there are discontinuities in the design space because the derivatives in these regions are not defined. Discontinuities are common in aircraft design. As Gage[20] points out, there are no partial engines or partial seats on an aircraft. Finally, gradient search procedures work efficiently when there are a small number of design variables and when the variables are essentially independent of each other. As the number of design variables increases and coupling of the variables occurs (most often the case for complex aerodynamic designs), gradient-based algorithms lose much of their efficiency. Only recently have attempts been made to couple artificial intelligence (i.e. learning) techniques to aerodynamic design. The research of Gage and Kroo[20] was focused on applying genetic algorithms to the topological design of nonplanar wings. In their work the wing was broken into several segments which were allowed to vary in incidence and dihedral angles. The goal of the optimization was to minimize induced drag given a fixed lift. Their approach used both a penalty and repair approach to deal with solutions not achieving the fixed lift value. Bramlette and Cusic[21] have also applied genetic methods to the parametric design of aircraft and Tong[22] has used genetic algorithms to conduct preliminary design of turbine engine airfoils after gradient methods had stalled in a local optimum. More recently Anderson[23] has applied genetic algorithms to subsonic wing design with the goal of producing good aerodynamic shapes, but also given the additional constraint that the structure must not break. Anderson used penalty weights to combine Lift, L/D ratio, and structural design margins into one global objective function. Anderson, like Gage[20], pointed out that the achieved solution is strongly dependent upon the values of the penalty weights, a very unappealing result. Later, Anderson[24] removed the weighting procedure and instead used a pareto genetic algorithm on the same problem. The results of this later work showed that pareto genetic algorithms are ideally suited for complex optimization problems with diverse goals. Such an approach does mean, however, that the designer must scan the resulting solutions in the pareto-optimal set to determine which solution or solutions are most desirable. Unlike single objective problems where there is a clear winner, multi-objective problems require judgement about which solutions are preferable. For cases where there are both aerodynamic and structural goals in the design, there is usually a willingness to trade-off some aerodynamic performance to ensure that structural integrity goals are met. Some very recent work by Sharatchandra, Sen, and Gad-el-Hak[25] shows that a genetic algorithm can be used to design micro-devices also. In this work a viscous micropump is designed to maximize the flow rate between two parallel plates encasing a rotating cylinder. The basic problem was to let the genetic algorithm determine the optimum plate spacing, the correct upper plate shape (only the lower plate was flat), and the optimal vertical location of the rotating cylinder in order to maximize the mass flow of the device at the exit plane of the two plates. The computational approach used to calculate the mass flow rate in the device was the Navier-Stokes equations. Since the Navier-Stokes equations can be rather time consuming to solve, what should be called a micro-genetic algorithm, in combination with a multi-point crossover (parameter-based) scheme and a high mutation rate, was employed in an attempt to reduce the total number of Navier-Stokes solutions attempted. The results indicated that the micro genetic algorithm with relatively few Navier-Stokes solutions (551 total) achieved good pump designs when compared to standard large population genetic algorithms. Gregg and Misegades[26] used a "parameter evolution" strategy coupled with Jamesons' FLO-22 code (full potential method) to design transonic wings in a parallel mode. By distributing individual cases to multiple processors significant turn-around time reductions were achieved. The authors noted that taking advantage of the inherent parallelization capability of population based search techniques was a strength of the approach. The "parameter evolution" strategy is similar to simulated annealing and relies more heavily on random parameter variations than does the survival-of-the-fittest strategy embedded in a genetic algorithm. Both Sharatchandra[25] and Gregg[26] used procedures with essentially high mutation rates when compared to the standard genetic algorithm. For small populations, however, there continues to be emerging evidence that higher than normal mutation rates are justified. Other papers using GA's for aerodynamic optimization worth referencing are by Oyama, Obayashi, and Nakashi[27,28] of Tohoku University in Japan.

Another emerging trend in optimization using genetic algorithms is to couple the genetic algorithm with more traditional design techniques to speed convergence. The goal of these hybrid methods is to use the global search strength of the genetic algorithm to provide useful information (starting point) for gradient ascent/descent methods. Anderson, Lawrence, and Lopez[29] successfully coupled a genetic algorithm with a differential corrections method to find flight condition estimates (Mach, angle of attack, aerodynamic roll angle, freestream pressure) from static pressures measured around a missile forebody. The genetic algorithm provided the initial "guess" to the gradient descent method. Fast and accurate flight condition estimates were consistently obtained with this method. Selig and Coverstone-Carroll[30] used a genetic algorithm coupled with an inverse design method to design wind turbines (wind mills) which maximize power output at varying wing speeds. The design variables included blade pitch, blade chord, and blade twist distributions with span. In this case the genetic algorithm executed the design search and the inverse procedure enforced certain constraints while giving the designer flexibility in choosing which variables to iterate and which to send to the genetic algorithm. Another hybrid approach by Cao and Blom[31] coupled a genetic algorithm with a standard gradient approach to maximize lift coefficient for a multi-element airfoil. The design variables included flap angle, gap, overlap, and flap surface geometry. At some prescribed threshold lift coefficient, the genetic algorithm was halted and the gradient method was begun. This approach is very similar to the method of Anderson[29], however, it should be noted that both of these works found that selecting the proper threshold for switching algorithms is critical to the final result. GA's have recently been growing in popularity for aerodynamic design studies using Euler equations. Jang, and Lee[32] recently presented a study of transonic airfoil design using GA's coupled to Euler equations.

Genetic Algorithms in Multidisciplinary Design Optimization

MDO studies have varied significantly over the past few years. Jones, Crossley, and Anastasios[33] have included both aerodynamic and acoustic optimization together. Other authors are trying to expand the use of GA's to complete systems. For example, Perez[34], et. al. has recently used a GA to conduct conceptual design studies for aircraft. Anderson, Burkhalter, and Jenkins[35,36] have performed the same type of conceptual design studies for missiles, both ground-launched and air-launched, to defeat high speed re-entry vehicles and highly maneuverable targets. Ewing and Downs[37] have included a feature in their MDO study not commonly found so far in these types of studies, namely, financial return on investment. The financial area will probably be a growth area for these types of conceptual studies, and a defense contractor that could master providing the most cost-effective solutions to a given threat would never hurt for business. Mastering cost estimating during the design process would likely lead to higher profit margins during manufacturing, so the motivation should be there in industry to begin pursuing these types of studies.

Genetic Algorithms in Propulsion

Airbreathing propulsion seems to dominate the current applications of GA's. Chernyavsky et.al[38] recently used GA's to design a 3-D hypersonic inlet. Torella and Blasi[39] recently published work on using GA's to design complete gas turbine engines. Uelschen and Lawerenz[40] were recently focused on the design of axial compressor airfoils using a combination of artificial neural networks and GA's. Schoonover, Crossley and Heister[41] recently used GA's to design hybrid rocket motors. Given the growing interest in hybrids, this type of application is expected to grow. On the solid rocket motor front, finding the optimal design of a solid rocket motor to meet certain criterion has been the subject of some limited research since the 1960's, although the 1970's saw some developments in capability with the advent of very capable mainframe computers. In 1980, Sforzini[42] commented in his own early solid rocket optimization paper that despite the tractable mathematics involved in designing a solid rocket motor, "limited treatment of this subject appears in the literature". Twenty-two years later, Sforzini's statement is still true to a great extent. The preferred optimization technique still relies heavily on experience and manual "hunt and peck" techniques to yield desirable rocket motor designs. Rigorous computer based optimization techniques have still not been able to penetrate the solid rocket design process in any more than just a few limited studies and publications. But given the vast increases in computing power over the past five years, it is worthwhile to apply computer-based optimization algorithms to preliminary design of solid rocket motor grains.

One of the first attempts at using an automated procedure to design a solid rocket motor was by Billheimer[43] in 1968. Although the physical modeling used in this study was limited by the computational resources available at the time, this paper really acknowledged the importance of automating the design process for solid rocket motors, and demonstrated through sensitivity parameters that automation can aide in the preliminary design environment. Another early attempt at using a computer to find an optimal solid grain geometry was by Woltosz[44] in 1977. Woltosz used a pattern search technique developed by Hook and Jeeves[45] and Whitney[46] to determine five critical design dimensions which maximizes the total impulse-to-motor weight ratio while meeting a specified minimum achieved velocity for a specific vehicle. This same pattern search technique was also used by Foster and Sforzini[47] to minimize the differences between computed and desired solid rocket motor ignition characteristics based upon five primary igniter design parameters. The pattern search technique was an early precursor to the type of GA work that would come much later. GA's also work on a pattern search basis, but are of course radically different.

One early study that was not based on sensitivity derivatives or any calculus-based methodology was done by Swaminathan and Madhavan[48]. A direct random search technique, similar to simulated annealing, was the method that was used to determine the "optimum" composition of propellant ingredients (such as binder, oxidizer, and additives) to generate the highest possible specific impulse.

Other gradient-based techniques have been used as recently as 1992 by Peretz[49] to look at extending the range of air-to-air missiles through a two-pulse thrust profile. A quasi-Newton algorithm was used to define thrust profiles that would allow the missile to enter a glide phase of flight to extend range prior to re-acceleration of the missile for intercept. One interesting conclusion of this study was that zero lift drag and minimum glide velocity had a much larger impact on eventual range than did conventional rocket motor design parameters such as specific impulse and mass fraction.

Two of the more modern approaches that have demonstrated applicability to solid rocket motor design were done by Clergen[50] and McCain[51]. The systems developed in these studies can be classified as soft-computing techniques. Clergen developed a computerized expert system with a hypertext user interface to aid designers in selecting preliminary designs based on past experience (i.e. the experts). With this system the designer can select, with a user-friendly interface, design criteria such as minimum motor mass and obtain preliminary design parameters that might be suitable for the mission being planned. This system is build around a data base of known systems. McCain's system is also an expert system in the sense that it is heuristic. The heuristic performs an independent design variable selection, and these design variables are then passed to an existing pattern search optimization package to develop rocket performance characteristics. The heuristic then analyzes the effect of altering each independent SRM design variable and selects, based on sensitivity/partial derivatives, the variables to alter for the next design attempt. This technique is an advancement on the pattern search technique of Sforzini, but still has some of the inherent weakness of any sensitivity derivative approach.

Based on the literature review, the design of solid rocket motor is mostly based upon experience (experts and hunt-and-peck methods) or gradient/calculus-based techniques. In either case, historical knowledge and gradient methods tend to focus on, or end up with, designs that are not too dissimilar from known designs. GA's by the nature of the way they operate, can jump beyond this local knowledge (or optima) base. One recent GA application to solid rocket motor design was by Anderson, Burkhalter, and Jenkins[52,53]. The links to the genetic algorithm are all straightforward. The GA is the controlling routine which calls the rocket performance code as needed. The GA passes down the design parameters through the subroutine call statement and the rocket code passes back a measure of how well the design performed compared to a specified thrust profile. The fitness measurement is basically a root mean square error between the achieved thrust history and the specified history. Several cases are presented in the reference, but summary results for two cases are shown below. For the first case the desire was to design a large rocket motor capable of producing 30,000 lbf of thrust for 50 seconds. Figure 5 shows how well the genetic algorithm met this goal.

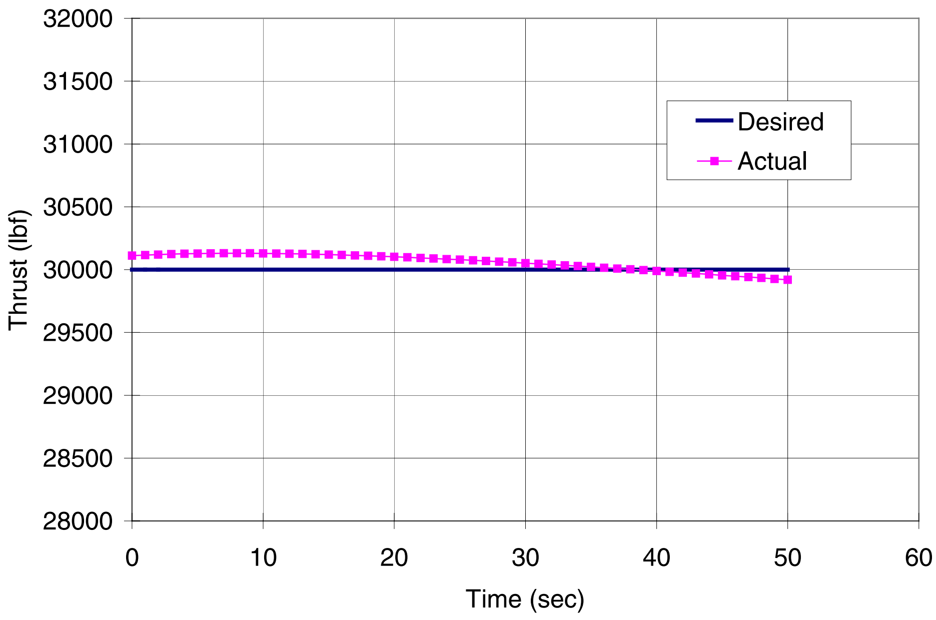


Figure 5. Genetically Engineered Constant Thrust Profile

The evolution of the grain design producing this thrust is shown in Figure 6. A five pointed star, which has a nearly constant burn area as a function of time, was quickly found by the GA.

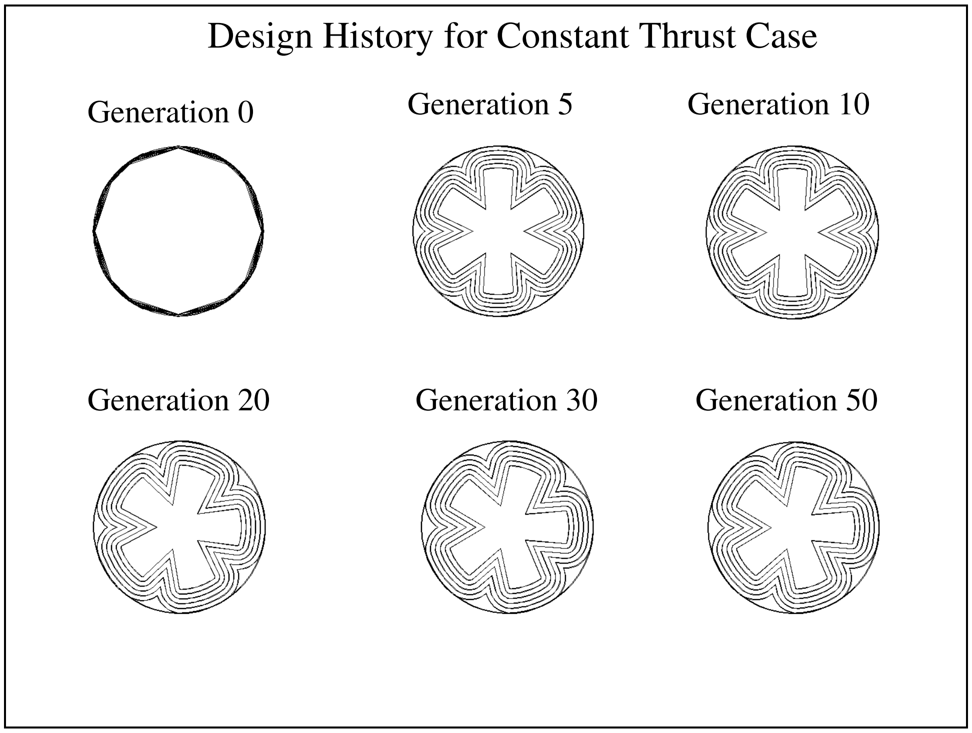


Figure 6. Evolution of Motor Design for Constant Thrust Case

Figure 7 shows an example of a regressive thrust profile for a much smaller rocket motor. As with the constant thrust case, the GA nearly perfectly matches the desired profile.

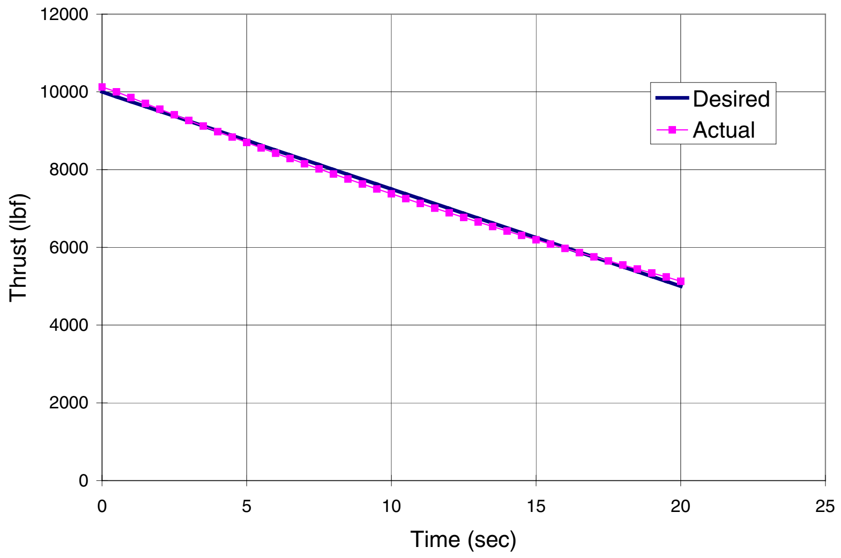


Figure 7. Genetically Engineered Regressive Thrust Profile

Figure 8 shows the evolution of the design for this case. Eight and nine pointed star grain designs were the “best” during the solution process, and the GA eventually settled on a nine-pointed star by generation 60.

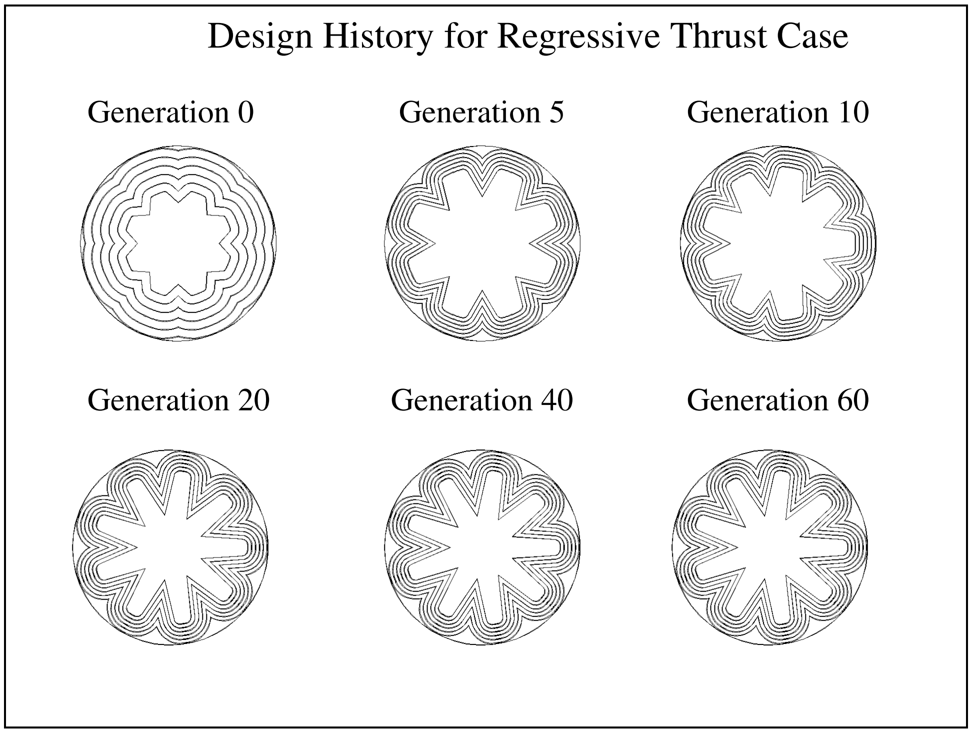


Figure 8. Evolution of Motor Design for Regressive Thrust Case

Genetic Algorithms in Structures

GA's applied to structures/structural-dynamics is one of the most popular GA applications. Composite laminate studies have been popular, and there are too many publications to mention each one, but there are some worth mentioning. Venter and Haftka[54] used a two species GA to design composite laminates subjected to uncertainty. Liu, Haftka, and Akgun also used GA's and response surface methods to design composite wing structures. Other researchers have also coupled GA's to other methods. Nair and Keane[55] coupled GA's with approximation techniques to design structures. Missoum[56] et. al have also developed a GA-based topology tool for designing continuum structures. Furuya and Lu[57] recently combined GA's and neural networks to produce optimal structural designs. On the space side, Bishop and Striz[58] used GA's to design vibration suppression systems in space structures.

Genetic Algorithms in Scheduling/Control

GA's have been applied to the problem of Air Traffic Control several times over the years. Perhaps most recently, GA's have been used to help controllers get the information they need in an optimal way to help do their jobs better. Shaviv and Grunwald[59] were responsible for this work, which focused on air traffic control displays. Communications is certainly an area where GA's have merit. Frayssinhes[60] recently showed how GA's can produce good satellite constellation geometries, which has a variety of applications beyond just communications. A couple of "directly applicable" non-aerospace applications are worth mentioning also. In one case, U.S. West[61] found that laying fiber optic cable, which is usually done based on experience and intuition, can be more efficiently done using a genetic algorithm. During the genetic algorithm process, cable networks that require more cable die and ones that require less cable survive. The end result is that network design cycle times has been cut from two months to two days, saving U.S. West $1 million to $10 million per network. Texas instruments[61], also unleashed a genetic algorithm on a computer chip design problem and the algorithm came up with a design that required 18 percent less space, using a cross connection strategy that no human had thought of. These two diverse examples show that GA's can be used to support the components that help overall aerospace systems function better and more efficiently.

Genetic Algorithms in Flight Test Data Extraction

Another growth area for GA's is extracting information from flight test data. Wollam, Kramer, and Campbell[62] recently showed that it is possible to do reverse engineering of foreign missiles given these kinds of data. Another significant growth area for GA's concerns simulation validation and verification. As test budgets shrink and the use of Modelling and Simulation grows, model verification becomes increasingly important. One example worth mentioning was recently done by Anderson[63]. In basic terms, verification of launch/jettison performance predictions requires implementation of accurate mathematical models of the weapon aerodynamics, the aircraft interference flowfield contributions, the ejector performance, the flight control system, and knowledge of the actual flight conditions at launch. The aerodynamic math models typically include coefficients or parameters whose numerical values must be determined or estimated for the various flight conditions of interest. Values for these parameters are obtained from wind tunnel tests, analytical or numerical methods, and flight test observations. Wind tunnel tests are usually conducted on scaled models of the actual vehicle of interest and in a simulated flight environment, characterized by non-dimensional parameters such as Mach number and Reynolds number for typical weapon delivery flight conditions. The extent of representation of the physical phenomena in the describing algebraic or differential equations limit analytical and numerical methods. Flight-testing provides the actual environment of interest, but aerodynamic loads are usually not measured directly but rather inferred or estimated based on measurements of the dynamic motion of the vehicle in response to either naturally or intentionally induced disturbances.

A number of parameter estimation methods, such as maximum likelihood, linear regression, and Kalman filter, among others, have been applied to flight data analyses[64,65,66]. The methods are based on the assumption that a correct math model of the aerodynamic loads acting on a vehicle will produce six degree-of-freedom (6-DOF) simulation results which closely match the observed data, with goodness of the model judged by the goodness of representation of the observed data. Consideration must be given to the fact that the observations usually contain measurement noise which cannot be exactly replicated in the simulation. The common approach is to seek to minimize in a least-squares sense the difference between the observed and predicted flight behavior at discrete times along the vehicle trajectory. Iterative procedures are usually invoked, beginning with initial estimates of the parameters, to modify the parameter estimates until convergence is achieved. Two sources of error need to be recognized in this approach; namely, (1) the modelling error in representing the fidelity of the dynamic equations governing the physical motion of the vehicle, and (2) the measurement errors associated with the observations of the vehicle motion itself. The quality of parameters extracted from any identification method can degrade in the presence of either of these error sources. In order to gain confidence in a simulation, however, it important for the simulation to be able to mimic the flight test data as closely as possible. Generally this comparison is done in the basic six degrees of freedom (yaw, pitch, roll, x, y and z). If a set of parameters cannot be found that causes the simulation to mimic the flight test data, this situation leads to an overall lack of confidence in the simulation.

For many flight vehicle configurations, a general form of the appropriate math model can be derived from knowledge of the vehicle shape and flight conditions, such as Mach number, altitude, and expected angle of attack range. The model may consist of explicit analytical functions, implicit functions requiring the solution of differential equations, or look-up tables, but will, in general, contain a finite number of parameters, the values of which need to be deduced from the measured information. With noise included in the measurements, uniqueness of the parameter set is not guaranteed for a minimum error solution, since, in general, different combinations of parameters may produce solutions having equivalent, least-squares differences relative to the data. In fact, local minima may exist in the search space, making convergence to a true minimum difficult for many traditional, gradient-based methods if the initial parameter estimates are of poor quality. A genetic algorithm has shown that it can provide good parameter estimates[67].

To better gauge how well the new post-flight analysis approach works, it will be compared directly against a "hill climbing" gradient optimization approach. Gradient methods have been the standard way to perform optimization problems such as this one for many years. However, for highly non-linear optimization problems, the advantages of this new approach will become readily apparent.

Figure 9 shows the Euler pitch angle history for the hill climber and the genetic algorithm compared to flight test data. The hill climber obviously got stuck in a local optima and did not compare nearly as well with the flight test data than the genetic algorithm. Forgetting the genetic algorithm comparison for a moment, it could be said that the hill climber captured the right "trend". Engineers often talk about trend comparisons with the actual comparisons are not as good as they would like.

The yaw angle history plot, Figure 9, shows much better performance by the hill climber, however, it is still not as good as the genetic algorithm. Both methods provide very reasonable yaw behavior for the vehicle.

Figure 11 shows the roll angle comparison. The hill climber missed the initial roll to the right and lacked overall trend performance. The genetic algorithm mimicked the flight test data very well. Even the fairly complex roll motion (right-left-right) is captured nearly exactly.

The trajectory of the vehicle (see reference) was captured very well by the genetic algorithm.

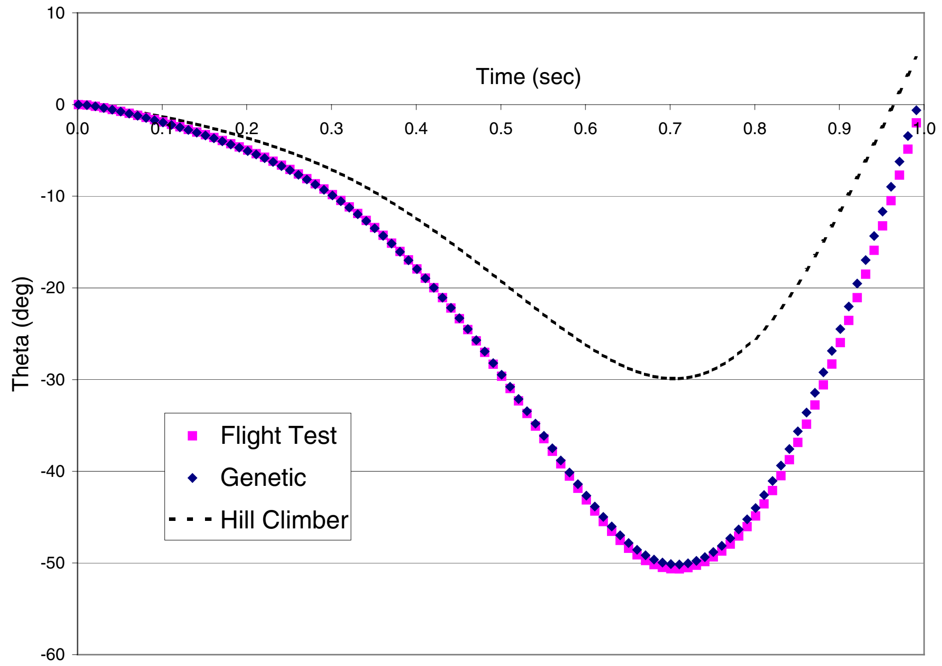


Figure 9. Euler Pitch Angle Comparison

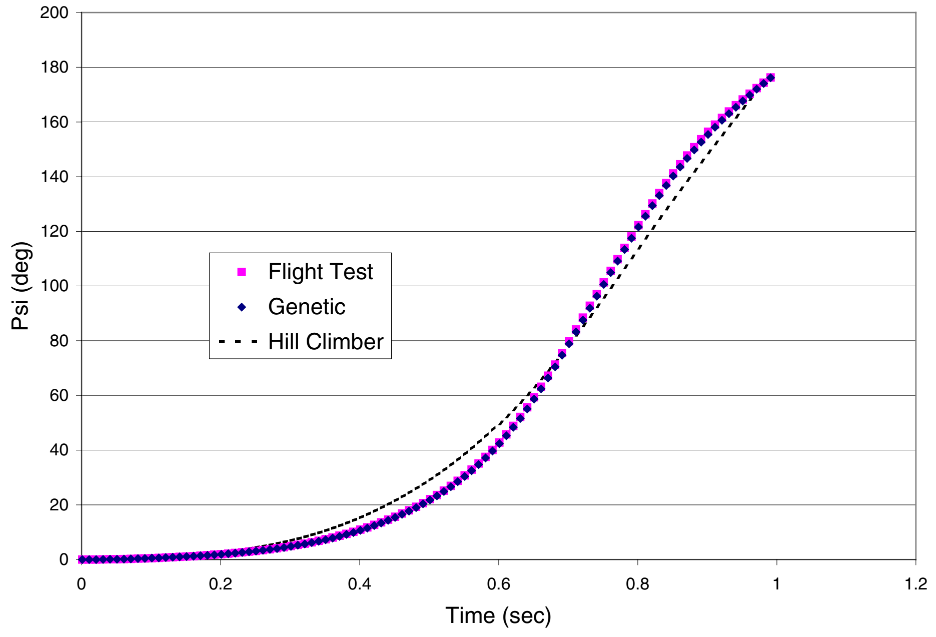


Figure 10. Euler Yaw Angle Comparison

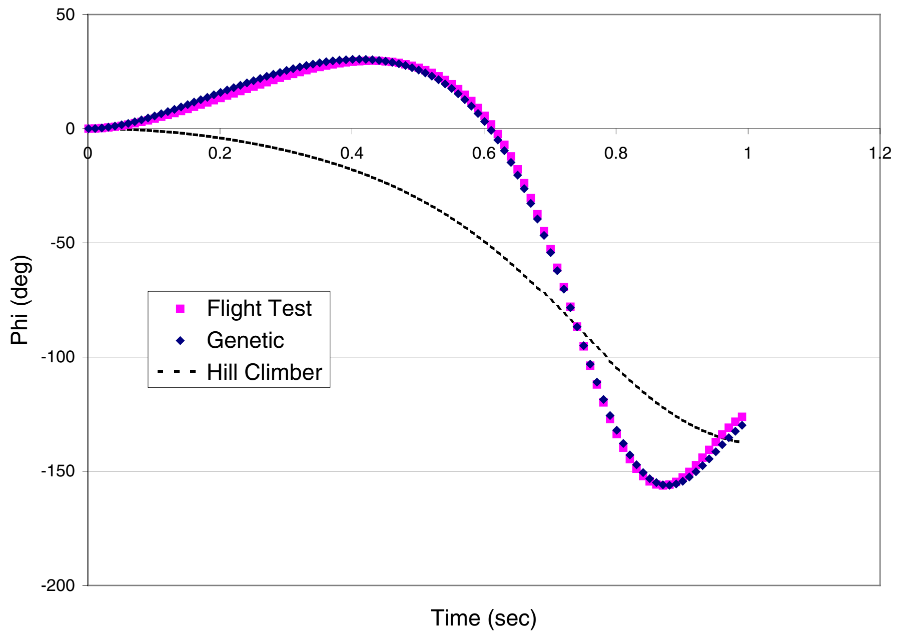


Figure 11. Roll Angle Comparison

The gains evidenced by the genetic algorithm are not, however, without cost. The hill climber found its answer in roughly 500 simulation runs (roughly 2 hours of CPU time). The genetic algorithm answer was generated in 50,000 simulation runs (500 generations with 100 members per generation).

Conclusions

As all these applications show, GA's have made significant contributions to the aerospace field. They have shown themselves to be superior to traditional design and analysis tools when complicated non-linear phenomena dominate the optimization space. The growth in the published applications of GA's shows that they are growing in popularity as design teams see their power. GA's have secured a significant role for themselves in future aerospace work. The conceptual design studies referenced in this paper show that they are capable of "system-level" optimization, and with the inclusion of additional considerations such as financial return or manufacturing suitability it is probable that GA's will become a tool for aerospace companies to use to gain a competitive edge.

References

[1] Goldberg, David E., Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Publishing Company, 1989.

[2] Krishnakumar, K., Goldberg, D.E., "Control System Optimization Using Genetic Algorithms", Journal of Guidance, Control, and Dynamics, Vol. 15, No. 3, May-June 1992.

[3] Rogers, James L., "A Parallel Approach to Optimum Actuator Selection with a Genetic Algorithm", AIAA Paper 2000-4484, Presented at the AIAA Guidance, Navigation, and Control Conference, Denver, CO, August 2000.

[4] Deshpande, S., Kumar, R., Seywald, H., Siemers, P., “Application of Genetic Algorithm to Air Data System Design”, AIAA Paper 92-4466, Presented at the AIAA Guidance, Navigation, and Control Conference, August 1992.
[5] Anderson, M.B., Lawrence, W.R., and Lopez, J.L., "Supplementing Gradient Search with Genetic Algorithm in Air Data Estimation System", AIAA paper 94-1931, presented at the 1994 Applied Aerodynamics Conference, Colorado Springs, CO., June 1994.
[6] Norris, S.R., and Crossley, W.A., "Pareto-Optimal Controller Gains Generated by a Genetic Algorithm," AIAA-98-1010, AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, 1998.
[7] McGookin, E.W., Murray-Smith, D.J., Li, Y., and Fossen, T.I., "Parameter Optimisation of a Non-linear Tanker Control System Using Genetic Algorithms," Proceedings of the 2nd IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems, Glasgow, United Kingdom, 1997.
[8] Martin, R.C. IV, "A Gain Scheduling Optimization Method Using Genetic Algorithms," Thesis, Air Force Institute of Technology, 1994.
[9] Karr, C.L., and Harper, T.R., "Genetic Algorithms in Adaptive Fuzzy Control," Conference Proceedings from The North American Fuzzy Logic Processing Society Meeting (NAFIPS 1992), Volume 2, pp. 506-514.
[10] Perhinschi, M.,G., "A Modified Genetic Algorithm for the Design of Autonomous Helicopter Control System," AIAA-97-3630, Presented at the AIAA Guidance, Navigation, and Control Conference, New Orleans, LA, August 1997.
[11] Karr, C.L., Freeman, L.M., and Meredith, D.L., "Genetic Algorithm based Fuzzy Control of Spacecraft Autonomous Rendezvous," NASA Marshall Space Flight Center, Fifth Conference on Artificial Intelligence for Space Applications, 1990.
[12] Homaifar, A., McCormack, E.D., "Full Design of Fuzzy Controllers Using Genetic Algorithms," NASA-N93-19452 06-80, The NASA Center for Aerospace Research at North Carolina A&T State University, 1992.
[13] Sweriduk, G.D., Menon, P.K., and Steinberg, M.L., “Robust Command Augmentation Systems Design Using Genetic Search Methods”, AIAA Paper 98-4131, AIAA Guidance, Navigation, and Control Conference and Exhibit, August 1998.
[14] Mondoloni, S., “A Genetic Algorithm for Determining Optimal Flight Trajectories”, AIAA Paper 98-4476, AIAA Guidance, Navigation, and Control Conference and Exhibit, August 1998.
[15] Perhinschi, M.G., “A Modified Genetic Algorithm for the Design of Autonomous Helicopter Control System”, AIAA Paper 97-3630, AIAA Guidance, Navigation, and Control Conference, August 1997.
[16] Kroo, I., Takia, M., “A Quasi-Procedure, Knowledge-Based System for Aircraft Design,” AIAA Paper No. 88-4428, Presented at the AIAA/AHS/ASEE Aircraft Design, Systems, and Operations Meeting, September 1988.
[17] Kroo, I., Altus, S., Braun, R., Gage, P., and Sobieski, I., “Multidisciplinary Optimization Methods for Aircraft Preliminary Design,”, AIAA Paper No. 94-4325.
[18]Gage, P., and Kroo, I., “Development of the Quasi-Procedural Method for Use in Aircraft Configuration Optimization,” AIAA Paper No. 92-4693, Presented at the Fourth AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization, September 1992.
[19] Consentino, G., and Holst, T., “Numerical Optimization Design of Advanced transonic Wing Configurations”, AIAA 85-0424, 1985.
[20] Gage, P., and Kroo, I., “A Role of Genetic Algorithms in a Preliminary Design Environment,” AIAA Paper No. 93-3933.
[21] Bramlette, M., and Cusic, R., “A Comparative Evaluation of Search Methods Applied to the Parametric Design of Aircraft”, Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann, 1989.

[22] Tong, S.S., “Turbine Preliminary Design Using Artificial Intelligence and Numerical Optimization Techniques.” Journal of Turbomachinery, Jan 1992, Vol 114/1.
[23] Anderson, M.B., “The Potential of Genetic Algorithms for Subsonic Wing Design”, AIAA Paper 95-3925, presented at the 1st AIAA Aircraft Engineering, Technology, and Operations Congress, Los Angeles, CA, September 1995.
[24] Anderson, M.B., “Using Pareto Genetic Algorithms for Preliminary Subsonic Wing Design”, AIAA Paper 96-4023, presented at the 6th AIAA/NASA/USAF Multidisciplinary Analysis and Optimization Symposium, Bellevue, WA, September 1996.
[25] Sharatchandra, M.C., Sen, M., and Gad-el-Hak, M., “New Approach to Constrained Shape Optimization Using Genetic Algorithms”, AIAA Journal, Vol. 36, No. 1, January 1998.
[26] Gregg, R.D., and Misegades, K.P., “Transonic Wing Optimization Using Evolution Theory”, AIAA 87-0520, presented at the AIAA 25th Aerospace Sciences Meeting, January 1987.
[27] Oyama, A., Obayashi, S., Nakahashi, K., “Transonic Wing Optimization Using Genetic Algorithm”, AIAA Paper 97-1854, 13th Computational Fluid Dynamics Conference, June 1997.
[28] Oyama, A., Obayashi, S., Nakahashi, K., “Fractional Factorial Design of Genetic Coding for Aerodynamic Optimization”, AIAA Paper 99-3298, 3rd AIAA Weakly Ionized Gases Workshop, November 1999.
[29] Anderson, M.B., Lawrence, W.R., and Lopez, J.L., "Supplementing Gradient Search with Genetic Algorithm in Air Data Estimation System", AIAA paper 94-1931, presented at the 1994 Applied Aerodynamics Conference, Colorado Springs, CO., June 1994.
[30] Selig, M.S., and Coverstone-Carroll, V.L., “Application of a Genetic Algorithm to Wind Turbine Design”, Presented at the 14th ASME ETCE Wind Energy Symposium, Houston, TX, January 1995.
[31] Cao, H.V., and Blom, G.A., “Navier-Stokes/Genetic Optimization of Multi-Element Airfoils”, AIAA 96-2487, Presented at the 14th AIAA Applied Aerodynamics Conference, New Orleans, LA, 1996.
[32] Jang, M., and Lee, J., “Genetic Algorithm Based Design of Transonic Airfoils Using Euler Equations”, AIAA Paper 2000-1584, Presented at the 41st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, April 2000.
[33] Jones, B.R., Crossley, W.A., and Anastasios, S.L., “Aerodynamic and Aeroacoustic Optimization of Airfoils Via a Parallel Genetic Algorithm”, AIAA Paper 98-4811, 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, September 1998.
[34] Perez, R.E., Chung, J., Behdinan, K., “Aircraft Conceptual Design Using Genetic Algorithms”, AIAA Paper 2000-4938, Presented at the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, September 2000.
[35] Anderson, M.B., Burkhalter, J.E., and Jenkins, R.M., "Design of an Air to Air Interceptor Using Genetic Algorithms", AIAA Paper 99-4081, presented at the 1999 AIAA Guidance, Navigation, and Control Conference, Portland, OR, August 1999.
[36] Anderson, M.B., Burkhalter, J.E., and Jenkins, R.M., “Intelligent Systems Approach to Designing an Interceptor to Defeat Highly Maneuverable Targets”, AIAA Paper 2001-1123, presented at the 39th Aerospace Sciences Meeting and Exhibit, Reno, NV, January 2001.
[37] Ewing, M.S., and Downs, K., “Conceptual Aircraft Design with Genetic Search Based on Financial Return on Investment”, AIAA Paper 96-4106, 6th AIAA/NASA/ISSMO Symposium of Multidisciplinary Analysis and Optimization”, September 1996.
[38] Chernyavsky, B., Stepanov, V., Rasheed, K., Blaize, M., and Knight, D., “3-D Hypersonic Inlet Optimization Using a Genetic Algorithm”, AIAA Paper 98-3582, 34th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, July 1998.

[39] Torella, G., Blasi, L., "The Optimization of Gas Turbine Engine Design by Genetic Algorithms", AIAA Paper 2000-3710, 36th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, July 2000.
[40] Uelschen, M., Lawerenz, M., "Design of Axial Compressor Airfoils With Artificial Neural Networks and Genetic Algorithms", AIAA Paper 2000-2546, Presented at the AIAA Fluid Dynamics Conference, June 2000.
[41] Schoonover, P.L., Crossley, W.A., and Heister, S.D., "Application of Genetic Algorithms to the Optimization of Hybrid Rockets", AIAA Paper 98-3349, 34th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, July 1998.
[42] Sforzini, Richard H., "An Automated Approach to Design of Solid Rockets Utilizing a Special Internal Ballistics Model", AIAA Paper 80-1135, Presented at the AIAA/SAE/ASME 16th Joint Propulsion Conference, June 30-July 2, 1980.
[43] Billheimer, J.S., "Optimization and Design Simulation in Solid Rocket Design," AIAA Paper 68-488, Presented at the 3rd AIAA Solid Propulsion Conference, June, 1968.
[44] Woltosz, W.S., "The Application of Numerical Optimization Techniques to Solid-Propellant Rocket Motor Design", M.S. Thesis, Auburn University, Auburn, Alabama, March 1977.
[45] Hooke, R., and Jeeves, T.A., "Direct Search Solution of Numerical and Statistical Problems", Journal of the Association of Computing Machinery, Vol. 8, No. 2, pp. 212-229.
[46] Whitney, D.E., "Pattern Search", Civil and Mechanical Engineering Joint Computer Facility, Massachusetts Institute of Technology, April 1974.
[47] Foster, W.A., Jr., and Sforzini, R.H., "Optimization of Solid Rocket Motor Igniter Performance Requirements", AIAA Paper 78-1016, Presented at the 14th AIAA/SAE Joint Propulsion Conference, Las Vegas, Nevada, July 1978.
[48] Swaminathan, V., Madhavan, N.S., "A Direct Random Search Technique for the Optimization of Propellant Systems," Indian Institute of Science, The Journal of the Aeronautical Society of India, Vol. 32, No. 1-4, February-November 1980, pp. 101-104.
[49] Peretz, A., Berger, M., "Thrust Profile Optimization of Solid-Propellant Two-Pulse Motors for Range Extension", AIAA Paper 92-3354, Presented at the AIAA/SAE/ASME/ASEE Joint Propulsion Conference, July 1992.
[50] Clergen, J.B., "Solid Rocket Motor Conceptual Design – The Development Of A Design Optimization Expert System With A Hypertext User Interface," AIAA Paper 93-2318, Presented at the 29th AIAA/ASME/SAE Joint Propulsion Conference, June, 1993.
[51] McCain, J.W., "A Variable Selection Heuristic for Solid Rocket Motor Design," The University of Alabama-Huntsville, Dissertation, University Microfilms No. DA9324488.
[52] Anderson, M.B., Burkhalter, J.E., and Jenkins, R.M., "Multi-Disciplinary Intelligent Systems Approach to Solid Rocket Motor Design, Part I: Single and Dual Goal Optimization", AIAA Paper 2001-3599, presented at the 37th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, Salt Lake City, UT, July 2001.
[53] Anderson, M.B., Burkhalter, J.E., and Jenkins, R.M., "Multi-Disciplinary Intelligent Systems Approach to Solid Rocket Motor Design, Part II: Multiple Goal Optimization", AIAA Paper 2001-3600, presented at the 37th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, Salt Lake City, UT, July 2001.
[54] Venter, G., and Haftka, R.T., "A Two Species Genetic Algorithm for Designing Composite Laminates Subject to Uncertainty", AIAA Paper 96-1535, 37th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference and Exhibit, April 1996.
[55] Nair, P.B., and Keane, A.J., "Combining Approximation Concepts with Genetic Algorithm-Based Structural Optimization Procedures", AIAA Paper 98-1912, 39th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference and Exhibit, April 1998.

[56] Missoum, S., Gurdal, Z., Hernandez, P., Guillot, J., “A Genetic Algorithm Based Topology Tool for Continuum Structures”, AIAA Paper 2000-4943, Presented at the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Long Beach, CA, September 2000.
[57] Furuya, H., “Combining Multilayer Neural Network and Genetic Algorithms for Strucutural Optimization”, AIAA Paper 99-1426, Presented at the 30th AIAA Plasmadynamics and Lasers Conference, June 1999.
[58] Bishop, J.A., and Striz, A.G., “Design of Vibration Suppression Systems in Space Trusses Using a Genetic Algorithm”, AIAA Paper 96-1537, 37th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference and Exhibit, April 1996.
[59] Shaviv, G.E., and Grunwald, A.J., “Genetic Algorithms for Optimizing Perspective Air Traffic Information Displays”, AIAA Paper 2000-0302, 38th AIAA Aerospace Sciences Meeting and Exhibit, January 2000.
[60] Frayssinhes, E., “Investigating New Satellite Constellation Geometries with Genetic Algorithms”, AIAA Paper 96-3636, AIAA/AAS Astrodynamics Conference, July 1996.
[61] Begley, S., “Software au Naturel”, Newsweek Magazine, May 8, 1995.
[62] Wollam, J.D., Kramer, S., Campbell, S., “Reverse Engineering of Foreign Missiles Via Genetic Algorithms”, AIAA Paper 2000-0685, 38th AIAA Aerospace Sciences Meeting and Exhibit, January 2000.
[63] Anderson, M.B., “A New Tool for Conducting Post-Flight Simulation Analysis and Validation Using an Intelligent Systems Approach”, presented at the ITEA 2001 Aircraft-Stores Compatibility Symposium, Destin, FL, March 2001.
[64] Iliff, K. W., “Parameter Estimation for Flight Vehicles,” AIAA Journal of Guidance, Control, and Dynamics, Vol. 12, No. 5, Sept-Oct 1989, pp. 609-622.
[65] Chapman, G.T. and D.B. Kirk, "A Method for Extracting Aerodynamic Coefficients from Free-Flight Data," AIAA Journal, Vol. 8, No. 4, April 1970.
[66] Welsh, C.J. and W.R. Lawrence, "Motion Analysis Procedure for Asymmetric Vehicles," AEDC-TR 75-153, May 1976, AIAA 3rd Atmospheric Flight Mechanics Conference, June 1976.
[67] Anderson, M.B., “A New Tool for Conducting Post-Flight Simulation Analysis and Validation Using an Intelligent Systems Approach”, presented at the ITEA 2001 Aircraft-Stores Compatibility Symposium, Destin, FL, March 2001.

Trends in Intelligent System Technologies for Airfoil Design: - Computer Algebra Techniques for Gradient Computation through the Continuous Adjoint Approach - Use of Approximate Fitness Evaluators in Evolutionary Optimization

Domenico Quagliarella[‡], Domenic D'Ambrosio[⋆], Angelo Iollo[⋆]
[‡] Centro Italiano Ricerche Aerospaziali
Via Maiorise, 81043 Capua, Italy
[⋆]Dipartimento di Ingegneria Aeronautica e Spaziale, Politecnico di Torino
C.so Duca degli Abruzzi 24, 10129 Torino, Italy
d.quagliarella@cira.it, domenic@athena.polito.it, iollo@polito.it

1 Introduction

Aerodynamic shape design is a challenging application for modern computational fluid dynamic. This is, indeed, a very productive field, both in term of scientific publications and developed applications. Furthermore it is possible to find excellent analysis/design software released in the public domain (See for example [1, 2]). However, industry, in particular automotive and aerospace, but also emerging fields like wind and sea extracted energy, continuously needs increased product performance and competitiveness. This requires increasingly refined, advanced and complex computational models capable of describing flow behavior with higher and higher fidelity.

Thus the design tools available to scientist and engineers are challenged by the growing complexity of simulation models. The increasing price to performance ratio of nowadays computers is a strong allied of designers in this though race, but often it is not enough.

Objectives of this lecture is to explore and analyze some of the technologies that, together with the increasingly available computational power, may help to exploit the tremendous potential of high fidelity flow field models to aerodynamic shape design challenges.

For the sake of simplicity we will concentrate on aerofoil design. Two aspects will be considered: the computer algebra assisted development of code related to gradient computation through adjoint techniques and the use of fitness approximation techniques to improve the performance of search procedures. The concern of this first lecture will be more methodological, while some applications will be illustrated in the next one.

The outline of the first first part of this lecture is as follows: some short remarks on adjoint problems will introduce the framework in which the Computer Algebra techniques will be mainly used here. Then a brief outline of symbolic computation systems will be given, with particular reference to the features that are here used for adjoint equation derivation. The implemented algorithm will be described and illustrated with an example on a simple problem. The adjoint and the related boundary conditions for 3D Navier-Stokes are then reported, while an example of derivation is given in appendix A.

The second part of this lecture will focus on the use of approximated fitness evaluators. The problem will be analyzed using the framework of multi-objective optimization, and some possible approaches to the problem of mixing exact and approximated evaluations will be discussed.

2 Adjoint formulation for gradient evaluation

The adjoint formulation appears in optimal control theory as a tool to compute the constrained gradient to a given functional. In fluid mechanics the applications span from shape design to flow control [3, 4, 5, 6]. The

Paper presented at the RTO AVT Course on "Intelligent Systems for Aeronautics", held in Rhode-Saint-Genèse, Belgium, 13-17 May 2002, and published in RTO-EN-022.

adjoint method is based on the solution of an additional set of equations for the Lagrange multipliers. The Lagrange multipliers local value is a measure of the functional sensitivity to the flow variables local variation. This provides, for example, a quantitative criteria for grid refinement [7].

In mathematical terms the adjoint approach to gradient computation can be concisely derived from the Lagrange identity:

$$(Av, w) = (v, A^* w) \tag{1}$$

where (,) is the scalar product in the appropriate Hilbert space. This identity defines the adjoint operator A^* when a linear or non-linear operator A and when $v \in D,\ w \in D^*$ are given. Now, if

$$J = (p, v) \tag{2}$$

is a given functional and

$$Av = f, \quad A^* w = p \tag{3}$$

are, respectively, the state and the adjoint equations, then it is possible to write

$$J = (p, v) = (f, w) \tag{4}$$

as a direct consequence of identity 1. This means that J can be expressed either in terms of the state vector v or in terms of the adjoint one w. The small perturbations theory can then be used to obtain the functional variation for non-linear state equations. In general, if $A(v)$ is an operator that depends non-linearly on v, it is possible to write, under the hypothesis of small perturbations, that

$$\begin{aligned} &f = f_0 + \varepsilon f_1 + \ldots \\ &v = v_0 + \varepsilon v_1 + \ldots \\ &A(v_0 + \varepsilon v_1 + \ldots) = A(v_0) + \varepsilon \left.\frac{dA}{dv}\right|_0 v_1 + \ldots \end{aligned} \tag{5}$$

Therefore the non-linear state equation $A(v)v = f$ can be decomposed as follows

$$\begin{aligned} &A(v_0)v_0 = f_0 \\ &A(v_0)v_1 + \left(\left.\frac{dA}{dv}\right|_0 v_1 \right) v_0 = f_1 \end{aligned} \tag{6}$$

From the last equation descends immediately the first order linearized operator:

$$A_1(v_0)\bullet = A(v_0)\bullet + \left(\left.\frac{dA}{dv}\right|_0 \bullet \right) v_0 \tag{7}$$

In the same way, it is possible to write the functional J as

$$J = J_0 + \varepsilon J_1 \tag{8}$$

where

$$\begin{aligned} &J_0 = (v_0, p) = (w_0, f_0) \\ &\delta J = J_1 = (v_1, p) = (w_1, f_1) \\ &\text{with} \quad A_1^*(v_0)w_1 = p \end{aligned} \tag{9}$$

In other words we obtain explicitly the functional variation with respect to the right hand side (RHS) of the governing equation, i.e., the gradient of J with respect to f. Hence, from a mathematical point of view, the derivation of the adjoint state equations for a given functional J is a straightforward process. A disadvantage of this method is that complicated functionals often require a lengthy and error-prone manual derivation and hand coding process. In this respect Computer Algebra (CA) techniques can be profitably used to ease and speed-up this process. In the following sections such technique will be described and then used to automatically obtain the adjoint equations and pertinent boundary conditions of the Navier-Stokes equations. Gradient expression is also automatically computed for a generic objective function. A standard computer algebra language (MAPLE [8]) is used to build the automated derivation procedure.

3 Automatic derivation of the adjoint system and gradient

3.1 General remarks on symbolic computation

Computer algebra systems for symbolic computation have a number of peculiarities that make them radically different from numeric computing tools. Indeed, they operate on symbols (the rich set of mathematical objects), rather than numbers as classic numerical programming languages. The mathematical objects are transformed following exact equivalence rules, and there are not precision limits in the representation of numbers. The algebraic manipulation algorithms can be very complex and sophisticated so that a computer algebra language can be considered a way of storing and efficiently using the knowledge and expertise of thousands of brilliant mathematicians. This allows an easy and straightforward use of computer algebra systems as "scratchpad" tools in the interactive solutions of complex mathematical problems.

On the other hand, there are some drawbacks that have to be taken into account when developing complex applications. In particular, there are cases in which the output is very complex, and considerable effort has to be devoted to keeping it easily understandable. Furthermore, it has to be also considered that it is not easy to learn how to write efficient code in a computer algebra language, because the programming model is fundamentally different with respect to "numerical programming", with recursion, list and set operations, and expression identification and and matching on a complex and rich set of objects.

The approach followed in the present work was the research of a compromise between output conciseness and ease of implementation, and the main steps of the developed adjoint equation derivation procedure can be summarized as follows:

- The Navier-Stokes equations have been specified term by term, but the volume and surface integrals of the Lagrangian have been left unexpanded.
- The classical manipulations for adjoint derivation were obtained using Green formulas, while the variation with respect to geometry were obtained through Hadamard formulas.

This approach required the capability of discriminating between the various possible structures of a given Lagrangian, in order to apply the proper transformations. This was obtained applying recursively the pattern matching functions of the CA system.

3.2 Adjoint computation

We provide here expressions and identities that will be useful for the automatic computation of the integral variations by the MAPLE V computer algebra system [8].

The first step in the automated derivation process is the expression of the flow field equation in indexed form, using, in the present implementation, the Grtensor software [9] for tensor calculus. The expanded equation and boundary conditions, referenced as $\mathbf{R}$, are then combined with the objective function $\mathbf{O}$ to form the Lagrangian equation

$$\begin{aligned} L \;=\; & \iint\limits_{\Sigma(h)} \mathbf{O}(x,\dots,u,u_x,u_y,u_z,u_{xx},u_{xy},\dots,h,\dots)\, d\sigma + \\ & \iiint\limits_{\Omega(h)} \lambda\mathbf{R}(x,\dots,u,u_x,u_y,u_z,u_{xx},u_{xy},\dots,h,\dots)\, d\Omega \end{aligned} \tag{10}$$

The adjoint equations and boundary conditions are obtained computing the variation of L with respect to the flow field unknowns u, while the gradient has been computed through the variation of L with respect to the boundary defining functions h. The formulas used for the variation of volume and surface integrals are reported in the following sections. One of the tricky aspects of this approach is that the variation of L has to be computed with respect to quantities that are, in turn, functions of the coordinates x, y and z. In fact, standard MAPLE differentiation function is unable to perform variational calculus. Yet, there are ways to get around such a difficulty, for example using the `pdiff` [10] function, available in MAPLE V share library. Such function enables the computation of expressions such as

$$\frac{\partial f(x, u(x))}{\partial u(x)}$$

that are not computable with the standard MAPLE differentiation function. The Green's theorem is then repeatedly applied to move the terms containing the derivative of a variation from the volume terms to the surface integral terms. MAPLE pattern matching functions have been extensively used both for the variation procedure and the Green's theorem procedure implementation.

3.3 Symbolic computation algorithm

Given the Lagrangian expression, in the following we give some details about the algorithm used to obtain the adjoint equations, the boundary conditions, and the gradient expression. The derivation of the formulas adopted to express the variation of the functionals is reported in appendix B.

The first MAPLE function implemented is called `variation`. This function takes as input a generic functional L, a function u to be variated and eventually the vector function $\mathbf{h}$ that defines the geometric deformation. In general, the expansion obtainable for a generic variation is of the kind

$$\delta L_u(x, \dots, u, u_x, u_y, u_z, u_{xx}, u_{xy}, \dots, h, \dots) =$$

$$\frac{\partial L}{\partial u}\delta u + \frac{\partial L}{\partial u_x}\delta u_x + \frac{\partial L}{\partial u_y}\delta u_y + \frac{\partial L}{\partial u_z}\delta u_z + \frac{\partial L}{\partial u_{xx}}\delta u_{xx} + \frac{\partial L}{\partial u_{xy}}\delta u_{xy} \tag{11}$$

When the variation is computed with respect to the geometry, a pattern matching algorithm checks each term of the functional L in order to find volume or surface integrals. In the case of a surface integral, then the following expression is used to compute the variation:

$$\delta F = \iint\limits_S \nabla f \cdot \frac{\partial \mathbf{h}}{\partial g}\,\delta g\, d\sigma + \iint\limits_S f \boldsymbol{\mathcal{H}} \cdot \frac{\partial \mathbf{h}}{\partial g}\,\delta g\, d\sigma + \iint\limits_S \frac{\partial f}{\partial g}\,\delta g\, d\sigma \tag{12}$$

Appendix sub-section B.1 reports the derivation of this expression and the meaning of the terms involved. In the case of volume integrals, the formula

$$\delta F = \iiint\limits_T \frac{\partial f}{\partial g}\,\delta g\, dV + \iint\limits_{FT} f\,\frac{\partial \mathbf{h}}{\partial g} \cdot \mathbf{n}\,\delta g\, d\sigma \tag{13}$$

is adopted. Expression derivation and terms definition are in B.3. These formulas are exactly equivalent to the Hadamard formulas. It is worth to remark here that volume and surface integrals, need special treatment only when the integration manifold is given in implicit form. In the other cases they can be reduced to equivalent double and triple definite integrals and treated with formula 11.

The second MAPLE function implemented is called `intparts3d`. The input to this function is the variated functional δL, and in output an equivalent form of the functional δL is generated, were the terms containing derivatives of the variation are moved to surface integrals. It works applying recursively the integration by parts and the Green's theorem to the volume integral terms containing derivatives of the variation, e.g. δu_x or δu_{xy}. The resulting functional variation presents terms containing derivatives of the variation only inside surface integrals. In a subsequent step, these terms can be eliminated through the imposition of suitable boundary conditions. This last step, currently, is not automated and require some care and attention on the part of the user. To help this process, the functions `extract_adj_eq` and `extract_adj_bceq` have been introduced. They extract the adjoint equations and boundary conditions, respectively, again by pattern recognition. Finally, two interface procedures, `compute_adj_eq` and `compute_adj_eq_bc`, heve been introduced to manage the whole process of adjoint equation and boundary conditions derivation. The listing in MAPLE code of the first one is reported below.

```
compute_adj_eq :=
   proc(lagrangian, v, dv, n_intp, h, verbose_level, aliasing::boolean)
   option remember;
   local lg_v, i, c1, sp, adj_eq;
       lg_v := variation(lagrangian, [v=v+dv], h, [s1, s2], [x1, x2, x3]);
       lg_v := expand(lg_v);
       if aliasing then autoAlias(lg_v) fi;
       if verbose_level >= 2 then print("LAGRANGIAN"); print(lg_v) fi;
       c1 := lg_v;
       for i from 1 by 1 to n_intp do
           c1 := intparts_3d(c1,dv);
           c1 := expand(convert(c1,diff));
           if aliasing then autoAlias(c1) fi;
           if verbose_level >= 1 then
               print("INTEGRATION BY PARTS N.", i); print(c1)
           fi;
       od;
       adj_eq := extract_adj_eq(c1,dv);
   end:
```

The equations obtained by the automated procedure are of course less concise compared to those obtained manually. Indeed, the governing equations are fully expanded as a first step in the variation computation, to ease the pattern matching process. Furthermore, the comparison, useful for debug purposes, with the manually obtained equations is complicated by some asymmetries that may be found in the automatically derived terms. This is due to the following identity (see sub-appendix B.4 for details):

$$\iiint\limits_{\Omega} v_{xy}\, d\Omega = \iint\limits_{\Sigma} v_x n_2\, d\Sigma = \iint\limits_{\Sigma} v_y n_1\, d\Sigma \tag{14}$$

This comes from the application of the divergence theorem to terms containing crossed derivatives. Indeed, the theorem can be applied here in two different ways, and the automatic procedure does not know which form has to be chosen to keep symmetry. However, the same identity 14 can be applied to suitably transform the obtained expressions.

3.4 A simple application example to potential flows

An example of adjoint equation derivation is here given to show how the developed MAPLE code may be used. The reader may refer to appendix A for the application to Navier-Stokes equations.

A symmetric airfoil is immersed in a potential flow at zero angle of attack (see figure 1), and a symmetric transpiration velocity W is assigned in the direction normal to the airfoil surface. The governing equations are therefore:

$$\nabla^2\phi = 0 \quad \text{in the field,} \qquad \nabla\phi\cdot\mathbf{n} = W \quad \text{on the airfoil} \tag{15}$$

The problem is to find a transpiration velocity distribution W^* such that a given target potential Φ is matched. Hence the objective function that will be minimized is:

$$\frac{1}{2}\oint(\phi-\Phi)^2\, d\sigma \tag{16}$$

on the airfoil surface. The current version of adjoint derivation code was developed taking into account three-dimensional fields; therefore in the following example, the two-dimensional field will be treated as 3D field with zero gradient of the flow variables in $x3$ direction. In the same spirit the airfoil is a cylindric surface identified by two curvilinear coordinates $s1, s2$. Thus we will have surface integrals instead of contour integrals on the boundary and volume integrals in the field.

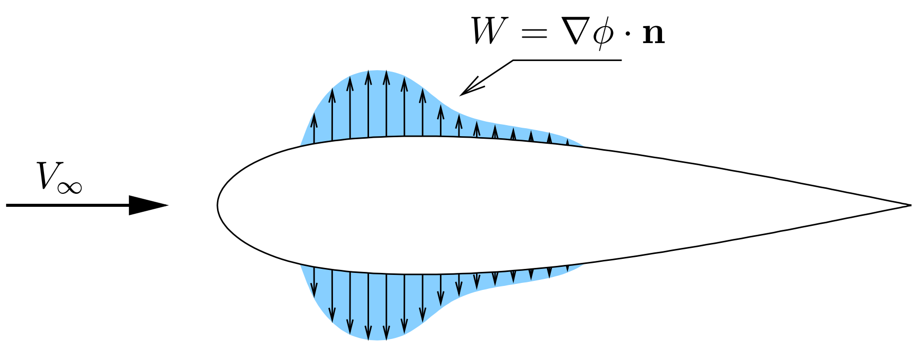


Figure 1: 2D symmetric potential flow with blowing.

Here the MAPLE interactive session for the adjoint computation follows. The `alias_dependencies` function set a functional dependence for a given list of variables; dW is the increment of the blowing velocity, y is the airfoil ordinate, $x1, x2, x3$ are the field coordinates, while $s1, s2$ are the surface coordinates. Note that to have the procedure working properly, we had to specify explicitly the dependence of ϕ and of its increment $d\phi$ on field and surface coordinates. The same dependence is specified for the vector normal to the airfoil surface $\mathbf{n} = [n1, n2, n3]$ and for the deformation field $\mathbf{h} = [h1, h2, h3]$ (not explicitly used here).

```
>  alias_dependencies([y,W,dW],[s1]):
>  varlist := [x1,x2,x3,s1,s2]:
>  alias_dependencies([phi,dphi,n1,n2,n3,h1,h2,h3],varlist):
>  alias_dependencies([Phi],[s1,s2]):
>  Obj:=Int(Int(1/2*(phi-Phi)ˆ2,s1),s2);
```

$$Obj := \int\int \frac{1}{2}(\phi - \Phi)^2 \, ds1 \, ds2$$

```
>  Potential_eq:=diff(phi,‘$‘(x1,2))+diff(phi,‘$‘(x2,2)):autoAlias(%);
```

$$\phi_{x1,\,x1} + \phi_{x2,\,x2}$$

```
>  n:=[n1,n2];
```

$$n := [n1,\ n2]$$

```
>  Neumann_bc:=dotprod(grad(phi,[x1,x2]),n,’orthogonal’)-W:
>  autoAlias(%);
```

$$\phi_{x1}\, n1 + \phi_{x2}\, n2 - \mathrm{W}$$

```
>  alias_dependencies([eta,beta],varlist):
>  Lagrangian:=Obj+Int(Int(Int(eta*Potential_eq,x1),x2),x3)+
>  Int(Int(beta*Neumann_bc,s1),s2);
```

$$Lagrangian := \int\int \frac{1}{2}(\phi - \Phi)^2 \, ds1 \, ds2 + \int\int\int \eta\,(\phi_{x1,\,x1} + \phi_{x2,\,x2})\, dx1\, dx2\, dx3$$
$$+ \int\int \beta\,(\phi_{x1}\, n1 + \phi_{x2}\, n2 - \mathrm{W})\, ds1\, ds2$$

```
>  adj_dphi:=compute_adj_eq(Lagrangian,phi,dphi,2,[h1,h2,h3],0,true);
```

$$adj\_dphi := \eta_{x1,\,x1} + \eta_{x2,\,x2}$$

```
>  adj_dphi_bc:=compute_adj_bc(Lagrangian, phi, dphi, 2, [n1, n2, n3],
>  [h1, h2, h3], 0, true);
```

$$adj\_dphi\_bc :=$$
$$\phi - \Phi + \frac{\eta\, dphi_{x1}\, n1}{dphi} - \eta_{x1}\, n1 + \frac{\eta\, dphi_{x2}\, n2}{dphi} - \eta_{x2}\, n2 + \frac{\beta\, n1\, dphi_{x1}}{dphi} + \frac{\beta\, n2\, dphi_{x2}}{dphi}$$

```
>  adj_dphi_bc_airf:=subs(beta=-eta,adj_dphi_bc);
```

$$adj\_dphi\_bc\_airf := \phi - \Phi - \eta_{x1}\, n1 - \eta_{x2}\, n2$$

```
>  c6:=variation(Lagrangian,[W=W+dW],[s1,y,s2],[s1,s2],[x1,x2,x3]);
```

$$-\iint \beta\, dW\, ds1\, ds2$$

In the above equation the gradient with respect to the blowing velocity W can be recognized as being given by the variation of the Lagrangian with respect to W.

4 The compressible Navier Stokes Adjoint Equations

In this section the mathematical adjoint to the Navier-Stokes equations are derived. The notation is slightly different from that of the previous chapter for the sake of conciseness and clarity. In particular, the variations will be indicated with a circumflex accent.

4.1 State equations

In this section we make the notation clear by giving the steady compressible Navier-Stokes equations under divergence form and indicial notation.

Conservation of mass

$$\frac{\partial \rho v_i}{\partial x_i} = 0 \tag{17}$$

Conservation of momentum

$$\frac{\partial}{\partial x_j}\left\{\rho v_i v_j + p\delta_{ij} - \mu\left[\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} - \frac{2}{3}\frac{\partial v_k}{\partial x_k}\delta_{ij}\right]\right\} = 0 \tag{18}$$

Conservation of energy

$$\frac{\partial}{\partial x_i}\left\{\rho v_i H - k\frac{\partial T}{\partial x_i} - \mu\left[\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} - \frac{2}{3}\frac{\partial v_k}{\partial x_k}\delta_{ij}\right] v_j\right\} = 0 \tag{19}$$

H is the total enthalpy per unit mass, k the heat conductivity, and the other symbols have their usual meaning.

Let Ω be the flow domain and Σ its surface. The no-slip boundary conditions are imposed on Σ, elsewhere we consider suitable far field conditions. The boundary condition on temperature is to be detailed later.

4.2 Lagrangian

I is a generic functional depending on the flow field and it is defined either over Ω or over Σ. We want to find a surface Σ such that the first order variation of I with respect to a perturbation of Σ is zero.

To this end, we introduce an auxiliary functional L defined as follows

$$\begin{aligned}
L = I \quad &+ \int_\Omega \eta\,\frac{\partial \rho v_i}{\partial x_i}\,d\Omega \\
&+ \int_\Omega \lambda_i\,\frac{\partial}{\partial x_j}\left\{\rho v_i v_j + p\delta_{ij} - \mu\left[\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} - \frac{2}{3}\frac{\partial v_k}{\partial x_k}\delta_{ij}\right]\right\} d\Omega \\
&+ \int_\Omega \zeta\frac{\partial}{\partial x_i}\left\{\rho v_i H - k\frac{\partial T}{\partial x_i} - \mu\left[\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} - \frac{2}{3}\frac{\partial v_k}{\partial x_k}\delta_{ij}\right] v_j\right\} d\Omega \\
&+ \int_\Sigma \xi_i v_i\,d\Sigma \\
&+ \int_\Sigma \tau\Theta(T)\,d\Sigma
\end{aligned} \tag{20}$$

$\Theta(T) = T - T_w$ with prescribed boundary temperature, whereas for adiabatic flows $\Theta(T) = \partial T/\partial x_i n_i$. The auxiliary functions η, λ_i and ζ belong to $H^2(\Omega)$. $\xi \in L^2(\Sigma)$ and $\tau \in L^2(\Sigma)$ or $\tau \in H^1(\Sigma)$ according to the boundary conditions selected for temperature. We refer to all these functions as Lagrange multipliers.

The lagrangian function L transforms the constrained problem defined for I into an unconstrained problem: it is now required to find the flow field, Σ and the Lagrange multipliers in order that the respective first order variations of L are zero. The variation of L with respect to the Lagrange multipliers yields the flow equations and boundary conditions. We concentrate on the variation of L with respect to the conservative flow variables ρ, ρv_i, ρE, with E the total energy per unit mass.

Take the variation of the first integral in eq. 20, using the Gauss identity we have

$$\int_\Sigma \eta\, \widetilde{\rho v_i}\, n_i \, d\Sigma - \int_\Omega \widetilde{\rho v_i} \frac{\partial \eta}{\partial x_i} d\Omega \tag{21}$$

where n_i is the outward unit normal to Σ, and $\widetilde{\cdot}$ denotes the function variation.

Consider now the second integral in eq. 20 and apply the Gauss identity twice to get

$$\begin{aligned}&\int_\Sigma \lambda_i f_i \, d\Sigma - \int_\Omega (\rho v_i v_j + p\delta_{ij}) \frac{\partial \lambda_i}{\partial x_j} d\Omega - \int_\Omega \frac{\partial}{\partial x_j}\left\{\mu\left[\frac{\partial \lambda_i}{\partial x_j} + \frac{\partial \lambda_j}{\partial x_i} - \frac{2}{3}\frac{\partial \lambda_k}{\partial x_k}\delta_{ij}\right]\right\} v_i \, d\Omega \\ + \; &\int_\Sigma \mu\left[\frac{\partial \lambda_i}{\partial x_j} + \frac{\partial \lambda_j}{\partial x_i} - \frac{2}{3}\frac{\partial \lambda_k}{\partial x_k}\delta_{ij}\right] v_i n_j \, d\Sigma\end{aligned} \tag{22}$$

where the components of the force per unit surface experienced at the boundary are

$$f_i = \left\{\rho v_i v_j + p\delta_{ij} - \mu\left[\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} - \frac{2}{3}\frac{\partial v_k}{\partial x_k}\delta_{ij}\right]\right\} n_j$$

To be remarked is that the linear second order adjoint operator keeps the same form of the viscous stresses.

In eq. 22 the flow variables do not appear under the form of conservative variables. However, in the first integral, for a reason that will be clear later, we leave the flow variation under the form $\widetilde{f_i}$, while in the others we write the variations in terms of conservative variables. For example we have

$$\widetilde{\rho v_i v_j} = \widetilde{\rho v_i} v_j + \widetilde{\rho v_j} v_i - v_i v_j \widetilde{\rho}$$

$$\widetilde{v_i} = \frac{1}{\rho}\widetilde{\rho v_i} - \frac{v_i}{\rho}\widetilde{\rho}$$

$$\widetilde{p} = \frac{\gamma - 1}{2}(2\,\widetilde{\rho E} - 2\,\widetilde{\rho v_i} v_i + V^2 \widetilde{\rho})$$

$$\widetilde{T} = \frac{\gamma - 1}{2\rho R}\left[2\,\widetilde{\rho E} - 2\,\widetilde{\rho v_i} v_i + \left(V^2 - \frac{2R}{\gamma - 1}T\right)\widetilde{\rho}\right]$$

$$\widetilde{\rho v_i H} = \widetilde{\rho v_i} H + \gamma v_i \widetilde{\rho E} - (\gamma - 1) v_i \widetilde{\rho v_j} v_j + v_i\left(\frac{\gamma - 1}{2}V^2 - H\right)\widetilde{\rho}$$

with γ the specific heats ratio, R the perfect gas constant and $V^2 = v_i v_i$. Substituting the above relations in eq. 22 and posing

$$h_i = \frac{\partial}{\partial x_j}\left\{\mu\left[\frac{\partial \lambda_i}{\partial x_j} + \frac{\partial \lambda_j}{\partial x_i} - \frac{2}{3}\frac{\partial \lambda_k}{\partial x_k}\delta_{ij}\right]\right\}$$

one finds

$$
\begin{aligned}
&\int_\Sigma \lambda_i \widetilde{f}_i \, d\Sigma - \int_\Omega \left[v_j \frac{\partial \lambda_i}{\partial x_j} + v_j \frac{\partial \lambda_j}{\partial x_i} - (\gamma - 1) v_i \frac{\partial \lambda_j}{\partial x_j} + \frac{h_i}{\rho} \right] \widetilde{\rho v_i} \, d\Omega \\
- \quad &\int_\Omega \left(\frac{\gamma - 1}{2} v_j v_j \frac{\partial \lambda_i}{\partial x_i} - v_i v_j \frac{\partial \lambda_i}{\partial x_j} - \frac{1}{\rho} v_i h_i \right) \widetilde{\rho} \, d\Omega \\
- \quad &\int_\Omega (\gamma - 1) \frac{\partial \lambda_i}{\partial x_i} \widetilde{\rho E} \, d\Omega + \int_\Sigma \mu \left[\frac{\partial \lambda_i}{\partial x_j} + \frac{\partial \lambda_j}{\partial x_i} - \frac{2}{3} \frac{\partial \lambda_k}{\partial x_k} \delta_{ij} \right] \widetilde{v}_i n_j \, d\Sigma
\end{aligned}
\tag{23}
$$

The third integral in eq. 20 is rearranged as above to get

$$
\begin{aligned}
&\int_\Sigma \zeta e \, d\Sigma - \int_\Sigma k \, \zeta \frac{\partial T}{\partial x_i} d\Sigma - \int_\Omega \rho v_i H \frac{\partial \zeta}{\partial x_i} d\Omega - \int_\Sigma k \, T \frac{\partial \zeta}{\partial x_i} n_i \, d\Sigma + \int_\Omega T \frac{\partial}{\partial x_i} \left(k \frac{\partial \zeta}{\partial x_i} \right) d\Omega \\
+ \quad &\int_\Omega \mu \frac{\partial \zeta}{\partial x_i} \left[\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} - \frac{2}{3} \frac{\partial v_k}{\partial x_k} \delta_{ij} \right] v_j \, d\Omega
\end{aligned}
\tag{24}
$$

with

$$
e = \left\{ \rho v_i H - \mu \left[\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} - \frac{2}{3} \frac{\partial v_k}{\partial x_k} \delta_{ij} \right] v_j \right\} n_i
$$

The variation of eq. 24 is complicated by the presence of non-linear terms involving both the v_j and the derivatives. As a first step, we write the variation of eq. 24 as

$$
\begin{aligned}
&\int_\Sigma \zeta \widetilde{e} \, d\Sigma - \int_\Sigma k \, \zeta \frac{\partial \widetilde{T}}{\partial x_i} n_i \, d\Sigma - \int_\Omega \widetilde{\rho v_i H} \frac{\partial \zeta}{\partial x_i} d\Omega + \int_\Sigma k \, \widetilde{T} \frac{\partial \zeta}{\partial x_i} n_i \, d\Sigma - \int_\Omega \widetilde{T} \frac{\partial}{\partial x_i} \left(k \frac{\partial \zeta}{\partial x_i} \right) d\Omega \\
+ &\int_\Omega \left\{ \mu \frac{\partial \zeta}{\partial x_i} \left[\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} - \frac{2}{3} \frac{\partial v_k}{\partial x_k} \delta_{ij} \right] - \frac{\partial}{\partial x_i} \left[\mu \left(v_j \frac{\partial \zeta}{\partial x_i} + v_i \frac{\partial \zeta}{\partial x_j} - \frac{2}{3} v_k \frac{\partial \zeta}{\partial x_k} \delta_{ij} \right) \right] \right\} \widetilde{v}_j d\Omega \\
+ &\int_\Sigma \mu \, n_i \left(\frac{\partial \zeta}{\partial x_j} v_i + \frac{\partial \zeta}{\partial x_i} v_j - \frac{2}{3} \frac{\partial \zeta}{\partial x_k} v_k \delta_{ij} \right) \widetilde{v}_j \, d\Sigma
\end{aligned}
\tag{25}
$$

Leaving unchanged the terms on the boundary and expressing the other variations in terms of the conservative variables variations, the equation above finally becomes

$$
\begin{aligned}
&\int_\Sigma \zeta \widetilde{e} \, d\Sigma - \int_\Sigma k \, \zeta \frac{\partial \widetilde{T}}{\partial x_i} d\Sigma + \int_\Sigma k \, \widetilde{T} \frac{\partial \zeta}{\partial x_i} n_i \, d\Sigma + \int_\Sigma \mu \, n_i \left(\frac{\partial \zeta}{\partial x_j} v_i + \frac{\partial \zeta}{\partial x_i} v_j - \frac{2}{3} \frac{\partial \zeta}{\partial x_k} v_k \delta_{ij} \right) \widetilde{v}_j \, d\Sigma \\
+ \quad &\int_\Omega \left\{ \left[(\gamma - 1) \, v_i v_j - H \delta_{ij} \right] \frac{\partial \zeta}{\partial x_i} + \frac{\gamma - 1}{\rho \, R} v_i \frac{\partial}{\partial x_i} \left(k \frac{\partial \zeta}{\partial x_i} \right) - \frac{g_j}{\rho} \right\} \widetilde{\rho v_j} \, d\Omega \\
+ \quad &\int_\Omega \left[\frac{g_j v_j}{\rho} - \frac{\gamma - 1}{2 \rho R} \left(V^2 - \frac{2R}{\gamma - 1} T \right) v_j \frac{\partial}{\partial x_i} \left(k \frac{\partial \zeta}{\partial x_i} \right) - \left(\frac{\gamma - 1}{2} V^2 - H \right) v_i \frac{\partial \zeta}{\partial x_i} \right] \widetilde{\rho} \, d\Omega \\
- \quad &\int_\Omega \left[\frac{\gamma - 1}{\rho R} \frac{\partial}{\partial x_i} \left(k \frac{\partial \zeta}{\partial x_i} \right) + \gamma v_i \frac{\partial \zeta}{\partial x_i} \right] \widetilde{\rho E} \, d\Omega
\end{aligned}
\tag{26}
$$

where

$$
g_j = \left\{ \frac{\partial}{\partial x_i} \left[\mu \left(v_j \frac{\partial \zeta}{\partial x_i} + v_i \frac{\partial \zeta}{\partial x_j} - \frac{2}{3} v_k \frac{\partial \zeta}{\partial x_k} \delta_{ij} \right) \right] - \mu \frac{\partial \zeta}{\partial x_i} \left[\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} - \frac{2}{3} \frac{\partial v_k}{\partial x_k} \delta_{ij} \right] \right\}
$$

Note the self-adjointness of the linear part. The variations of the last two integrals in eq. 20 are easily computed as

$$
\int_\Sigma \xi_i \widetilde{v}_i \, d\Sigma \tag{27}
$$

$$
\int_\Sigma \tau \widetilde{\Theta} \, d\Sigma \tag{28}
$$

where $\widetilde{\Theta} = \widetilde{T}$ or $\widetilde{\Theta} = \partial \widetilde{T} / \partial x_i n_i$. It is now completed the variation of eq. 20. Consider all the field integrals and factor out the conservative variables variation.

4.3 Adjoint equations and boundary conditions

In order to be

$$\widetilde{L} = 0 \qquad \forall \left\{ \widetilde{\rho}, \widetilde{\rho v_i}, \widetilde{\rho E} \right\}$$

the following equations are to be satisfied

$$-\frac{\gamma-1}{2} v_j v_j \frac{\partial \lambda_i}{\partial x_i} + v_i v_j \frac{\partial \lambda_i}{\partial x_j} - \left(\frac{\gamma-1}{2} V^2 - H \right) v_i \frac{\partial \zeta}{\partial x_i} + \frac{v_j(h_j+g_j)}{\rho} - \left(\frac{\gamma-1}{2\rho R} V^2 - \frac{T}{\rho} \right) \frac{\partial}{\partial x_i} \left(k \frac{\partial \zeta}{\partial x_i} \right) = 0 \tag{29}$$

$$\left[(\gamma-1) v_i v_j - H \delta_{ij} \right] \frac{\partial \zeta}{\partial x_j} - \frac{\partial \eta}{\partial x_i} - v_j \frac{\partial \lambda_i}{\partial x_j} - v_j \frac{\partial \lambda_j}{\partial x_i} + (\gamma-1) v_i \frac{\partial \lambda_j}{\partial x_j} - \frac{h_i+g_i}{\rho} + \frac{\gamma-1}{\rho R} v_i \frac{\partial}{\partial x_i} \left(k \frac{\partial \zeta}{\partial x_i} \right) = 0 \tag{30}$$

$$-(\gamma-1) \frac{\partial \lambda_i}{\partial x_i} - \gamma v_i \frac{\partial \zeta}{\partial x_i} - \frac{\gamma-1}{\rho R} \frac{\partial}{\partial x_i} \left(k \frac{\partial \zeta}{\partial x_i} \right) = 0 \tag{31}$$

These are the compressible Navier-Stokes adjoint equations. The boundary conditions to be satisfied by such equations are determined by considering the boundary terms of $\widetilde{L}$. We focus on wing-like geometries, such that in the far field the flow is considered unperturbed, and the no-slip boundary condition is applied on a finite closed surface. In this respect $\widetilde{L}$ depends on the boundary terms left in eqs. 21, 23, 26, 27, 28, once the field integrals are elided by satisfying the adjoint equations. We are left with

$$\begin{aligned} \widetilde{L} = \widetilde{I} &+ \int_\Sigma \lambda_i \widetilde{f}_i \, d\Sigma + \int_\Sigma \eta \, \widetilde{\rho v_i} n_i \, d\Sigma + \int_\Sigma \zeta \widetilde{e} \, d\Sigma - k \int_\Sigma \zeta \frac{\partial \widetilde{T}}{\partial x_i} n_i \, d\Sigma + k \int_\Sigma \widetilde{T} \frac{\partial \zeta}{\partial x_i} n_i \, d\Sigma + \int_\Sigma \tau \widetilde{\Theta} d\Sigma \\ &+ \int_\Sigma \left[\xi_i + \eta \rho n_i + \mu \left(\frac{\partial \lambda_i}{\partial x_j} n_j + \frac{\partial \lambda_j}{\partial x_i} n_j - \frac{2}{3} \frac{\partial \lambda_j}{\partial x_j} n_i + n_j \frac{\partial \zeta}{\partial x_i} v_j + n_j \frac{\partial \zeta}{\partial x_j} v_i - \frac{2}{3} n_i \frac{\partial \zeta}{\partial x_j} v_j \right) \right] \widetilde{v}_i \, d\Sigma \end{aligned} \tag{32}$$

Suppose we give a certain wall temperature. Then $\widetilde{\Theta} = \widetilde{T}$, and therefore by taking

$$\zeta = 0 \text{ and } \tau = -k \frac{\partial \zeta}{\partial x_i} n_i \qquad \textit{Prescribed wall temperature} \tag{33}$$

on Σ, the boundary integrals involving $\widetilde{T} = 0$ and $\partial \widetilde{T} \partial x_i$ are eliminated. In contrast, the adiabatic wall corresponds to $\widetilde{\Theta} = (\partial \widetilde{T} / \partial x_i) n_i$, and hence if

$$\frac{\partial \zeta}{\partial x_i} n_i = 0 \text{ and } \tau = k\zeta \qquad \textit{Adiabatic wall} \tag{34}$$

on Σ, the respective boundary integrals are eliminated as in the previous case. As $v_i = 0$ on the solid walls, we have remarkable simplifications for $\widetilde{e}$:

$$\widetilde{e} = \left\{ \rho H n_i - \mu \left[\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} - \frac{2}{3} \frac{\partial v_k}{\partial x_k} \delta_{ij} \right] n_j \right\} \widetilde{v}_i$$

Also the third term in 32 is 0 and if we take

$$\xi_i = -\left[\rho H n_i - \mu \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} - \frac{2}{3} \frac{\partial v_k}{\partial x_k} \delta_{ij} \right) n_j \right] \zeta - \mu \left(\frac{\partial \lambda_i}{\partial x_j} + \frac{\partial \lambda_j}{\partial x_i} - \frac{2}{3} \frac{\partial \lambda_k}{\partial x_k} \delta_{ij} \right) n_j - \rho n_i \eta \tag{35}$$

we are left with

$$\widetilde{L} = \widetilde{I} + \int_\Sigma \lambda_i \widetilde{f}_i \, d\Sigma \tag{36}$$

where $\widetilde{f}$ is the variation of the force per unit surface acting on the fluid at the boundary.

Such variation is unconstrained, and there are two different ways to elide the two terms left in the equation above. The first is that $\widetilde{I}$ can be put under the form

$$\widetilde{I} = \int_\Sigma J_i \widetilde{f}_i \, d\Sigma \tag{37}$$

so that taking

$$\lambda_i = -J_i \tag{38}$$

the variation of the lagrangian is finally zero. In turn this condition shows how the only allowable surface cost functionals are those involving f_i, as other functionals will not lead to $\widetilde{L} = 0$. For a deeper discussion see [11].

Otherwise, if I is a field integral then

$$\widetilde{I} = \int_\Omega J\{\widetilde{\rho}, \widetilde{\rho v_i}, \widetilde{\rho E}\} \widetilde{f} \, d\Omega \tag{39}$$

where in this case J is the derivative of the integrand with respect to the conservative variables. The three terms under integral become source terms of the adjoint equations, and by taking $\lambda_i = 0$ on Σ we are left again with $\widetilde{L} = 0$.

4.4 Gradient

As mentioned, the variation of L with respect to the Lagrange multipliers yields back the governing equations and boundary conditions, therefore the gradient is simply the variation with respect to the geometry. If I is a field integral, and if $\epsilon\,\omega$ is variation in the direction of the normal to the boundary, we have

$$\mathcal{D}I = \int_\Omega \frac{\partial F}{\partial \omega} \omega \, d\Omega + \int_\Sigma F \omega \, d\Sigma \tag{40}$$

where F is the integrand of I and ϵ is small. When I is defined on the boundary we have

$$\mathcal{D}I = \int_\Sigma \frac{\partial F}{\partial \omega} \omega \, d\Sigma + \int_\Sigma \frac{\partial F}{\partial x_i} n_i \omega \, d\Sigma + \int_\Sigma H F \, \omega \, d\Sigma \tag{41}$$

and H is the local surface curvature.

For the other terms of L the derivation rules are the same, but some simplifications occur since the governing and adjoint equations with the respective boundary conditions are satisfied on Σ. The only terms left are those relative to the boundary terms and the gradient, that is

$$\mathcal{D}L = \mathcal{D}I + \int_\Sigma \frac{\partial \xi_i v_i}{\partial x_j} n_j \omega \, d\Sigma + \int_\Sigma \frac{\partial \tau \Theta}{\partial x_i} n_i \omega \, d\Sigma \tag{42}$$

In the particular case under examination the functional to be minimized is

$$\mathcal{L} = \omega_1 D + \omega_2 \frac{(L - L^*)^2}{2} \tag{43}$$

where D is the drag (to be minimized), L is the lift, L^* is the desired lift and the ω_i are weights. Consequently, the gradient expression is:

$$\begin{aligned} \delta\mathcal{L} \;=\; & \omega_1 \int_\Sigma - \left(-\tfrac{\partial p \, n_i}{\partial y} \delta y + \tfrac{\partial \tau_{ij} \, n_j}{\partial y} \delta y \right) t_i^\infty \, d\Sigma + \\ & \omega_2 \left(L - L^* \right) \int_\Sigma - \left(-\tfrac{\partial p \, n_i}{\partial y} \delta y + \tfrac{\partial \tau_{ij} \, n_j}{\partial y} \delta y \right) n_i^\infty \, d\Sigma + \\ & \int_\Sigma \xi_i \tfrac{\partial v_i}{\partial y} \delta y \, d\Sigma + \int_\Sigma \tau \tfrac{\partial \Theta}{\partial y} \delta y \, d\Sigma \end{aligned}$$

5 Use of approximate fitness evaluators

Optimization techniques based on evolutionary computing are very attractive in terms of robustness and global extremum location capabilities, but these favorable characteristics are obtained to the expense of a rather large number of fitness evaluations. This is a problem in many engineering applications, where fitness evaluation often requires a substantial amount of computational resources. This makes attractive the use of approximate fitness evaluators, with lower computational requirements, whenever it is possible. In aerodynamic optimization design, for example, various approaches have been experimented, such as the use of Neural Network-based interpolators to reduce the number of true flow field evaluations [12]. The mixed use of solvers with different levels of approximation in a hierarchical organization has also been object of investigation. Both procedures have advantages and limits. Neural Networks, for example, are able to improve the quality of their approximation when the number of exact evaluations available increase. Their performance, therefore, may improve in real time during the optimization process. On the other hand, their more serious drawback is the fast decrease in approximation fidelity when the parameter space dimension increases [13, 14, 15, 16]. Low fidelity solvers (e.g. Euler+boundary layer versus Navier-Stokes), do not have this problem, but it is not always easy to understand the limits of the approximate model. In both cases, the problem that arises is the difficulty of devising a good and sound strategy to mix and interleave high-fidelity and low-fidelity fitness approximations.

5.1 Problem analysis

A simple analysis of the implications of mixing fitness evaluators with different levels of precision within an (evolutionary) optimization process is here carried out. This analysis is made using concepts of multi-objective optimization [17, 18, 19], like Pareto fronts and dominance criteria, even if the class of problems considered is strictly single-objective. In fact, the relations between the exact function f and its approximated model g is analyzed with the following two-objective problem:

$$\begin{cases} \min f(\mathbf{x}),\ g(\mathbf{x}, X_0) \\ \ \mathbf{x} \in X \\ \ X_0 \in \wp(X) \end{cases} \tag{44}$$

where $f(\mathbf{x}) \in \Re$ is the function defined in a domain $X \subseteq \Re^n$ for which we are interested in finding the global extrema, and $g(\mathbf{x}, X_0) \in \Re$ is defined in the domain $X^* \equiv X \cup \wp(X)$; $\wp(X)$ is the power set of X and $X_0 \in \wp(X)$ represents a sampling of the domain X. In other terms X_0 can be thought as the training set of g, e.g. the set of point in which f (and possibly its derivatives) are known and their values are used to obtain g. Let us have, for example, the following function:

$$f(x) = [x^2 - 10\cos(2\pi x) + 10] \quad x \in [-1, 3] \tag{45}$$

and let $g(x)$ a polynomial of degree 10 that minimizes $\sum\limits_{x_i \in X_0} [f(x_i) - g(x_i)]^2$ to best fit $f(x)$ in the least squares sense for a given training set X_0 (see Figure 2).

The Pareto Front related to problem (44), reported here in Figure 3 for function 45 and its approximator, is useful to understand the relationship between f and its approximating function g.

The training set X_0 is, for the moment, supposed fixed and unchanged during the optimization process. In other words, X_0 fixed means that no learning procedure is involved in the approximation process; there are, instead, two different models, from the beginning of the process, the exact and the approximate one, that can be used to search the optimum.

When using an approximated model for fitness evaluation, there are two things that have to be taken into account:

- Shift of extremum points between the two models;
- Value shift between exact and approximate functions.

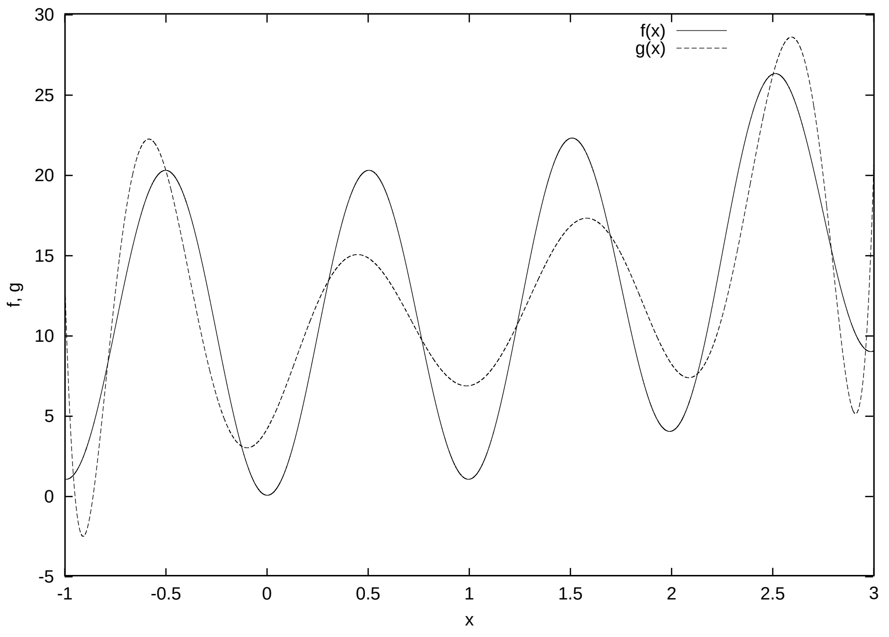


Figure 2: $f(x) = [x^2 - 10\cos(2\pi x) + 10]$ and g = best fitting polynomial of degree 10.

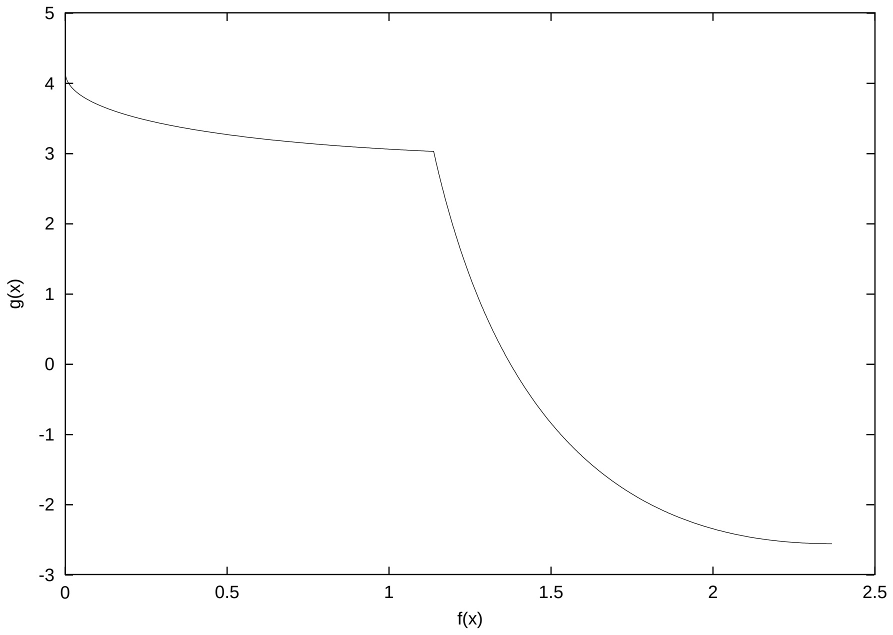


Figure 3: Pareto front of problem 44 with f specified in eq. 45 and g 10^{th} degree polynomial approximator.

The lesser are these shifts in the extremum points, the better and more reliable are optimization results obtained using the approximate model. Furthermore in these conditions it is safe to interleave exact and approximate evaluations.

A generic Pareto front of a problem of type (1) is reported in Figure 4. Four different zones can be found, that are characterized by different behavior of g and f in moving towards their global optimum points. In zone (I), a solution that dominates the initial one is surely closer to both f and g global minimum points. In zone (II), a solution that dominates the initial one is closer to the global minimum of f and farther from g global minimum. Conversely, zone (IV) has an opposite behavior, as a dominating solution leads close to g and far from f global minimum points. In zone (III), finally, a solution that dominates the initial one is farther from both global minimum points.

Given this framework, we want check if there is a particular interleaving strategy that minimizes the number of exact function evaluations. The obvious limit of this approach is that in a real optimization run the Pareto front is unknown and only a rough estimate of the collocation of a particular solution in one of the four zones is possible.

At the beginning of the optimization process the candidate solution (or population) is likely to be in zone (I). The approximated function g can be safely used here, and only occasional control is needed on f values to avoid the generation of too many non dominated solutions (relatively to f).

Zones (II) and (III) are difficult to detect if g global minimum value is unknown. In principle, use of g without check on f should be avoided in zone (II), as it may lead to unsatisfactory results in terms of f. In zone (III) both the use of g alone and of g and f interleaved is detrimental, as it would lead to a point in the Pareto front that would be likely to be far from the global optimum of f.

Zone (IV) is similar to zone (I) from the point of view of searching for the global optimum of f. However, here interleaving could be detrimental for reaching the global optimum of g.

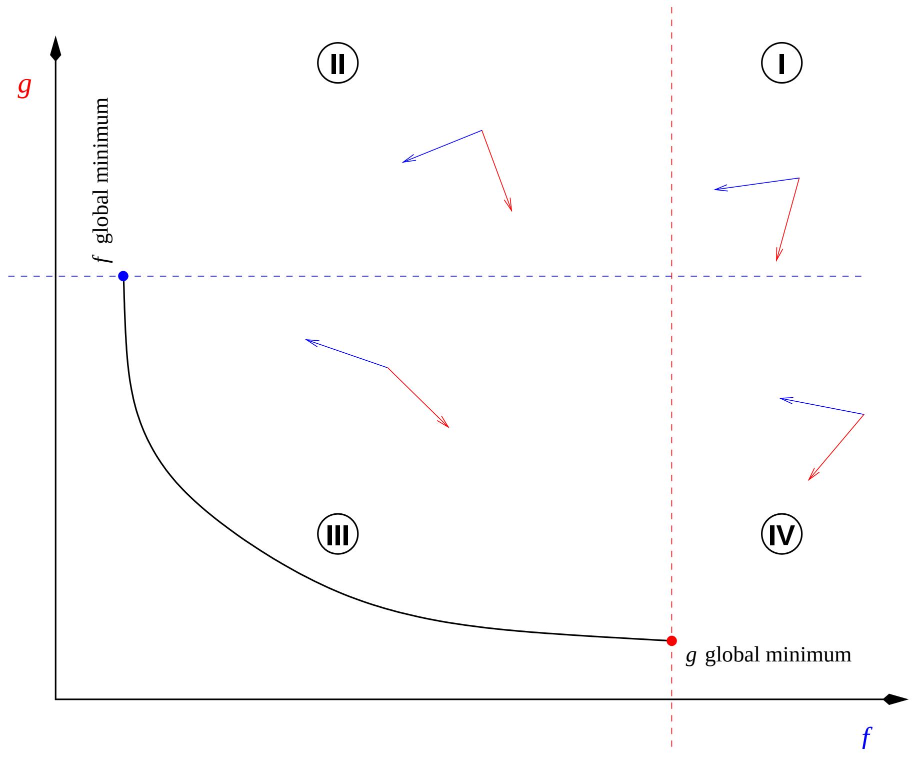


Figure 4: Generic Pareto front related to problem 44.

5.2 Possible approaches to mixing exact and approximated evaluations

From a practical standpoint the above considerations suggest the following search strategy:

1. Search the global optimum of g first, eventually checking in very few points if the values of f do not get worse too much.

2. Search the global optimum of f, eventually using g as second objective if there is interest in finding solutions belonging to the f-g front.

When X_0 changes during the optimization process, the approximation g can increase its precision when more evaluation of f are available. Therefore the Pareto front tends to decrease in size, and after some time it can reduce to a point.

Devising a good interleaving strategy is here more difficult, however the previous strategy, adapted as follows, should have an acceptably good behavior:

1. f is sampled in some points and the resulting values (training set) are used to find a first expression of g.
2. The global optimum of g is searched.
3. A short two-objective (f-g) run is performed and the computed values of f are added to the training set.
4. A new expression of g is found.
5. The process is repeated starting from step 2 until a satisfactory value of f is found.

5.3 Analysis of an example problem

The RAE 2822 airfoil is assigned as starting configuration, and the optimization goal is the reduction of the drag coefficient c_d at given mach ($M = 0.78$) and lift coefficient $c_l = 0.60$. The maximum airfoil thickness has to remain unchanged.

A two-objective optimization run is carried out using two different flow solvers:

1. Euler + Boundary layer, interactive (high fidelity solver, objective f).
2. Full potential, non-conservative formulation (low fidelity solver, objective g).

It is important to observe that these flow solvers represent the real flow at a very different approximation level, and that a real optimization application would benefit more from the use of more closely related flow solver models, such as Navier-Stokes and Euler + Boundary layer. The only aim of this run is the analysis of the relationships between different flow models.

The aerofunctions obtained from the airfoils reported in figure 5 were used to modify the airfoil shape.

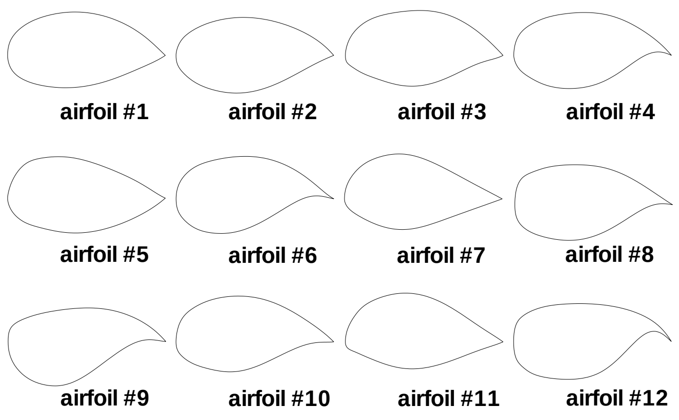


Figure 5: Airfoils used to build the modification functions.

The GA parameters are:

- bit resolution: 10
- selection method: two-step random walk with elitism
- bit-mutation with 4% rate
- population size: 20
- generations: 20

Figure 6 reports the Pareto front obtained in the two-objective run. As it can be observed, the Pareto front shows a sensible difference between the two flow models. These solvers should, therefore, not be used in an interleaved fashion unless a viscous correction is introduced in the full potential one. However, a hierarchical use of these solver could be useful to discard very bad solution such as those with a strong shock wave.

The above example may help in tracing the guidelines for the implementation of the hierarchical approach in an optimizer for viscous flows.

The optimization could be made in two different steps: a pre-optimization phase aimed at the reduction of the search space complexity, and a subsequent optimization phase that explores this reduced space using a high fidelity fitness evaluator.

The number of active variables, their variation ranges and a new starting configuration are a possible output of the first phase. In a more sophisticated approach, using for example a multi-objective GA, the pre-optimization can be used to generate an aerofunction basis with a reduced number of elements.

In the second phase, the modification of the geometry could be driven by a very simple evolution strategy, such as a (1,1)-ES [20, 21] that, due reduction of the search space complexity, should give an improvement of the objective function in an acceptable number of attempts.

6 Conclusions

It has been shown how computer algebra techniques can be used to obtain the adjoint equations and pertinent boundary conditions of the Navier-Stokes equations. The obtained equations can be directly used for the adjoint equations solver coding.

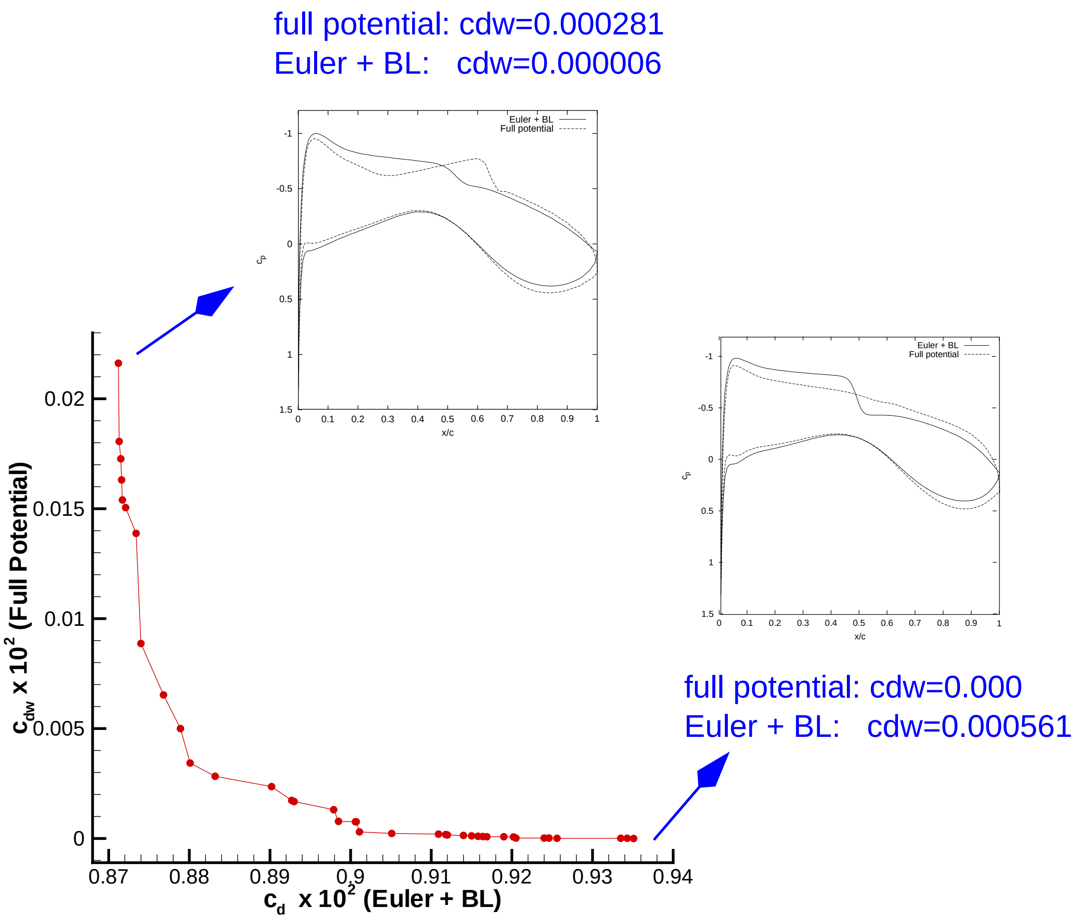


Figure 6: Euler + Boundary Layer — Full Potential comparison.

MAPLE was used as computer algebra language, complemented with an original code taking advantage of public domain libraries for tensor algebra and function variation computation.

Further development will include derivation of the adjoint equations of the Reynolds averaged Navier-Stokes equations with κ-ϵ turbulence modeling.

The relationship between an objective function and an approximated form has been investigated using a Pareto front to show the trade-off between exact and expensive objective function evaluations and inexpensive but inexact approximations. Approximating the objective function values, if properly done, shortens the time in time-consuming design problems like those typical of CFD applications. Evolutionary algorithms normally require more objective function evaluations than other methods and, therefore, are very likely to take advantage by using such approximation techniques.

Acknowledgments

Part of the work here presented was carried out within the framework of the AEROSHAPE project. This project (Multi-Point Aerodynamic Shape Optimization) is a collaboration between Aerospatiale Matra Airbus, Alenia Aeronautica (Coordinator), Daimler Crysler Airbus, EADS-M, Dassault Aviation, SAAB, SENER, SYNAPS, CIRA, DERA, DLR, FFA, INRIA, HCSA, NLR, ONERA, and NuTech Solutions. The project is funded by the European Commission, DG Research, under the GROWTH initiative (Project Ref.: GRD1-1999-10752).

A Example of adjoint NS derivation using MAPLE

For the sake of clarity and conciseness, the output of MAPLE has been slightly modified such that indexed variables and derivatived such as $x1$ and $v3_{x2}$ appear as x_1 and $v3_{x_2}$. Furthermore, we will work only on a piece of the functional 20, namely the scalar products between the adjoint variables and the state equations. Therefore, in the output will not appear the terms coming from I. Hence we have the following expanded form:

$$
\begin{aligned}
LG1 :=\ & \iiint \eta\,(\rho_{x_1}\, v1 + \rho\, v1_{x_1} + \rho_{x_2}\, v2 + \rho\, v2_{x_2} + \rho_{x_3}\, v3 + \rho\, v3_{x_3}) + \lambda_1(\rho_{x_1}\, {v1}^2 \\
& + 2\,\rho\, v1\, v1_{x_1} + p_{x_1} - \frac{4}{3}\,\mu\, v1_{x_1,x_1} - \frac{1}{3}\,\mu\, v2_{x_1,x_2} - \frac{1}{3}\,\mu\, v3_{x_1,x_3} + \rho_{x_2}\, v2\, v1 + \rho\, v2_{x_2}\, v1 \\
& + \rho\, v2\, v1_{x_2} - \mu\, v1_{x_2,x_2} + \rho_{x_3}\, v3\, v1 + \rho\, v3_{x_3}\, v1 + \rho\, v3\, v1_{x_3} - \mu\, v1_{x_3,x_3}) + \lambda_2 \\
& (\rho_{x_1}\, v2\, v1 + \rho\, v2_{x_1}\, v1 + \rho\, v2\, v1_{x_1} - \mu\, v2_{x_1,x_1} - \frac{1}{3}\,\mu\, v1_{x_1,x_2} + \rho_{x_2}\, {v2}^2 + 2\,\rho\, v2\, v2_{x_2} \\
& + p_{x_2} - \frac{4}{3}\,\mu\, v2_{x_2,x_2} - \frac{1}{3}\,\mu\, v3_{x_2,x_3} + \rho_{x_3}\, v3\, v2 + \rho\, v3_{x_3}\, v2 + \rho\, v3\, v2_{x_3} - \mu\, v2_{x_3,x_3}) + \\
& \lambda_3(\rho_{x_1}\, v3\, v1 + \rho\, v3_{x_1}\, v1 + \rho\, v3\, v1_{x_1} - \mu\, v3_{x_1,x_1} - \frac{1}{3}\,\mu\, v1_{x_1,x_3} + \rho_{x_2}\, v3\, v2 \\
& + \rho\, v3_{x_2}\, v2 + \rho\, v3\, v2_{x_2} - \mu\, v3_{x_2,x_2} - \frac{1}{3}\,\mu\, v2_{x_2,x_3} + \rho_{x_3}\, {v3}^2 + 2\,\rho\, v3\, v3_{x_3} + p_{x_3} \\
& - \frac{4}{3}\,\mu\, v3_{x_3,x_3}) + \zeta(-\frac{1}{3}\, v3\,\mu\, v1_{x_1,x_3} - \frac{4}{3}\, {v3_{x_3}}^2\,\mu - {v2_{x_3}}^2\,\mu + \rho_{x_3}\, v3\, H + \rho\, v3_{x_3}\, H \\
& + \rho\, v3\, H_{x_3} - k\, T_{x_3,x_3} - {v1_{x_3}}^2\,\mu - k\, T_{x_1,x_1} - \frac{1}{3}\, v2\,\mu\, v1_{x_1,x_2} - k\, T_{x_2,x_2} - \frac{1}{3}\, v2\,\mu\, v3_{x_2,x_3} \\
& - v1\,\mu\, v1_{x_2,x_2} - \frac{4}{3}\, v2\,\mu\, v2_{x_2,x_2} - v3\,\mu\, v3_{x_1,x_1} - \frac{1}{3}\, v1\,\mu\, v2_{x_1,x_2} - \frac{4}{3}\, v3\,\mu\, v3_{x_3,x_3} \\
& - \frac{1}{3}\, v3\,\mu\, v2_{x_2,x_3} - v1\,\mu\, v1_{x_3,x_3} - v3\,\mu\, v3_{x_2,x_2} - \frac{4}{3}\, {v1_{x_1}}^2\,\mu + \rho\, v1\, H_{x_1} + \rho\, v1_{x_1}\, H \\
& + \rho_{x_1}\, v1\, H - {v3_{x_1}}^2\,\mu - v2\,\mu\, v2_{x_3,x_3} - 2\, v2_{x_1}\,\mu\, v1_{x_2} - {v2_{x_1}}^2\,\mu + \frac{4}{3}\, v1_{x_1}\,\mu\, v3_{x_3} \\
& + \frac{4}{3}\, v1_{x_1}\,\mu\, v2_{x_2} - v2\,\mu\, v2_{x_1,x_1} - 2\, v3_{x_1}\,\mu\, v1_{x_3} - \frac{4}{3}\, v1\,\mu\, v1_{x_1,x_1} - \frac{1}{3}\, v1\,\mu\, v3_{x_1,x_3} \\
& - \frac{4}{3}\, {v2_{x_2}}^2\,\mu - {v1_{x_2}}^2\,\mu + \rho\, v2\, H_{x_2} + \rho\, v2_{x_2}\, H + \rho_{x_2}\, v2\, H - {v3_{x_2}}^2\,\mu + \frac{4}{3}\, v2_{x_2}\,\mu\, v3_{x_3} \\
& - 2\, v3_{x_2}\,\mu\, v2_{x_3})dx_1dx_2dx_3 + \iint \xi_1\, v1 + \xi_2\, v2 + \xi_3\, v3 + \tau\,\Theta(T)\, ds_1\, ds_2
\end{aligned}
$$

Now the perfect gas model is introduced in order to close the equation system:

```
>  pexp := p = -1/2*rho*(gamma-1)*(-2*H+v1^2+v2^2+v3^2)/gamma;
```

$$pexp := p = -\frac{1}{2}\,\frac{\rho\,(\gamma-1)\,(-2\,H + {v1}^2 + {v2}^2 + {v3}^2)}{\gamma}$$

```
>  Texp := T = -1/2*(gamma-1)*(-2*H+v1^2+v2^2+v3^2)/(gamma*R);
```

$$Texp := T = -\frac{1}{2}\,\frac{(\gamma-1)\,(-2\,H + {v1}^2 + {v2}^2 + {v3}^2)}{\gamma\,R}$$

```
>  LG1:=subs({pexp,Texp}, LG1):
```

The prescribed boundary temperature condition is imposed:

```
>  Theta := (w) -> w- Tw(x1,x2,x3,s1,s2);
```

$$\Theta := w \to w - \mathrm{Tw}(x1,\, x2,\, x3,\, s1,\, s2)$$

The variation with respect to ρ gives the first adjoint equation:

```
>  adj_drho:=compute_adj_eq(LG1, rho, drho, 2, 0, true);
```

$$-v_2\,H\,\zeta_{x_2}-\frac{1}{2}\,\frac{v_3{}^2\,\lambda_{3\,x_3}}{\gamma}-v_2\,v_3\,\lambda_{3\,x_2}-v_2\,v_3\,\lambda_{2\,x_3}+\frac{H\,\lambda_{3\,x_3}}{\gamma}+\frac{H\,\lambda_{2\,x_2}}{\gamma}-\eta_{x_3}\,v_3+\frac{\lambda_{1\,x_1}\,H}{\gamma}$$
$$-\lambda_{1\,x_1}\,H-\eta_{x_1}\,v_1-H\,\lambda_{3\,x_3}-\eta_{x_2}\,v_2+\frac{1}{2}\,v_1{}^2\,\lambda_{3\,x_3}-\frac{1}{2}\,v_1{}^2\,\lambda_{1\,x_1}+\frac{1}{2}\,v_1{}^2\,\lambda_{2\,x_2}$$
$$+\frac{1}{2}\,v_3{}^2\,\lambda_{2\,x_2}+\frac{1}{2}\,\lambda_{1\,x_1}\,v_3{}^2+\frac{1}{2}\,\lambda_{1\,x_1}\,v_2{}^2+\frac{1}{2}\,v_2{}^2\,\lambda_{3\,x_3}-\frac{1}{2}\,v_2{}^2\,\lambda_{2\,x_2}-\frac{1}{2}\,v_3{}^2\,\lambda_{3\,x_3}-H\,\lambda_{2\,x_2}$$
$$-v_1\,H\,\zeta_{x_1}-\frac{1}{2}\,\frac{\lambda_{1\,x_1}\,v_2{}^2}{\gamma}-v_1\,v_3\,\lambda_{1\,x_3}-\frac{1}{2}\,\frac{v_1{}^2\,\lambda_{3\,x3}}{\gamma}-\frac{1}{2}\,\frac{v_3{}^2\,\lambda_{2\,x_2}}{\gamma}-v_1\,v_3\,\lambda_{3\,x_1}$$
$$-\frac{1}{2}\,\frac{v_1{}^2\,\lambda_{2\,x_2}}{\gamma}-\frac{1}{2}\,\frac{v_1{}^2\,\lambda_{1\,x_1}}{\gamma}-\zeta_{x_3}\,v_3\,H-\frac{1}{2}\,\frac{v_2{}^2\,\lambda_{3\,x_3}}{\gamma}-v_1\,v_2\,\lambda_{1\,x_2}-v_1\,v_2\,\lambda_{2\,x_1}$$
$$-\frac{1}{2}\,\frac{\lambda_{1\,x_1}\,v_3{}^2}{\gamma}-\frac{1}{2}\,\frac{v_2{}^2\,\lambda_{2\,x_2}}{\gamma}$$

Now the adjoint equations coming from the variations with respect to v_1, v_2, v_3:

```
> adj_dv1:=compute_adj_eq(LG1, v1, dv1, 2, 0, true);
```

$$-\rho\,H\,\zeta_{x_1}+\frac{5}{3}\,\mu\,v_{2\,x_1}\,\zeta_{x_2}-\mu\,\zeta_{x_3,\,x_3}\,v_1-\frac{4}{3}\,\mu\,v_1\,\zeta_{x_1,\,x_1}-\mu\,\lambda_{1\,x2,\,x_2}-\rho\,v_3\,\lambda_{1\,x_3}$$
$$+\rho\,v_1\,\lambda_{2\,x_2}-\rho\,v_3\,\lambda_{3\,x_1}-\frac{\rho\,v_1\,\lambda_{1\,x_1}}{\gamma}-\frac{\rho\,v_1\,\lambda_{2\,x_2}}{\gamma}-\frac{\rho\,v_1\,\lambda_{3\,x_3}}{\gamma}+\frac{k\,v_1\,\zeta_{x_1,\,x_1}}{R}$$
$$+\frac{k\,v_1\,\zeta_{x_2,\,x_2}}{R}+\frac{k\,\zeta_{x_3,\,x_3}\,v_1}{R}-\eta_{x1}\,\rho-\frac{5}{3}\,\mu\,v_{3\,x_3}\,\zeta_{x_1}-\frac{5}{3}\,\mu\,\zeta_{x_1}\,v_{2\,x_2}$$
$$-\frac{1}{3}\,\mu\,v_3\,\zeta_{x_1,\,x_3}+\frac{5}{3}\,\mu\,v_{3\,x_1}\,\zeta_{x_3}-\frac{4}{3}\,\mu\,\lambda_{1\,x_1,\,x_1}+\rho\,v_1\,\lambda_{3\,x_3}-\frac{1}{3}\,\mu\,\lambda_{2\,x_1,\,x_2}-\rho\,v_2\,\lambda_{1\,x2}$$
$$-\frac{1}{3}\,\mu\,v_2\,\zeta_{x_1,\,x_2}-\frac{1}{3}\,\mu\,\lambda_{3\,x_1,\,x_3}-\rho\,v_1\,\lambda_{1\,x_1}-\mu\,v_1\,yH_{x_2,\,x_2}-\mu\,\lambda_{1\,x_3,\,x_3}-\frac{k\,\zeta_{x_3,\,x_3}\,v_1}{\gamma\,R}$$
$$-\frac{k\,v_1\,\zeta_{x_1,\,x_1}}{\gamma\,R}-\frac{k\,v_1\,\zeta_{x_2,\,x_2}}{\gamma\,R}-\rho\,v_2\,\lambda_{2\,x_1}$$

```
> adj_dv2:=compute_adj_eq(LG1, v2, dv2, 2, 0, true);
```

$$-\mu\,\zeta_{x_1,\,x_1}\,v_2-\frac{1}{3}\,\mu\,v_3\,\zeta_{x_2,\,x_3}+\rho\,v_2\,\lambda_{3\,x_3}+\rho\,v_2\,\lambda_{1\,x_1}+\frac{5}{3}\,\mu\,v_{3\,x_2}\,\zeta_{x_3}+\frac{5}{3}\,\mu\,v_{1\,x_2}\,\zeta_{x_1}$$
$$-\mu\,\lambda_{2\,x_1,\,x_1}-\frac{\rho\,v_2\,\lambda_{1\,x_1}}{\gamma}-\frac{\rho\,v_2\,\lambda_{3\,x_3}}{\gamma}-\frac{\rho\,v_2\,\lambda_{2\,x_2}}{\gamma}-\eta_{x_2}\,\rho-\frac{4}{3}\,\mu\,\lambda_{2\,x_2,\,x_2}$$
$$-\frac{k\,\zeta_{x_1,\,x_1}\,v_2}{\gamma\,R}-\frac{k\,v_2\,\zeta_{x_3,\,x_3}}{\gamma\,R}-\frac{k\,v_2\,\zeta_{x_2,\,x_2}}{\gamma\,R}-\frac{1}{3}\,\mu\,\lambda_{3\,x_2,\,x_3}-\frac{4}{3}\,\mu\,v_2\,\zeta_{x_2,\,x_2}$$
$$-\rho\,v_3\,\lambda_{2\,x_3}-\frac{5}{3}\,\mu\,\zeta_{x_2}\,v_{1\,x_1}-\rho\,v_2\,\lambda_{2\,x_2}-\mu\,v_2\,\zeta_{x_3,\,x_3}-\rho\,H\,\zeta_{x_2}-\frac{1}{3}\,\mu\,v_1\,\zeta_{x_1,\,x_2}$$
$$-\rho\,v_3\,\lambda_{3\,x_2}-\rho\,v_1\,\lambda_{2\,x_1}-\frac{1}{3}\,\mu\,\lambda_{1\,x_1,\,x_2}-\rho\,v_1\,\lambda_{1\,x_2}+\frac{k\,v_2\,\zeta_{x_3,\,x_3}}{R}+\frac{k\,v_2\,\zeta_{x_2,\,x_2}}{R}$$
$$+\frac{k\,\zeta_{x_1,\,x_1}\,v_2}{R}-\mu\,\lambda_{2\,x_3,\,x_3}-\frac{5}{3}\,\mu\,v_{3\,x_3}\,\zeta_{x_2}$$

```
> adj_dv3:=compute_adj_eq(LG1, v3, dv3, 2, 0, true);
```

$$
\begin{aligned}
&-\frac{1}{3}\,\mu\,\zeta_{x_1,x_3}\,v_1-\frac{1}{3}\,\mu\,\lambda_{2\,x_2,x_3}-\rho\,v_2\,\lambda_{2\,x_3}-\rho\,v_3\,\lambda_{3\,x_3}-\rho\,v_1\,\lambda_{1\,x3}-\rho\,H\,\zeta_{x_3}-\frac{1}{3}\,\mu\,\lambda_{1\,x_1,x_3}\\
&-\mu\,\lambda_{3\,x_1,x_1}-\mu\,\lambda_{3\,x_2,x_2}-\eta_{x_3}\,\rho-\frac{\rho\,v_3\,\lambda_{2\,x_2}}{\gamma}-\frac{\rho\,v_3\,\lambda_{1\,x_1}}{\gamma}-\frac{\rho\,v_3\,\lambda_{3\,x_3}}{\gamma}\\
&-\frac{4}{3}\,\mu\,\lambda_{3\,x_3,x_3}-\frac{k\,\zeta_{x_2,x_2}\,v_3}{\gamma\,R}-\frac{k\,\zeta_{x_1,x_1}\,v_3}{\gamma\,R}+\frac{k\,v_3\,\zeta_{x_3,x_3}}{R}+\frac{k\,\zeta_{x_1,x_1}\,v_3}{R}\\
&+\frac{k\,\zeta_{x_2,x_2}\,v_3}{R}-\frac{k\,v_3\,\zeta_{x_3,x_3}}{\gamma\,R}-\rho\,v_2\,\lambda_{3\,x_2}+\frac{5}{3}\,\mu\,v_{1\,x_3}\,\zeta_{x_1}-\frac{5}{3}\,\mu\,\zeta_{x_3}\,v_{1\,x_1}\\
&-\rho\,v_1\,\lambda_{3\,x_1}+\rho\,v_3\,\lambda_{1\,x_1}-\frac{5}{3}\,\mu\,\zeta_{x_3}\,v_{2\,x_2}+\frac{5}{3}\,\mu\,\zeta_{x_2}\,v_{2\,x_3}+\rho\,v_3\,\lambda_{2\,x_2}-\mu\,\zeta_{x_2,x_2}\,v_3\\
&-\frac{4}{3}\,\mu\,v_3\,\zeta_{x_3,x_3}-\mu\,\zeta_{x_1,x_1}\,v_3-\frac{1}{3}\,\mu\,v_2\,\zeta_{x_2,x_3}
\end{aligned}
$$

Finally the variation with respect to H gives:

```
>  adj_dH:=compute_adj_eq(LG1, H, dH, 2, 0, true);
```

$$
\begin{aligned}
&-\rho\,\lambda_{3\,x_3}+\frac{k\,\zeta_{x_1,x_1}}{\gamma\,R}-\rho\,v_2\,\zeta_{x_2}-\rho\,\lambda_{2\,x_2}-\rho\,v_1\,\zeta_{x_1}-\rho\,v_3\,\zeta_{x_3}+\frac{k\,\zeta_{x_3,x_3}}{\gamma\,R}+\frac{k\,\zeta_{x_2,x_2}}{\gamma\,R}\\
&+\frac{\rho\,\lambda_{1\,x_1}}{\gamma}-\frac{k\,\zeta_{x_1,x_1}}{R}-\frac{k\,\zeta_{x_3,x_3}}{R}+\frac{\rho\,\lambda_{2\,x_2}}{\gamma}-\frac{k\,\zeta_{x_2,x_2}}{R}-\rho\,\lambda_{1\,x_1}+\frac{\rho\,\lambda_{3\,x_3}}{\gamma}
\end{aligned}
$$

These equations can then be combined in order to obtain the variation related to the conservative variables $\rho, \rho v_i, \rho E$.

The same process is followed to obtain the boundary conditions. However, in this case there is a further complication, because the equations have terms containing the variations $(\delta\rho, \delta v_i, \delta H)$. See sub-section 4.3, and eq. 38 for a proper treatment of such terms.

B Formulas for functional variations

B.1 Variation of surface integrals

In the following section the variation of a surface integral (see eq. 42) with respect to surface changes is explicitly derived, in a form useful for computer algebra. Let

$$S = x(u,v)\,\mathbf{i} + y(u,v)\,\mathbf{j} + z(u,v)\,\mathbf{k} \tag{46}$$

be a parametric representation of the integration surface. The surface integral

$$\iint_S f(x,y,z)\,d\sigma$$

can be expressed as a double integral as follows:

$$\int_a^b \int_{\gamma(v)}^{\delta(v)} f(x(u,v), y(u,v), z(u,v)) W(u,v)\,du\,dv,$$

where

$$W(u,v) = \sqrt{J_1^2 + J_2^2 + J_3^2}$$

with

$$J_1 = \left|\frac{\partial(y,z)}{\partial(u,v)}\right| \quad J_2 = \left|\frac{\partial(z,x)}{\partial(u,v)}\right| \quad J_3 = \left|\frac{\partial(x,y)}{\partial(u,v)}\right|.$$

Let

$$S' = [x(u,v) + h_1(u,v,\mathbf{a})]\,\mathbf{i} + [y(u,v) + h_2(u,v,\mathbf{a})]\,\mathbf{j} + [z(u,v) + h_3(u,v,\mathbf{a})]\,\mathbf{k}$$

an arbitrary variation of S controlled by a generic parameter vector $\mathbf{a}$. Now we can write:

$$F(\mathbf{a}) = \int_a^b \int_{\gamma(v)}^{\delta(v)} f(x + h_1, y + h_2, z + h_3) W(u, v, \mathbf{a})\, du\, dv.$$

The integration boundaries are unchanged and independent on $\mathbf{a}$. Therefore only the variation of the integrand has to be considered when computing the derivative with respect to a generic element α of $\mathbf{a}$:

$$\begin{aligned}\frac{\partial F(\mathbf{a})}{\partial \alpha} &= \int_a^b \int_{\gamma(v)}^{\delta(v)} \nabla f \cdot \frac{\partial \mathbf{h}(\mathbf{a})}{\partial \alpha} W(u, v, \mathbf{a})\, du\, dv + \\ &\quad \int_a^b \int_{\gamma(v)}^{\delta(v)} f \frac{W(u, v, \mathbf{a})}{\partial \alpha}\, du\, dv.\end{aligned}$$

If f depends explicitly on $\mathbf{a}$, then a further term has to be considered, and the above equation becomes:

$$\begin{aligned}\frac{\partial F(\mathbf{a})}{\partial \alpha} &= \int_a^b \int_{\gamma(v)}^{\delta(v)} \nabla f \cdot \frac{\partial \mathbf{h}(\mathbf{a})}{\partial \alpha} W(u, v, \mathbf{a})\, du\, dv + \\ &\quad \int_a^b \int_{\gamma(v)}^{\delta(v)} f \frac{\partial W(u, v, \mathbf{a})}{\partial \alpha}\, du\, dv + \int_a^b \int_{\gamma(v)}^{\delta(v)} \frac{\partial f}{\partial \alpha} W(u, v, \mathbf{a})\, du\, dv.\end{aligned}$$

Considering that

$$\frac{\partial W(u, v, \mathbf{a})}{\partial \alpha} = \frac{J_1 J_{1_\alpha} + J_2 J_{2_\alpha} + J_3 J_{3_\alpha}}{\sqrt{J_1^2 + J_2^2 + J_3^2}} = \frac{J_1 J_{1_\alpha} + J_2 J_{2_\alpha} + J_3 J_{3_\alpha}}{J_1^2 + J_2^2 + J_3^2} W = HW$$

we finally have:

$$\begin{aligned}\frac{\partial F(\mathbf{a})}{\partial \alpha} &= \int_a^b \int_{\gamma(v)}^{\delta(v)} \nabla f \cdot \frac{\partial \mathbf{h}(\mathbf{a})}{\partial \alpha} W(u, v, \mathbf{a})\, du\, dv + \\ &\quad \int_a^b \int_{\gamma(v)}^{\delta(v)} f H(u, v, \mathbf{a}) W(u, v, \mathbf{a})\, du\, dv + \int_a^b \int_{\gamma(v)}^{\delta(v)} \frac{\partial f}{\partial \alpha} W(u, v, \mathbf{a})\, du\, dv.\end{aligned}$$

It can be shown (see next sub-section) that H does not depend on the particular parametric representation chosen, and it is therefore possible to write, finally:

$$\frac{\partial F(\mathbf{a})}{\partial \alpha} = \iint_S \nabla f \cdot \frac{\partial \mathbf{h}}{\partial \alpha}\, d\sigma + \iint_S f H\, d\sigma + \iint_S f_\alpha\, d\sigma.$$

In terms of variation of $\mathbf{h}$ the above formula can be expressed as:

$$\delta F = \iint_S \nabla f \cdot \delta \mathbf{h}\, d\sigma + \iint_S f \delta \boldsymbol{\mathcal{H}}_{\mathbf{h}}\, d\sigma + \iint_S \delta f_{\mathbf{h}}\, d\sigma$$

with

$$\mathcal{H}_j = \frac{J_i \delta J_{i h_j}}{J_1^2 + J_2^2 + J_3^2}.$$

Finally, in terms of variation of a generic function g, we have:

$$\delta F = \iint_S \nabla f \cdot \frac{\partial \mathbf{h}}{\partial g} \delta g\, d\sigma + \iint_S f \boldsymbol{\mathcal{H}} \cdot \delta \boldsymbol{\mathcal{H}}_g\, d\sigma + \iint_S \frac{\partial f}{\partial g} \delta g\, d\sigma \tag{47}$$

B.2 Invariance of $\mathcal{H}$ with the parametric surface representation

To demonstrate the invariance of $\mathcal{H}_j$ it is sufficient to show that

$$\frac{J_i(s,t)}{J_i(u,v)} = u_s v_t - u_t v_s$$

and

$$\frac{\dfrac{\partial J_i(s,t,\mathbf{h})}{\partial h_j}}{\dfrac{\partial J_i(u,v,\mathbf{h})}{\partial h_j}} = u_s v_t - u_t v_s$$

This can be achieved with simple algebraic passages:

$$\frac{\partial(y,z)}{\partial(u,v)} = \begin{bmatrix} \frac{\partial}{\partial u}\,\mathrm{y}(u,v) & \frac{\partial}{\partial v}\,\mathrm{y}(u,v) \\ \frac{\partial}{\partial u}\,\mathrm{z}(u,v) & \frac{\partial}{\partial v}\,\mathrm{z}(u,v) \end{bmatrix}$$

$$\frac{\partial(u,v)}{\partial(s,t)} = \begin{bmatrix} \frac{\partial}{\partial s}\,\mathrm{u}(s,t) & \frac{\partial}{\partial t}\,\mathrm{u}(s,t) \\ \frac{\partial}{\partial s}\,\mathrm{v}(s,t) & \frac{\partial}{\partial t}\,\mathrm{v}(s,t) \end{bmatrix}$$

$$\frac{\partial(y,z)}{\partial(s,t)} = \frac{\partial(y,z)}{\partial(u,v)} \cdot \frac{\partial(u,v)}{\partial(s,t)} = \begin{bmatrix} \frac{\partial}{\partial u}\,\mathrm{y}\,\frac{\partial}{\partial s}\,\mathrm{u} + \frac{\partial}{\partial v}\,\mathrm{y}\,\frac{\partial}{\partial s}\,\mathrm{v} & \frac{\partial}{\partial u}\,\mathrm{y}\,\frac{\partial}{\partial t}\,\mathrm{u} + \frac{\partial}{\partial v}\,\mathrm{y}\,\frac{\partial}{\partial t}\,\mathrm{v} \\ \frac{\partial}{\partial u}\,\mathrm{z}\,\frac{\partial}{\partial s}\,\mathrm{u} + \frac{\partial}{\partial v}\,\mathrm{z}\,\frac{\partial}{\partial s}\,\mathrm{v} & \frac{\partial}{\partial u}\,\mathrm{z}\,\frac{\partial}{\partial t}\,\mathrm{u} + \frac{\partial}{\partial v}\,\mathrm{z}\,\frac{\partial}{\partial t}\,\mathrm{v} \end{bmatrix}$$

$$\frac{J_i(s,t)}{J_i(u,v)} = -\frac{\partial}{\partial s}\,\mathrm{v}\,\frac{\partial}{\partial t}\,\mathrm{u} + \frac{\partial}{\partial s}\,\mathrm{u}\,\frac{\partial}{\partial t}\,\mathrm{v}$$

$$\frac{\partial J_1(u,v)}{\partial h_j} = \frac{\partial^2}{\partial u\,\partial h_j}\,\mathrm{y}\,\frac{\partial}{\partial v}\,\mathrm{z} + \frac{\partial}{\partial u}\,\mathrm{y}\,\frac{\partial^2}{\partial v\,\partial h_j}\,\mathrm{z} - \frac{\partial^2}{\partial v\,\partial h_j}\,\mathrm{y}\,\frac{\partial}{\partial u}\,\mathrm{z} - \frac{\partial}{\partial v}\,\mathrm{y}\,\frac{\partial^2}{\partial u\,\partial h_j}\,\mathrm{z}$$

$$\frac{\partial\left(\frac{\partial y}{\partial h_j}, z\right)}{\partial(u,v)} = \begin{bmatrix} \frac{\partial^2}{\partial u\,\partial h_j}\,\mathrm{y} & \frac{\partial^2}{\partial v\,\partial h_j}\,\mathrm{y} \\ \frac{\partial}{\partial u}\,\mathrm{z} & \frac{\partial}{\partial v}\,\mathrm{z} \end{bmatrix}$$

$$\frac{\partial\left(y, \frac{\partial z}{\partial h_j}\right)}{\partial(u,v)} = \begin{bmatrix} \frac{\partial}{\partial u}\,\mathrm{y} & \frac{\partial}{\partial v}\,\mathrm{y} \\ \frac{\partial^2}{\partial u\,\partial h_j}\,\mathrm{z} & \frac{\partial^2}{\partial v\,\partial h_j}\,\mathrm{z} \end{bmatrix}$$

$$\frac{\partial J_i(s,t,\mathbf{h})}{\partial h_j} = \left|\frac{\partial(y_{h_j},z)}{\partial(u,v)} \cdot \frac{\partial(u,v)}{\partial(s,t)}\right| + \left|\frac{\partial(y,z_{h_j})}{\partial(u,v)} \cdot \frac{\partial(u,v)}{\partial(s,t)}\right| =$$
$$\frac{\partial^2}{\partial u\,\partial h_j}\,\mathrm{y}\,\frac{\partial}{\partial s}\,\mathrm{u}\,\frac{\partial}{\partial v}\,\mathrm{z}\,\frac{\partial}{\partial t}\,\mathrm{v} + \frac{\partial^2}{\partial v\,\partial h_j}\,\mathrm{y}\,\frac{\partial}{\partial s}\,\mathrm{v}\,\frac{\partial}{\partial u}\,\mathrm{z}\,\frac{\partial}{\partial t}\,\mathrm{u} - \frac{\partial^2}{\partial u\,\partial h_j}\,\mathrm{y}\,\frac{\partial}{\partial t}\,\mathrm{u}\,\frac{\partial}{\partial v}\,\mathrm{z}\,\frac{\partial}{\partial s}\,\mathrm{v} - \frac{\partial^2}{\partial v\,\partial h_j}\,\mathrm{y}\,\frac{\partial}{\partial t}\,\mathrm{v}\,\frac{\partial}{\partial u}\,\mathrm{z}\,\frac{\partial}{\partial s}\,\mathrm{u} +$$
$$\frac{\partial}{\partial u}\,\mathrm{y}\,\frac{\partial}{\partial s}\,\mathrm{u}\,\frac{\partial^2}{\partial v\,\partial h_j}\,\mathrm{z}\,\frac{\partial}{\partial t}\,\mathrm{v} + \frac{\partial}{\partial v}\,\mathrm{y}\,\frac{\partial}{\partial s}\,\mathrm{v}\,\frac{\partial^2}{\partial u\,\partial h_j}\,\mathrm{z}\,\frac{\partial}{\partial t}\,\mathrm{u} - \frac{\partial}{\partial u}\,\mathrm{y}\,\frac{\partial}{\partial t}\,\mathrm{u}\,\frac{\partial^2}{\partial v\,\partial h_j}\,\mathrm{z}\,\frac{\partial}{\partial s}\,\mathrm{v} - \frac{\partial}{\partial v}\,\mathrm{y}\,\frac{\partial}{\partial t}\,\mathrm{v}\,\frac{\partial^2}{\partial u\,\partial h_j}\,\mathrm{z}\,\frac{\partial}{\partial s}\,\mathrm{u}$$

$$\frac{\dfrac{\partial J_i(s,t,\mathbf{h})}{\partial h_j}}{\dfrac{\partial J_i(u,v,\mathbf{h})}{\partial h_j}} = -\frac{\partial}{\partial s}\,\mathrm{v}\,\frac{\partial}{\partial t}\,\mathrm{u} + \frac{\partial}{\partial s}\,\mathrm{u}\,\frac{\partial}{\partial t}\,\mathrm{v}$$

B.3 Volume integral variation

The volume integral variation is obtained making explicit reference to the reduction formulas of a volume integral into a triple integral, and applying in a straightforward way the formula for the derivative of a definite integral:

$$\int_{u(t)}^{v(t)} f(x,t)dx = \int_{u(t)}^{v(t)} \frac{\partial}{\partial t} f(x,t)dx + f(v(t),t)\frac{d}{dt}v(t) - f(u(t),t)\frac{d}{dt}u(t)$$

which, applied to a triple integral, gives:

$$\begin{aligned}
&\frac{\partial}{\partial t}\int_{a(t)}^{b(t)}\int_{\gamma(x,t)}^{\delta(x,t)}\int_{\phi(x,y,t)}^{\psi(x,y,t)} f(x,y,z,t)dz\,dy\,dx = \\
&\quad \int_{a(t)}^{b(t)}\int_{\gamma(x,t)}^{\delta(x,t)}\int_{\phi(x,y,t)}^{\psi(x,y,t)} \frac{\partial}{\partial t} f(x,y,z,t)dz\,dy\,dx \\
&\quad + \int_{a(t)}^{b(t)}\int_{\gamma(x,t)}^{\delta(x,t)} \left(\frac{\partial}{\partial t}\psi(x,y,t)\right) f(x,y,\psi(x,y,t),t)dy\,dx \\
&\quad - \int_{a(t)}^{b(t)}\int_{\gamma(x,t)}^{\delta(x,t)} \left(\frac{\partial}{\partial t}\phi(x,y,t)\right) f(x,y,\phi(x,y,t),t)dy\,dx \\
&\quad + \int_{a(t)}^{b(t)} \left(\frac{\partial}{\partial t}\delta(x,t)\right) \int_{\phi(x,\delta(x,t),t)}^{\psi(x,\delta(x,t),t)} f(x,\delta(x,t),z,t)dzdx \\
&\quad - \int_{a(t)}^{b(t)} \left(\frac{\partial}{\partial t}\gamma(x,t)\right) \int_{\phi(x,\gamma(x,t),t)}^{\psi(x,\gamma(x,t),t)} f(x,\gamma(x,t),z,t)dzdx \\
&\quad + \left(\frac{d}{dt}b(t)\right) \int_{\gamma(b(t),t)}^{\delta(b(t),t)}\int_{\phi(b(t),y,t)}^{\psi(b(t),y,t)} f(b(t),y,z,t)dz\,dy \\
&\quad - \left(\frac{d}{dt}a(t)\right) \int_{\gamma(a(t),t)}^{\delta(a(t),t)}\int_{\phi(a(t),y,t)}^{\psi(a(t),y,t)} f(a(t),y,z,t)dz\,dy
\end{aligned}$$

Introducing the deformation field $\mathbf{h}(x,y,z)$, the boundary function ψ and ϕ can be written as:

$$\psi(x,y,t) = \psi_0(x,y) + h_3(x,y,\psi_0(x,y),t)$$

$$\phi(x,y,t) = \phi_0(x,y) + h_3(x,y,\phi_0(x,y),t)$$

This allows the identity below:

$$\begin{aligned}
&\int_{a(t)}^{b(t)}\int_{\gamma(x,t)}^{\delta(x,t)} \left(\frac{\partial}{\partial t}\psi(x,y,t)\right) f(x,y,\psi(x,y,t),t)dy\,dx + \\
&- \int_{a(t)}^{b(t)}\int_{\gamma(x,t)}^{\delta(x,t)} \left(\frac{\partial}{\partial t}\phi(x,y,t)\right) f(x,y,\phi(x,y,t),t)dy\,dx = \\
&\int_{a(t)}^{b(t)}\int_{\gamma(x,t)}^{\delta(x,t)} D_4(h_3)(x,y,\psi_0(x,y),t) f(x,y,\psi(x,y,t),t)dy\,dx + \\
&- \int_{a(t)}^{b(t)}\int_{\gamma(x,t)}^{\delta(x,t)} D_4(h_3)(x,y,\phi_0(x,y),t) f(x,y,\phi(x,y,t),t)dy\,dx = \iint\limits_{+FT} \frac{\partial h_3}{\partial t} f dy\,dx
\end{aligned}$$

Reasoning in a similar way for the remaining couples of integrals, we can finally write:

$$\frac{\partial}{\partial t}\iiint\limits_{T} f\,dV = \iiint\limits_{T} \frac{\partial f}{\partial t}\,dV + \iint\limits_{+FT} \frac{\partial h_1}{\partial t} f dy\,dz + \iint\limits_{+FT} \frac{\partial h_2}{\partial t} f dz\,dx + \iint\limits_{+FT} \frac{\partial h_3}{\partial t} f dx\,dy$$

That, in terms of surface integrals, becomes:

$$\frac{\partial}{\partial t}\iiint\limits_{T} f\,dV = \iiint\limits_{T} \frac{\partial f}{\partial t}\,dV + \iint\limits_{FT} f\,\frac{\partial \mathbf{h}}{\partial t}\cdot\mathbf{n}\,d\sigma$$

Finally, in terms of variation of a generic function g, we have:

$$\delta F = \iiint\limits_{T} \frac{\partial f}{\partial g}\delta g\,dV + \iint\limits_{FT} f\,\frac{\partial \mathbf{h}}{\partial g}\cdot\mathbf{n}\delta g\,d\sigma \tag{48}$$

B.4 Application of divergence theorem to terms involving crossed derivatives

The divergence theorem

$$\iiint\limits_{\Omega} \nabla\cdot\mathbf{b}\,d\Omega = \iint\limits_{\Sigma} \mathbf{b}\cdot\mathbf{n}\,d\Sigma$$

can be applied in two different ways to transform the following volume integral

$$\iiint\limits_{\Omega} v_{xy}\,d\Omega$$

with v a generic scalar function continuous in Ω with its first- and second-order partial derivatives, such that $v_{xy} = v_{yx}$. It is, indeed, possible to write v_{xy} either as $\nabla\cdot(0, v_x, 0)$ or as $\nabla\cdot(v_y, 0, 0)$. This leads immediately to the following identity:

$$\iiint\limits_{\Omega} v_{xy}\,d\Omega = \iint\limits_{\Sigma} v_x n_2\,d\Sigma = \iint\limits_{\Sigma} v_y n_1\,d\Sigma$$

with $\mathbf{n} = (n_1, n_2, n_3)$.

This identity can be used to transform some adjoint boundary conditions. This operation, indeed, can be useful to restore symmetry in the adjoint boundary conditions automatically derived.

References

[1] M. Drela. Xfoil: An analysis and design system for low reynolds number airfoils. In *Conference on Low Reynolds Number Airfoil Aerodynamics*. University of Notre Dame, June 1989.

[2] M. Drela and M.B. Giles. Viscous-inviscid analysis of transonic and low reynolds number airfoils. *AIAA Journal*, 25(10):1347–1355, October 1987.

[3] A. Jameson. Aerodynamic design via control theory. Technical Report ICASE Report 88-64, NASA Langley Research Center, 1988.

[4] A. Jameson, L. Martinelli, and N. A. Pierce. Optimum aerodynamic design using the Navier-Stokes equations. *TCFD*, (10):213–237, 1998.

[5] A. Iollo and L. Zannetti. Trapped vortex optimal control by suction and blowing at the wall. *EJM/B Fluids*, 20:417–430, 2001.

[6] A. Iollo, M. Ferlauto, and L. Zannetti. An aerodynamic optimization method based on the inverse problem adjoint equations. *Journal of Computational Physics*, 173(1):87–115, 2001.

[7] N.A. Pierce and M.B. Giles. Adjoint recovery of superconvergent functionals from approximate solutions of partial differential equations. Technical Report Report no. 98/18, Oxford Computing Laboratory, 1998.

[8] D. Redfern. *Maple V Handbook - Release 4*, 1996. ISBN 0-387-94538-5.

[9] P. Musgrave, D. Pollney, and K. Lake. GRTensorII release 1.50 for MapleV releases 3 and 4 — introduction and overview. Technical report, Queen's University, Kingston, Ontario, July 1996.

[10] E. S. Cheb-Terrab. Maple procedures for partial and functional derivatives. *Computer Physics Communications*, 79:409–424, 1994.

[11] E. Arian and M. D. Salas. Adjoint formulation using auxiliary boundary equations. In Charles-Henri Bruneau, editor, *Lecture Notes in Physics*, volume 515, pages 355–360, Berlin Heidelberg, 1998. Springer-Verlag.

[12] A. P. Giotis, K. C. Giannakoglou, and Jacques Périaux. A reduced-cost multi-objective optimization method based on the pareto front technique, neural networks and pvm. In *Proceedings of ECCOMAS 2000 Conference*, Barcelona, Spain, September 11–14 2000.

[13] T. Poggio and F. Girosi. A theory of networks for approximation and learning. Technical Report AIM–1140, Artificial Intelligence Laboratory – MIT, 1989.

[14] T. Poggio and F. Girosi. Extension of a theory of networks for approximation and learning: Dimensionality reduction and clustering. In *Proceedings Image Understanding Workshop*, pages 597–603, Pittsburgh, Pennsylvania, September 1990.

[15] T. Poggio and F. Girosi. Hyperbf: A powerful approximation technique for learning. In P. H. Winston and S. A. Shellard, editors, *Artificial Intelligence at MIT: Expanding Frontiers*, pages 270–285, Cambridge, Massachusetts, 1990. The MIT Press.

[16] T. Poggio and F. Girosi. Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, (247):978–982, 1990.

[17] Kalyanmoy Deb. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation*, 7(3):205–230, 1999.

[18] Alessandro Vicini and Domenico Quagliarella. Inverse and direct airfoil design using a multiobjective genetic algorithm. *AIAA Journal*, 35(9):1499–1505, September 1997.

[19] Domenico Quagliarella and Alessandro Vicini. GAs for aerodynamic shape design II: multiobjective optimization and multi-criteria design. In *Lecture Series 2000-07, Genetic Algorithms for Optimisation in Aeronautics and Turbomachinery*, Belgium, May 2000. Von Karman Institute.

[20] Hans-Paul Schwefel. *Evolution and Optimum Seeking*. Sixth-Generation Computer Technology. John Wiley & Sons, New York, U.S.A., 1994.

[21] Thomas Bäck and Hans-Paul Schwefel. Evolutionary algorithms: Some very old strategies for optimization and adaptation. In D. Perret-Gallix, editor, *New Computing Techniques in Physics Research II*, Singapore, 1992. World Scientific.

This page has been deliberately left blank

Page intentionnellement blanche

Neural Networks and other Techniques for Fault Identification and Isolation of Aircraft Systems

M. Innocenti[1] and M. Napolitano[2]

[1]Department of Electrical Systems and Automation
University of Pisa, Via Diotisalvi 2, 56126 Pisa, Italy

[2]Department of Mechanical and Aerospace Engineering
G-60 Engineering Science Building, West Virginia University
Morgantown, WV 26506, USA

Summary

Fault identification, isolation, and accomodation have become critical issues in the overall performance of advanced aircraft systems. Neural Networks have shown to be a very attractive alternative to classic adaptation methods for identification and control of non-linear dynamic systems. The purpose of this paper is to show the improvements in neural network applications achievable through the use of learning algorithms more efficient than the classic Back-Propagation, and through the implementation of the neural schemes in parallel hardware. The results of the analysis of a scheme for Sensor Failure, Detection, Identification and Accommodation (SFDIA) using experimental flight data of a research aircraft model are presented. Conventional approaches to the problem are based on observers and Kalman Filters while more recent methods are based on neural approximators. The work described in this paper is based on the use of neural networks (NNs) as on-line learning non-linear approximators. The performances of two different neural architectures were compared. The first architecture is based on a Multi Layer Perceptron (MLP) NN trained with the Extended Back Propagation algorithm (EBPA). The second architecture is based on a Radial Basis Function (RBF) NN trained with the Extended-MRAN (EMRAN) algorithms. In addition, alternative methods for communications links fault detection and accomodation are presented, relative to multiple unmanned aircraft applications.

Introduction

The previous decade has witnessed a dramatic increase in neural networks (NN) research, mostly induced by the introduction of the Back-Propagation algorithm for feedforward NNs with supervised learning. During the initial stage of this growth NNs have been applied to pattern recognition and classification, which can be classified as static problems. Within a later trend NNs were proposed for identification and control of dynamic systems of both linear and nonlinear nature. The reason behind this application lies in the fact that, despite progresses were made in adaptive control theory for linear systems, a solution for the identification and control of a non-linear uncertain system was yet to be formulated. A complete review on NN theoretical principles is widely available in the literature.

In general, a neural network is defined as an architecture featuring input and output parameters interconnected through one or more layers of neurons, also called processing elements, which provide a non-linear activation. The operations at each neuron consist of adding the total weighted sum of all the input signals of the preceding layer, with the option of adding a threshold value; this total sum becomes then the argument of a non-linear function, which provides the activation, to generate an output signal. During learning, the weights and the thresholds of the architecture are updated, typically with a gradient based method, with the goal of minimizing a cost function associated with an identification, control or other task.

Fault tolerant flight control systems (FTFCS) are required to accommodate failures in different components, mainly sensors and actuators. Some of the procedures described in this paper focus on the sensor failure problem. Sensor failures are critical when the measurements from the failed sensor are used in the feedback loop of the flight control laws. In fact, sensor failures, if left undetected and uncompensated, can lead to

Paper presented at the RTO AVT Course on "Intelligent Systems for Aeronautics", held in Rhode-Saint-Genèse, Belgium, 13-17 May 2002, and published in RTO-EN-022.

closed-loop instability and, potentially, unrecoverable flight conditions. Most of today's high performance military aircraft as well as commercial jetliners feature a flight control system with triple physical redundancy in the sensory capabilities. A Fault Detection (FD) scheme monitors these components and, as a sensor failure is detected, it recovers the nominal functionality by switching from the faulty unit to one of the two remaining back-up units. A voting-scheme between the different signals from the sensors measuring the same parameter is typically used for this task. However, there are special purpose aircraft (e.g. UAVs) and spacecraft where reduced complexity, lower costs, and weight optimization are major design specifications.

For these classes of aircraft and spacecraft, an alternative approach can take advantage of the analytical redundancy existing in the system to provide fault tolerance capabilities. Analytical redundancy is the known functional relationship existing between the system's outputs, states and inputs. In other words, the information provided by a set of sensors along with a priori knowledge of the system allows detecting and identifying the faulty sensor, while estimating the related variable as a function of other measured variables. Research on fault tolerance based on analytical redundancy has produced a quite mature framework especially for linear systems. Currently, substantial research challenges are in the extension of the previous schemes to the case of nonlinear systems. Adaptive and on-line approximation methodologies are receiving considerable attention and the use of different types of Neural Networks has been proposed to take advantage of some of their properties. The performance of a NN in approximating a given function critically depends on its underlying structure as well as on the featured training algorithm. The two classes of NNs here considered for sensor failure evaluation are the Multi-Layer Perceptron NNs (MLP NNs) and the Radial Basis Function NNs (RBF NNs). Due to their approximation and adaptation capabilities, both MLP and RBF NNs have been proposed as estimators to approximate complex non-linear systems. Within a larger picture both classes of NNs can be considered as "nonlinear approximators" and a systematic procedure based on Lyapunov's theory for creating non-linear approximators with stability characteristics could be applied to each class.

Other aspects of interest in fault detection are those related to succesfull missions of unmanned air vehicles, possibly flying in formation. Here, one of the main concerns is the reliability of the overall communications system, since each vehicle relies heavily on it, in order to maintain the desired position within a formation, and follow some prescribed flight path. In this case, there are many forms of analytical redundancy that can take place, based on neural network techniques, or other methodologies using traditional optimizers. One interesting aspect is the capability of using NNs to estimate the wake effects in the absence of position sensor information, or as a redundancy for a sensor information failure. Another application is the formation management flow change due to generic failures in the transmitters and/or receivers. In this case, in addition to fault accomodation, the problem is also that of reconfiguring the overall formation structure, in order to continue the mission and/or the assigned task(s). After a generic fault, a fast reconfiguration procedure is run to restore formation-keeping as quickly as possible. When the formation communications are in a safe configuration again, it may be necessary to switch aircraft positions or move an aircraft to an empty node to maximize the formation keeping and safety of all aircraft inside the formation. Since the node-changing decision is decentralized, the algorithm that makes the decision must be deterministic to avoid simultaneous conflicting decisions by more than one aircraft in response to the same post-fault-reconfiguration requirements. The second problem to solve is distribution of fault event information so that all aircraft can apply appropriate reconfiguration decisions. Even if the available communication channels, those used to exchange flight data for formation control, are sufficient for optimal channel reconfiguration, it is possible that the best aircraft candidate for making the node change decision may not be informed of the fault event.

Basic Concepts in Identification and Control

A generic multi-input/multi-output (MIMO) discrete-time system with n states, m inputs and l measurable outputs can be represented by:

$$x(k+1) = \Phi\left[x(k), u(k)\right]$$
$$y(k) = \Psi\left[x(k)\right]$$

where: $x(k) = [x_1(k), x_2(k), \dots, x_n(k)]^{T,}$ $u(k) = [u_1(k), u_2(k), \dots, u_m(k)]^T$, $y(k) = [y_1(k), y_2(k), \dots, y_l(k)]^T$.

If the system is linear and time-invariant then the equations take on the well-known form:

$$x(k+1) = Ax(k) + Bu(k)$$
$$y(k) = C[x(k)]$$

where A_{nxn}, B_{nxm}, and C_{nxl} are the traditional state, control and output matrices. The relative system can be then defined through the set of matrices (*A,B,C*). If the system is linear and known, controllability of the system can be asserted and different feedback control strategies, either of classic compensation type or from modern control theory, can be introduced to meet different specifications. For the identification task at linear conditions, that is if the system is linear and (*A,B,C*) are partially or completely unknown, the problem of identifying (*A,B,C*) can be clearly formulated and solved.

However, if the system is non-linear and Φ, Ψ are unknown, the identification problem is quite more complicated. Furthermore, for the control task at non-linear conditions, there is a lack of control schemes regardless whether Φ, Ψ are known or unknown. Adaptive control of uncertain systems is feasible in the linear case. In particular, within this direction, SISO adaptive systems tracking a desired dynamics have been extensively analyzed with the Model Reference Adaptive Control (MRAC) method being one of the most representative approaches.

Let us for example consider the problem of identification and control of SISO nonlinear unknown systems. There are four discrete-time models, which can be used to describe this type of systems:

$$Model\,I\colon y(k+1) = f[y(k),..,y(k-n)] + \sum_{i=0}^{m} b_i u(k-i)$$
$$Model\,II\colon y(k+1) = f[y(k),...,y(k-n)] + \sum_{i=0}^{m} b_i u(k-i)$$
$$Model\,III : y(k+1) = f\,[\,y(k),...,y(k-n)] + g\,[u(k),...,u(k-m)]$$
$$Model\,IV\colon y(k+1) = f[y(k),...,y(k-n),u(k),...,u(k-m)]$$

There has been a large variety of studies related to observability, controllability for these systems as well as for designing controllers and observers for them. In this example we will refer to multi-layer feed-forward NNs with supervised learning performed with gradient based algorithms. The suitability of a NN approach to the general identification and control task becomes somewhat immediate if one considers the following properties.
Applicability to non-linear systems. The applicability of NNs to non-linear systems originates from their mapping capabilities. It has been shown that a feedforward NN with at least one hidden layer is capable of approximating any nonlinear function once enough training is provided. This property can be extremely useful when the input/output data are related to a non-linear time-varying system.
Parallel processing and hardware implementation. NNs have an inherent parallel architecture, which, leads naturally to parallel hardware implementations. These implementation have also the advantages of having, in general, high degree of fault-tolerance and high processing speed, due to the simplicity of the computations (additions, subtractions, multiplications). These advantages have been magnified by the introduction of digital microprocessors with ever increasing performance.
Multivariable systems. NNs, by definition, are multi-input, multi-output (MIMO) entities and this, naturally, leads to their application to multivariable systems.
Learning and adaptation. NNs can be trained using past recorded data or simulated data (off-line training) or current data (on-line learning). Furthermore, among the learning capabilities of a NN, one must differentiate between local and global learning capabilities. Local learning leads to adaptation capabilities to new dynamic conditions while global learning leads to memorization capabilities from previously presented data.

Consider a generic SISO non-linear unknown system described by any of the four models in the previous section. The objective is to provide a NN-based identification scheme for the functions *'f'* and/or *'g'*. A necessary assumption for a well-posed approach is that the system have a bounded response for any bounded set of input data (that is, it is a BIBO stable system). This, in turn, implies that the neural identifier will also need to be BIBO stable. Extensive research efforts are undergoing at this time for the analysis and the proof of stability properties of multi-layer neural networks. Another assumption is that the neural identifier will have to be of series-parallel type meaning that the actual system output *y(k)*, rather than its estimate, is used as input by the identification scheme during and after the training. The alternative is to use a parallel-type identifier, meaning that the estimate of *y(k)* from the neural identifier is obtained using as input estimates of *y(k)* at previous time steps. It should be underlined that the stability for this type of identifiers has yet to be proved even for simple linear systems. The selection of a series-parallel mode instead of a parallel model is mainly due to the fact that, since the system is assumed to be BIBO and stable, all the input to the NNs are bounded. However, once the neural identifier has been sufficiently trained and the estimation error has converged to a sufficiently low asymptotic value, the series-parallel mode may be replaced by the parallel-working mode. With these working assumptions in place, the next issue to be addressed is related to the selection of the training algorithm for the neural identifiers. To date, the majority of the training for neural networks is being performed with the Back-Propagation (BP) algorithm, a gradient-based optimization method. Slow learning, especially for large order systems, and local minima are "classical" well documented problems of the BP. Unfortunately, these two problems are known to be mutually related and interactive. For example, it has been shown that the learning speed can be improved by setting a proper learning rate. However, this also increases the possibility for the BP to be trapped in a local minimum or to oscillate around the global minimum.

To solve address problems, several alternatives to the BP have been proposed by many researchers. Most of the work has been concentrated on introducing different activation functions or particular procedures for selecting the initial weights. Within this stream of investigations, the authors have used an approach based on the introduction of an heterogeneous network. In an heterogeneous network each neuron in the hidden and output layers has its own capability of updating some new parameters giving the overall architecture increased mapping and adaptation performance. Specifically, in a heterogeneous network each neuron is able to change its output range (upper and lower bounds: U, L) and the slope of the sigmoid activation function (temperature: T). The new non-linear activation function will be given by:

$$f(x_{ij}, U_{ij}, L_{ij}, T_{ij}) = \frac{U_{ij} - L_{ij}}{1 + e^{-x_{ij}/T_{ij}}} + L_{ij}$$

where *i,j* are the indices for the generic neurons of the hidden and output layers and *'x'* is the argument of the classic sigmoid function of the BPA. This function is implemented at each processing element of the hidden layer(s) and output layer. The gradient is calculated to perform the steepest descent as in the BP. The only difference is that the gradient descent will be found with respect to each of the independent variables *'x','U','L','T'*, as opposed to *'x'* only. The training algorithm associated with this heterogeneous network is named Extended Back-Propagation (EBP) and is described with details in the listed references.

The design of on-line learning neural controllers, and neural identifiers can be complicated by the large number of degrees of freedom (number of hidden layers, number of neurons per hidden layer, size of input data window, magnitude of the learning rate and momentum coefficients) in the selection of the neural architectures. Previous work provided the following guidelines for the selection of suitable neural architectures :

- on-line learning neural controllers and neural identifiers are generally very robust to different number of neurons per hidden layer and a different input data window;
- medium-high learning rates, either constant or slowly decreasing with time, performed better for on-line learning neural controllers while low learning rates perform better for on-line learning neural identifiers;
- simple single hidden layer architectures perform as well or better than more complicated and more computationally intensive multiple hidden layers architectures;
- a proper selection of the data to be presented as input and an appropriate choice of the performance indices to be minimized are, by far, the most effective degrees of freedom in the selection of neural architectures.

Next, neural identifiers will be introduced using both BP and EBP algorithms. Consider a different SISO unknown non-linear system described by :

$$y(k+1) = \frac{y(k)}{1 + [y(k)]^2} + [u(k)]^3$$

representing a Model III system. In this case a neural network based identification scheme will consist of two neural identifiers, that is an identifier N_f for the function:

$$f\left[y(k)\right] = \frac{y(k)}{1+\left[y(k)\right]^2}$$

and an identifier N_g for the function:

$$g\left[u(k)\right] = \left[u(k)\right]^3$$

such that the estimation dynamics can be expressed as:

$$\hat{y}(k+1) = N_f\left[y(k)\right] + N_g\left[u(k)\right]$$

As two hidden layer architecture was selected: both neural identifiers are of the type $N_{1,20,10,1}$. The study involved two different phases, that is an initial training phase followed by a testing phase. During the training phase two sets of two similar $N_{1,20,10,1}$ architectures (one using the BP and the other using the EBP) were trained for 100,000 steps with *u(k)* being a random number uniformly distributed between [-2,2] and using a 0.25 learning rate. The range for *u(k)* implies that the estimate of *g* will span between [-8,8] while the estimate of *f* will span between [-10,10]. During this phase the identifier is working in a series-parallel mode, that is the actual system outputs at previous time steps are used by the identifier to provide an estimate at the present time step.

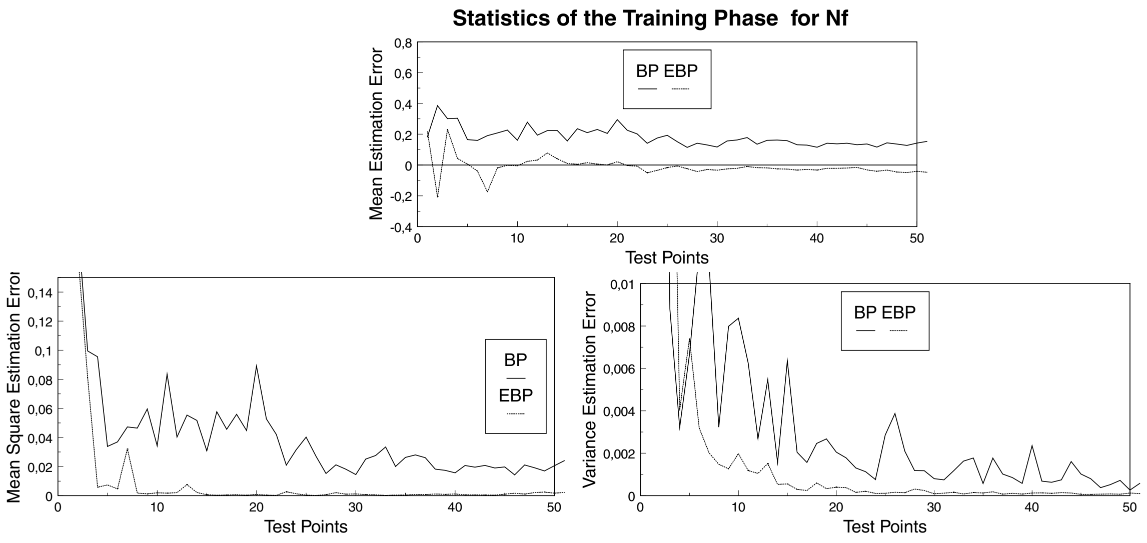


Figure 1. Statistics of Training Phase for Function *'f'*

The training performances associated with the two algorithms were evaluated every 2,000 steps during the 100,000 steps simulation for a total of 50 test points. At each test point the performance of the neural identifier were analyzed using a 100 steps simulation. Classical statistical parameters, that is mean, mean square and variance, were introduced to monitor the estimation errors for *'f'* and *'g'* and given by

$$M_{EE} = \sum_{i=1}^{100} \frac{\left[\hat{y}(i) - y(i)\right]}{100} = \sum_{i=1}^{100} \frac{e(i)}{100}, \;, \; MS_{EE} = \sum_{i=1}^{100} \frac{\left[\hat{y}(i) - y(i)\right]^2}{100} = \sum_{i=1}^{100} \frac{e(i)^2}{100} \; V_{EE} = \sum_{i=1}^{100} \frac{\left[e(i) - M_{EE}\right]^2}{100}$$

In terms of complexity and required computational effort for the neural architectures, the number of parameters to be updated at each time is 354 using the EBP and 261 using the BP. Therefore, there is a 35% increase in complexity using the EBP due to the additional parameters *U,L,* and *T*. The statistical results are shown in Figure 1 for the identification of the function *'f'*, similar results are available for function *'g'*. The trends for M_{EE}, MS_{EE} and V_{EE} show that the EBP clearly outperforms the BP. In particular, it can be seen that the estimates using BP exhibit a bias most likely caused by the BP algorithm getting stuck in a local minimum point. Efforts were made in repeating the training phase with the BP several times; however, local minimum points problems seemed always to be present to a certain extent. The bias problem for the BP is even more clear in Figure 2, showing estimates of *'f'* following the training phase. After the 100,000 steps training phase the two combined neural identifiers, working in a parallel mode, providing an estimate for *y(k+1)* are tested for 1000 steps with the control sequence *u(k)=sin(2πk/25)+sin(2πk/10).* The results are reported in Figure 3 and Table I indicating better performance of the EBP algorithm. The reason for the improved performance of the EBP vs. the BP algorithm has to be searched in the different learning characteristics.

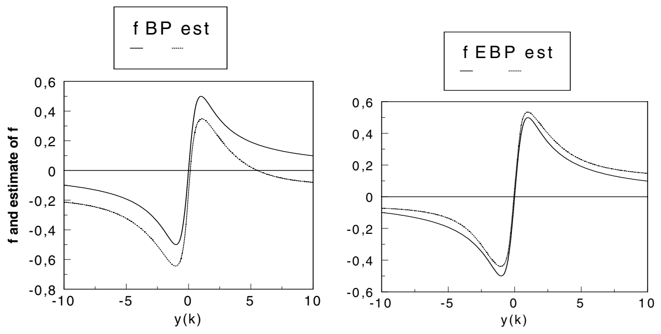


Figure 2. Testing Phase, Estimate of Function *'f'*

Table I. Statistics of Estimation Error: Testing Phase

	BP Algorithm	EBP Algorithm
Time	100,001-100,100	100,001-100,100
Mee	-.0277	-.0154
Msee	.0017	.000518
Vee	.000945	.000750

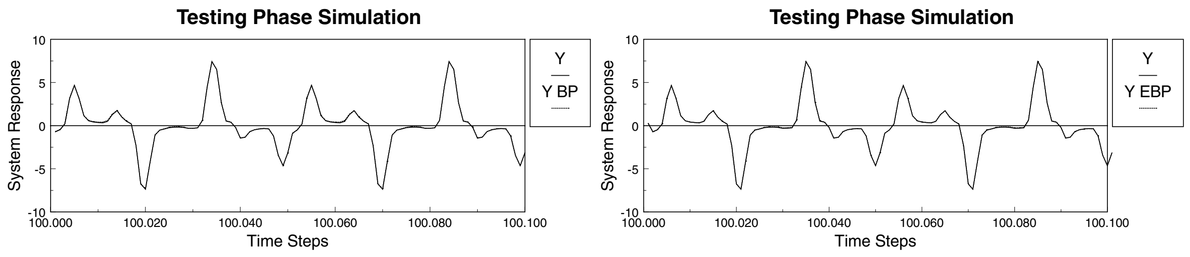


Figure 3. Testing Phase: Simulation

The reason for the improved performance of the EBP vs. the BP algorithm has to be searched in the different learning characteristics. The EBP has more degrees of freedom in the learning through the use in the gradient method of the following partial derivatives :

$$\frac{\partial f\left[x_{ij},U_{ij},L_{ij},T_{ij}\right]}{\partial x_{ij}} = -\frac{\left(O_{ij}-U_{ij}\right)\left(O_{ij}-L_{ij}\right)}{T_{ij}\left(U_{ij}-L_{ij}\right)}, \quad \frac{\partial f\left[x_{ij},U_{ij},L_{ij},T_{ij}\right]}{\partial U_{ij}} = \frac{1}{1+e^{-x_{ij}/T_{ij}}}$$

$$\frac{\partial f\left[x_{ij},U_{ij},L_{ij},T_{ij}\right]}{\partial L_{ij}} = 1-\frac{1}{1+e^{-x_{ij}/T_{ij}}}, \; .. \; \frac{\partial f\left[x_{ij},U_{ij},L_{ij},T_{ij}\right]}{\partial T_{ij}} = \frac{x_{ij}\left(O_{ij}-U_{ij}\right)\left(O_{ij}-L_{ij}\right)}{T_{ij}^{2}\left(U_{ij}-L_{ij}\right)}$$

as opposed to the derivative

$$\frac{\partial f\left[x_{ij}\right]}{\partial x_{ij}} = \frac{e^{-x_{ij}}}{\left(1+e^{-x_{ij}}\right)^2}$$

in the BP algorithm. In the above equations, $O_{i,j}$ represents the generic output of a neuron at the hidden layer(s), and at the output layer respectively.

The previous example illustrated successful applications of neural identifiers. Next, the problem of adaptive control via NNs is considered. The problem of controlling a generic dynamic system can be divided into a regulation or a tracking. In a regulation problem the goal is to achieve, through an appropriate feedback control law, an equilibrium condition around a certain set point. In the tracking problem the objective is instead to achieve, through the feedback control law, an accurate following of a reference model response. The control part of this example is related to the tracking problem for an unknown non-linear SISO system. It is known that most of adaptive control theory is based on the Model Reference Adaptive Control (MRAC) relative to linear time-invariant systems. Particularly, Reference 14 shows two different approaches, that is the direct and the indirect approaches. It should be emphasized that the closed-loop dynamics with the adaptive control scheme, both in the direct and indirect form, will be nonlinear even if the open-loop system is linear and time-invariant. In a direct adaptive control scheme the parameters of the control scheme are adjusted on-line to satisfy some control specifications. In indirect adaptive control scheme instead the parameter identification scheme is first implemented providing estimates of the actual system parameters with the parameters of the controller calculated assuming these estimates of the system parameters.

There is a well known problem in using a neural network approach for a direct adaptive control of a non-linear system. In fact, the neural network controller output error, which is used to update the weights and the thresholds of the neural controller, is not directly available. The only available information is the tracking error between the desired and the actual response, which is only a function of the error at the neural controller output. Several methods have been proposed in the literature to overcome this problem and tested on non-linear SISO systems. Reference 21 suggests a method where the system dynamics are treated as not modifiable layer of the neural controller and the tracking error is back-propagated through the dynamics. In [5] another approach is suggested, where the tracking error is transformed into the neural controller error using the dynamics inverse transfer matrix. Reference [12] presents a third method where a feedback gain is introduced to generate a transformed error signal used to update the architecture of the neural controller. Finally in [26] a scheme is introduced, where the architecture of the neural controller is updated using the tracking error with the addition of an extra single layer gain between the neural controller and the system.

This section, however, describes results related to indirect adaptive control with a neural approach. In the absence of well formulated stability and robustness criteria a natural approach for nonlinear systems can be given by trying to extend some results relative to linear adaptive control theory. For example, in linear adaptive control theory it was concluded that adaptive control laws for a stable closed-loop system can be found if some "a priori" knowledge of the system was available. Similarly, the probability of success of the adaptive control of a nonlinear SISO system can be assumed to be a strong function of the amount of "a priori" information on the unknown system.

A safer and more conservative set-up of the problem is to apply adaptive control, in order to achieve tracking of a desired system response using an indirect approach. That is only after the non-linear system has been identified within a desirable level of accuracy. It should be underlined, however, that even this conservative approach does not guarantee closed-loop stability and acceptable tracking performance, even in the case of a BIBO stable system, since the adaptive control law can lead to an unstable system during the transient. It can be speculated that fast on-line learning capabilities of the neural controllers, which are direct function of the selected learning algorithm, are critical for this purpose. On the other side, closed-loop stability, and possibly good tracking performance are only a remote possibility if adaptive control using neural identifiers and neural controllers is attempted with a direct approach, that is without any "a priori" identification phase. In the following, the tracking capabilities and the robustness of a NN-based indirect adaptive control scheme are evaluated.

Consider the same unknown SISO nonlinear system described before, and representing a Model III system. This time the objective is to find an adaptive control law *u(k)* such that the actual system response *y(k)* tracks a desired model response *y_m(k),* with the dynamics for the reference model given by:

$$y_m(k+1) = 0.6y_m(k) + r(k)$$

recall that previously we had:

$$\hat{y}(k+1) = N_f\left[y(k)\right] + N_g\left[u(k)\right] = \hat{f}\left[y(k)\right] + \hat{g}\left[u(k)\right]$$

Following the training phase we had that the estimates of *y(k)* were converging to the actual *y(k)* response. For the actual system to track the desired model response we would have

$$\hat{f}\left[y(k)\right] + \hat{g}\left[u(k)\right] = 0.6y_m(k) + r(k)$$

isolating $\hat{g}\left[u(k)\right]$ yields:

$$u(k) = \hat{g}^{-1}\left\{0.6y_m(k) + r(k) - \hat{f}\left[y(k)\right]\right\} = \hat{g}^{-1}\left[z(k)\right]$$

The tracking problem is solved by introducing a neural adaptive controller N_c for the online determination of *u(k)*. In particularly, two neural controllers, trained using EBP and BP respectively, are introduced. Both of them will have architecture $N_{1,20,10,1}$ with input *z(k)* with a learning rate of 0.01. The closed-loop system will therefore include two neural identifiers (N_f,N_g); and one neural controller (N_c).

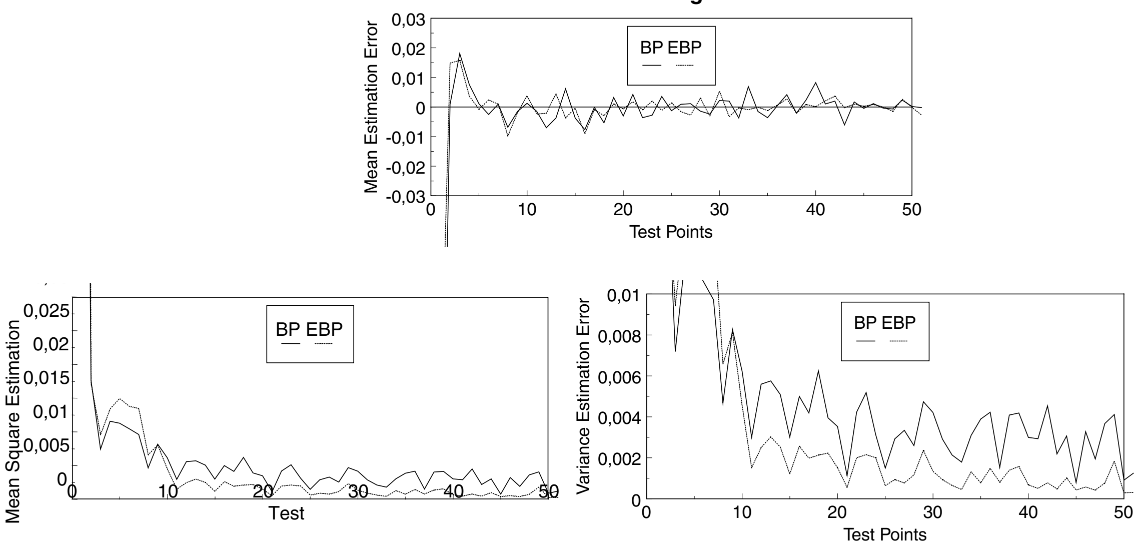


Figure 4. Statistics of the Training Phase for "g-inverse"

Again, there is an initial training phase and a testing phase. During the training phase two similar sets (one using BP and the other using EBP) of 3 neural architectures (N_f,N_g,N_c) were trained for 100,000 steps using the same random input *u(k)* between [-2,2] used previously. The training performance were evaluated every 2,000 steps for a total of 50 test points. The statistical results for the identification of *'f'* and *'g'* were very similar to the ones obtained previously, while the results for the identification of the inverse function of *'g'* are shown in Figure 4. The tracking capabilities of the closed-loop system are then evaluated with a 5000 steps simulation. The results are summarized in Figure 5; in particular Figure 5 shows 2 different 100 steps simulation starting respectively at instants 1, and 4,901. The superior performance of the EBP algorithm are once again confirmed by the statistical data on the tracking error shown in Table II. It can be seen that the EBP neural estimate is practically unbiased and has a variance approximately 4.25 times smaller than the correspondent BP variance.

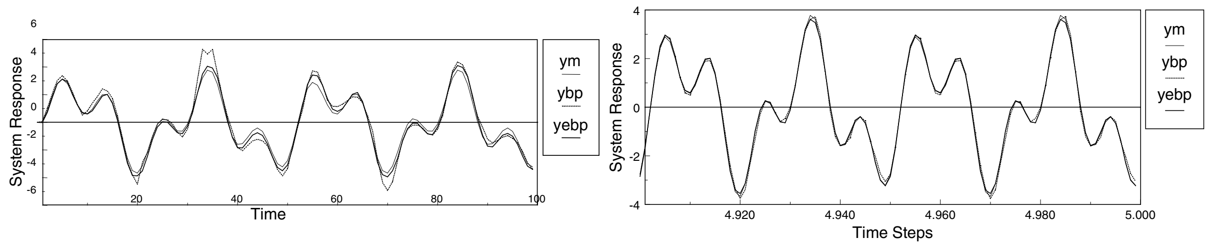


Figure 5. Testing Phase: Simulation

Table II. Statistics of the Tracking Error: Testing Phase

	BP Algorithm		
Time	1-1,000	2,501-2,600	4,901-5000
Mte	-0.0237	.0004	-.0003
MSte	.2743	.0139	.0148
Vte	.2765	.014	.0149
	EBP Algorithm		
Time	1-1,000	2,501-2,600	4,901-5000
Mte	.0707	.0005	.0002
MSte	.0720	.0042	.0035
Vte	.0677	.0042	.0035

The next step of this study is to evaluate the robustness of the neural-based tracking to a time-varying system. For this purpose the original nonlinear SISO system is simulated to change instantaneously, starting at the 5,000-th step, to 3 different arbitrarily selected configurations before returning to the original configuration. At each configuration a 5,000 steps simulation is performed using both the EBP and the BP algorithms. The different configurations are:

For k=5,000-9,999, Config. 1

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + [1.5u(k)]^3 + 2$$

For k=10,000-14,999, Config. 2

$$y(k+1) = \frac{2y(k)+2}{0.5+y^2(k)} + [1+u(k)]^3 - 1$$

For k=15,000-19,999, Config. 3

$$y(k+1) = \frac{2y(k)+2}{0.5+y^2(k)} + [1+u(k)]^3 - 1$$

For k=20,000-24,999, Config. 4

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + [u(k)]^3$$

Several 100 steps simulations time histories of the reference model along with the closed-loop system response using both EBP and BP algorithms are shown in the next figures for different points of the 5,000 step simulation. For the tracking problem, the statistics are expressed in terms of the tracking error, defined as:

$$M_{TE} = \sum_{i=1}^{100} \frac{[y(i) - y_m(i)]}{100} = \sum_{i=1}^{100} \frac{e(i)}{100}, \quad MS_{TE} = \sum_{i=1}^{100} \frac{[y(i) - y_m(i)]^2}{100} = \sum_{i=1}^{100} \frac{e(i)^2}{100}, \quad V_{TE} = \sum_{i=1}^{100} \frac{[e(i) - M_{TE}(i)]^2}{100}$$

From the plots and the statistical results, it can be seen that the tracking performance associated with the EBP algorithm are somewhat worse than the correspondent BP performance only during the transient phase of the transition of the system between different configurations. Once this transient phase is past, the closed-loop system with EBP trained neural identifiers and controller clearly outperform the BP version.

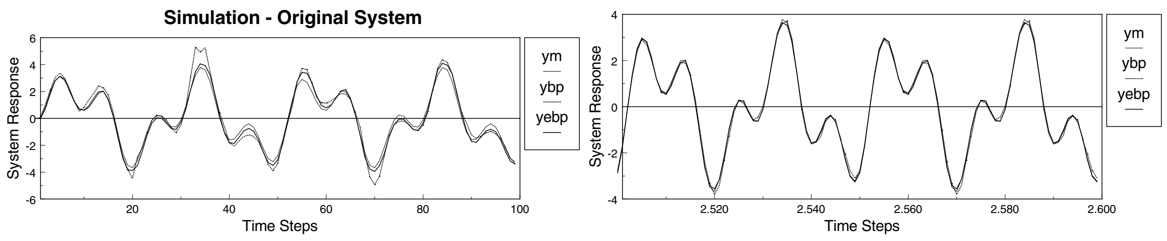


Figure 6. Simulation Comparison Original System

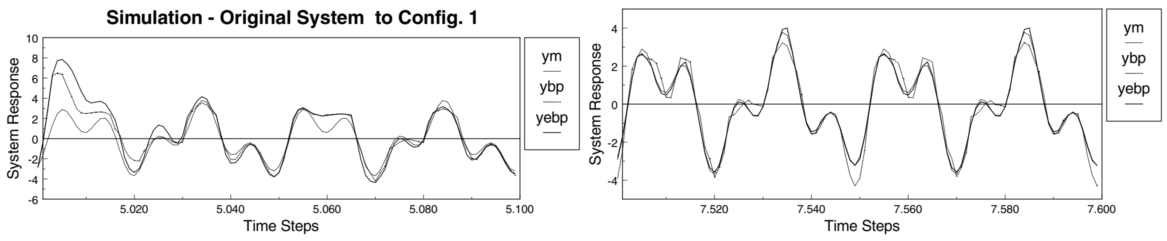


Figure 7. Simulation Comparison to Config. 1

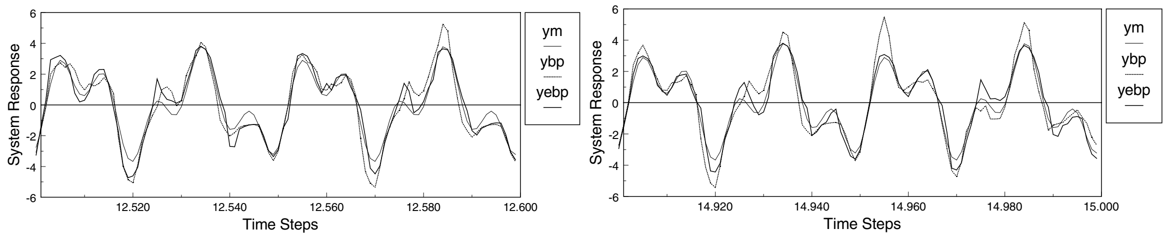


Figure 8. Simulation Comparison to Config. 2

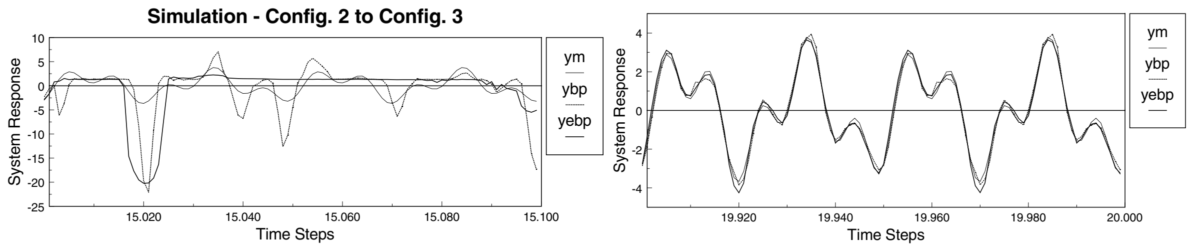


Figure 9. Simulation Comparison to Config. 3

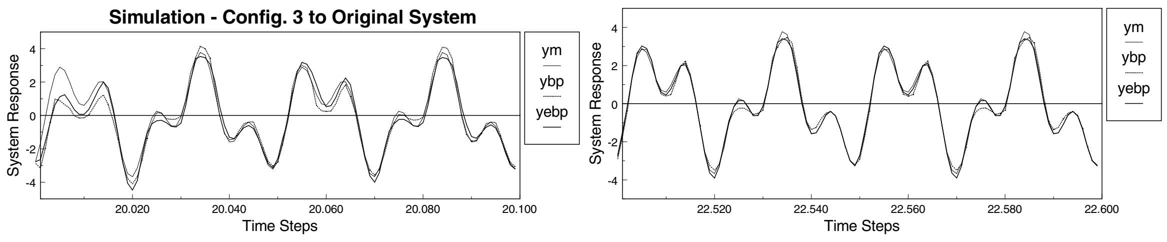


Figure 10. Simulation Comparison to Original System

Failure Detection using Neural Networks

Failure detection (FD) is a widely investigated problem in current flight control research, due to the ever-increasing complexity, and interconnectivity of the system at end. Generally speaking, FD implies some sort of continuous monitoring of the measurable outputs of the system. Under nominal conditions, these variable tend to follow rather well established patterns, within the uncertainty due to process disturbances, such as atmospheric turbulence, sensor noise, and actuator biases. A failure to any component of the flight control system (FCS) will produce deviations from nominal and somewhat predictable trajectories. Spotting a deviation between nominal and failed trajectory is usually the basis of any FD technique. The knowledge of the system's behavior by the flight control system at any operating point needs then to be available at a very heavy computational and storage costs, limiting the prediction to linear time invariant, low order, and white noise-based models of predicted trajectory behavior.

Neural network architectures have been primary candidates to maintaining the same level of, or improving FD capability without increasing computational burden. The mapping property of a NN architecture can be extremely attractive when the input-output data are related to completely or partially unknown dynamics. Several factors must be taken into account however, when considering the implementation of NN estimators. A key issue in the design of a NN based state estimator is the selection of online learning vs. offline learning. Of course in an offline situation all training has been performed beforehand, and the network has a frozen architecture. Due to the variability of the dynamics and failure conditions, however, the potential of on-line architectures was studied in the present work because of its inherent added flexibility. In the following, we refer to an on-line structure of feed-forward type with interlayer connections schematically shown in Figure 1 as implemented by a three-layer NN. More details on the architecture can be found in [32].

General Sensor Failure Detection

The overall sensor failure detection, identification, and accommodation (SFDIA) problem is addressed in the context of aircraft dynamics, by using multiple architectures consisting of a centralized structure (MNN), and a set of 'n' decentralized structures (DNN), where 'n' is the number of non-redundant sensors. Figure 11 shows a diagram of the layout of the main neural network. A similar one, but with a single output, can be obtained for the n DNNs which, after sensor failure, will provide accommodation. Figure 11 shows the flow of the SFDIA process in a three-sensor case, with a simulated failure for sensor #2. A more detailed description of the process can be found in the references (32, 35).

The main network MNN receives the input from control commands and the vector of measurements $\underline{y}$, providing the estimate $\underline{y}^E$ of the same measurements as output. The EBPA updates MNN based on the error, and the learning process is monitored by a quadratic error parameter

$$E_{MNN} = \sum_{i=1}^{outputs} \left(\underline{y}^E - \underline{y}\right)^2$$

Each i^{th} DNN receives the input form the control and *n-1* sensors, where the missing signal is relative to the corresponding measurement. The training procedure is similar to that of MNN, until failure is detected. Upon failure of the i^{th} sensor, detection can be accomplished by the value of E_{MNN} exceeding a given threshold.

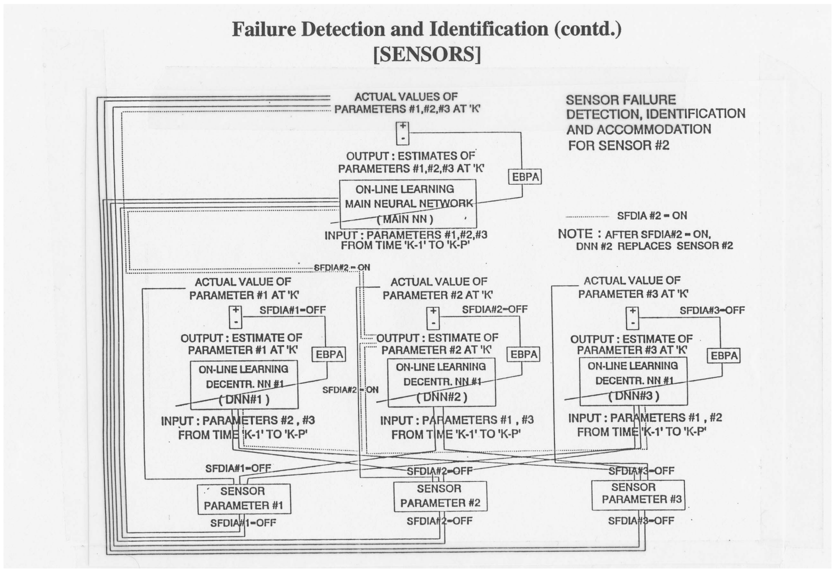

Failure Detection and Identification (contd.)
[SENSORS]
ACTUAL VALUES OF PARAMETERS #1,#2,#3 AT 'K'
OUTPUT : ESTIMATES OF PARAMETERS #1,#2,#3 AT 'K'
ON-LINE LEARNING MAIN NEURAL NETWORK (MAIN NN)
INPUT : PARAMETERS #1,#2,#3 FROM TIME 'K-1' TO 'K-P'
EBPA
SENSOR FAILURE DETECTION, IDENTIFICATION AND ACCOMMODATION FOR SENSOR #2
SFDIA #2 = ON
NOTE : AFTER SFDIA#2 = ON, DNN #2 REPLACES SENSOR #2
ACTUAL VALUE OF PARAMETER #1 AT 'K'
OUTPUT : ESTIMATE OF PARAMETER #1 AT 'K'
ON-LINE LEARNING DECENTR. NN #1 (DNN#1)
INPUT : PARAMETERS #2 , #3 FROM TIME 'K-1' TO 'K-P'
ACTUAL VALUE OF PARAMETER #2 AT 'K'
OUTPUT : ESTIMATE OF PARAMETER #2 AT 'K'
ON-LINE LEARNING DECENTR. NN #1 (DNN#2)
INPUT : PARAMETERS #1 , #3 FROM TIME 'K-1' TO 'K-P'
ACTUAL VALUE OF PARAMETER #3 AT 'K'
OUTPUT : ESTIMATE OF PARAMETER #3 AT 'K'
ON-LINE LEARNING DECENTR. NN #1 (DNN#3)
INPUT : PARAMETERS #1 , #2 FROM TIME 'K-1' TO 'K-P'
SFDIA#1=OFF
SFDIA#2=OFF
SFDIA#3=OFF
SFDIA#2 = ON
SENSOR PARAMETER #1
SENSOR PARAMETER #2
SENSOR PARAMETER #3

Figure 11. SFDIA Diagram, Example of failed #2 Sensor

When this occurs, identification can be achieved by monitoring the absolute value of the estimation error of the i^{th} DNN

$$E_{DNN}^{ID} = \left| E_j \right|$$

Identification is again assumed if the error in the above equation is greater than a specified level. The procedure outlined above consists of two steps, thus reducing the chance of false alarm. Failure accommodation is then performed by replacing the value of the data coming from the failed sensor, with the estimate of the corresponding DNN, which stops the learning process and "freezes" its architecture at this point.

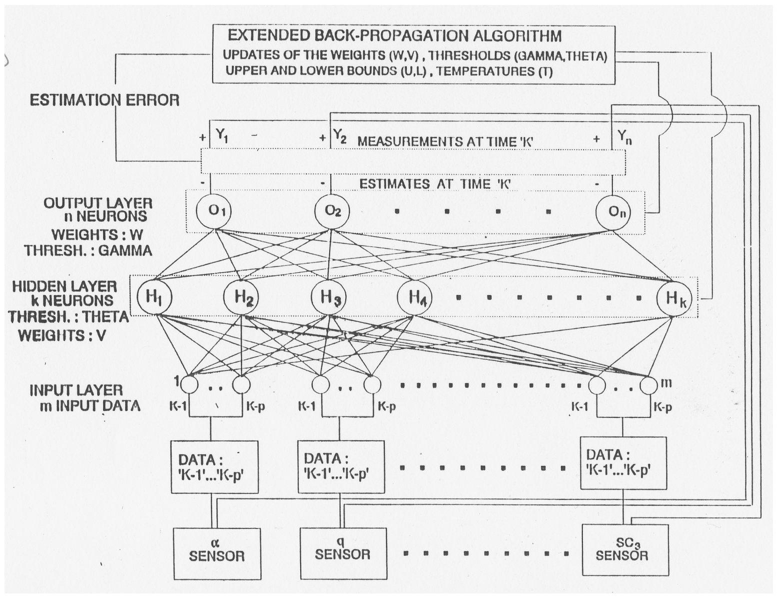


Figure 12. Sensor Failure Accomodation

Figure 12 with a single output describes therefore the accommodation procedure as well, whereas the entire SFDIA is described in Figure 11, noting that DNNs update is performed at each step, at the frequency of the sensors. References 34, and 35 present several applications simulating soft and hard failures in the dynamic behavior of a full-envelope nonlinear aircraft model representative of a high performance twin engine flight vehicle. The system considered here has 11 sensors and 5 actuators. As an example, we consider the case of a soft failure (the sensor does not fail completely) representative for instance of added bias in the instrument readings. The sensor considered here is the roll-rate gyro, which is one of the primary measurements in the lateral-directional motion of the aircraft. Figure 13 shows the time history of the error in the MNN with a detection spike at 1700 seconds, while the measured variable, roll rate is plotted in Figure 14. From that figure, the time history of the roll rate DNN follows the actual roll rate in a satisfactory manner.

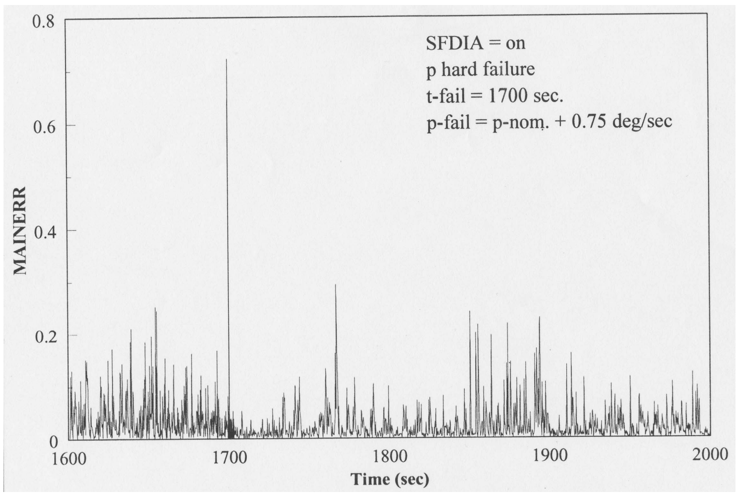


Figure 13. Roll Rate Gyro Bias Failure, NN Est. Error, SFDIA on

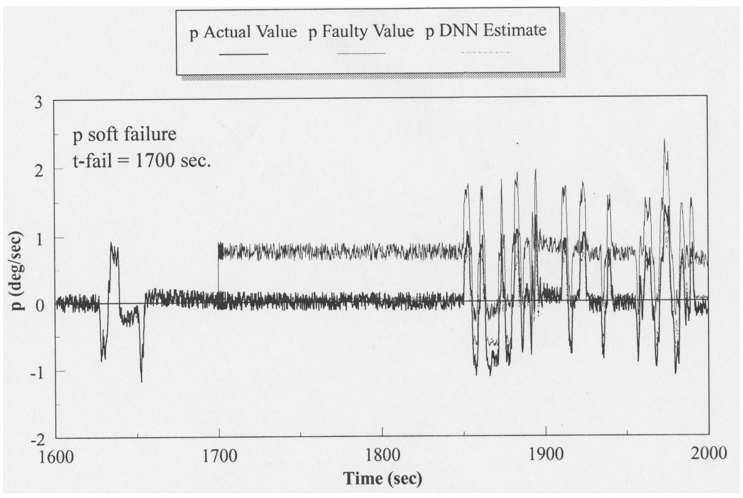


Figure 14. Roll Rate Gyro Failure, Time Histories

General Actuator Failure Detection

During the operation of a flight vehicle, actuator failures can occur for reasons internal to the system, as well as for external causes such as battle damage for military aircraft. Dealing properly with actuator failure is usually critical in that reduced controllability and even loss of controllability may happen with heavy financial losses as well life-threatening situations. The same dynamic system described in the previous section is considered here, which is modeled by a full nonlinear simulation code, and a FCS consisting of five aerodynamic control surfaces providing control moments on the three axes (pitch, roll, and yaw). The simulation is related to a typical high maneuverability task with simultaneous commands on the actuators, plus the appropriate thrust setting. Several actuator failures have been studied involving a fault in the hinge moment capability, and percent loss of control effectiveness. These failures require extensive analysis of the modified aerodynamic effects on the aircraft, in order to have a realistic simulation capability and were part of a concerted research effort, as reported in Reference 35.

The measurement data are all fed into the neural network, which is training on-line, and the process is monitored by evaluating the difference between the estimated and actual angular velocity in terms of body axes components at each instant *k* as

$$\Omega_{err}(k) = \frac{1}{2}\left[\left(p(k)-p^{est}(k)\right)^2+\left(q(k)-q^{est}(k)\right)^2+\left(r(k)-r^{est}(k)\right)^2\right]$$

It must be noted that in this case, unlike SFDIA, the network must be able to separate sudden variations in the actuator time histories due to pilot maneuvering commands, from failures, therefore training plays a much more critical role here. A parametric analysis was carried out to determine the most appropriate failure detection and identification scheme keeping in mind a compromise between NN architecture complexity and on-line computational requirements. The results led to a structure consisting of 9 inputs 2 patterns, 18 input data, 1 hidden layer, 10 P.E. in the hidden layer, and 3 P.E. in the output layer.

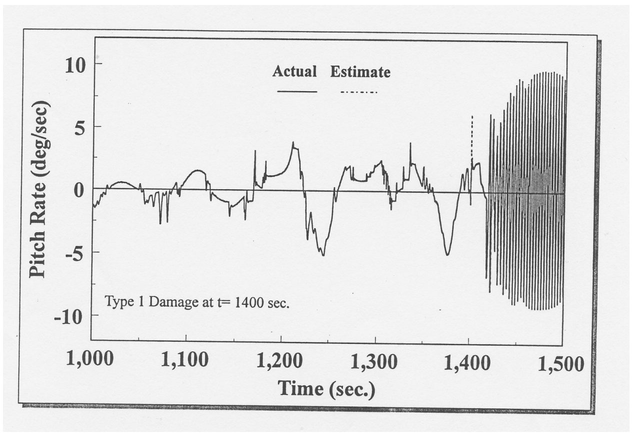


Figure 15. True and Estimated Pitch Rate after Failure

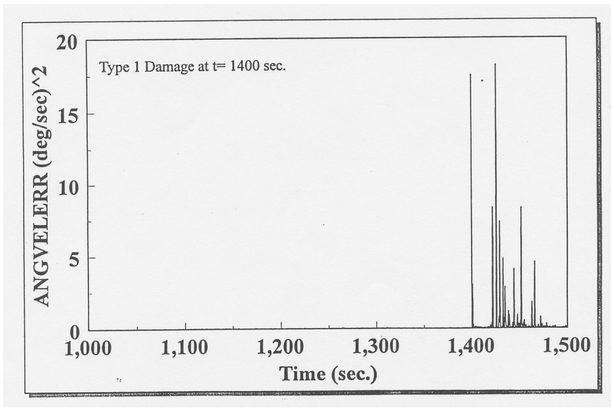


Figure 16. Actuator Failure Detection using Angular Velocity Error

Figure 15 gives an example of pitch rate (q) time history following a damage of the left stabilator of the aircraft at t=1400 sec (control surface stuck at 20 degrees, and 50% loss of control effectiveness). The change in behavior of q, which is coupled with similar changes in the other components of angular velocity, induces a variation in $\Omega_{err}(k)$, interpreted as failure detection as shown in Figure 16. Note that a different parameter could have been chosen in the failure detection scheme. Choice of $\Omega_{err}(k)$ was preferred since it contains variables directly affected by the aircraft dynamics, and dimensionally consistent. Following the detection of failure, sensor data must be interpreted in order to identify the damage location. The identification procedure relies on on-line evaluation of the cross-correlation function between key variables, which is then stored in temporary memory on the FCS computer. In the above example of a damaged stabilator, since both pitch and roll rate are involved, the cross-correlation function used is given by

$$\sum \left| R_{pq}^{coef}(k-1) \right| - \sum \left| R_{pq}^{coef}(k-2) \right| > \varepsilon_{FI}^{Thres}$$

and similarly for the other actuators. The behavior of the cross correlation time history is shown in Figure 17 taken with a time window of 7 instances. Although the presence of false alarms is not avoided, especially at the detection level, the use of above equation has proven very effective in terms of false alarm rate of the entire failure detection and identification process.

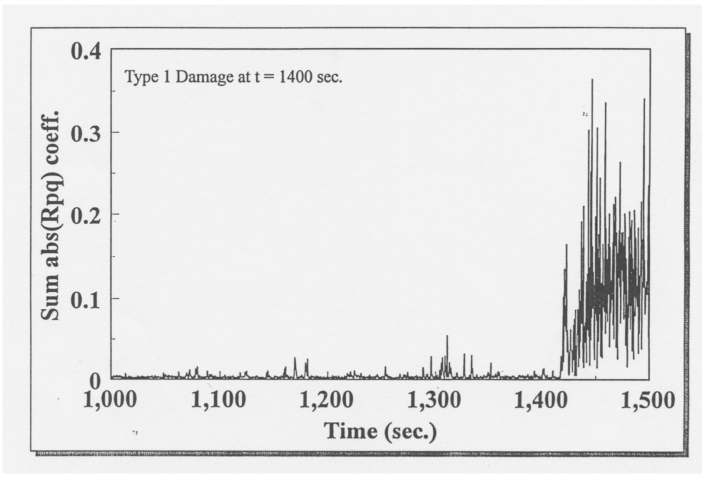


Figure 17. Pitch Rate and Roll Rate Cross Correlation as Failure Identification Parameter

Comparison with Standard Methods

One of the techniques traditionally used is the failure detection problem is based on the application of a Kalman filter both in its linear (KF) and nonlinear (EKF) versions. Valuable contribution of Kalman filtering relies heavily on the particular structure of the dynamic system, that is its linearity or linearization properties. In the following, a simple comparison between the neural network approach presented and the application of KF is presented. The dynamic system considered is representative of a high altitude unmanned aircraft used for scientific missions, sensor failure detection and identification is analyzed, based on a linearized model of the system itself, and relative to angular velocity rate gyros. The structure of the KF-based scheme is taken at first as specular of that relative to the neural network SFDIA previously described This implies a main Kalman filter block (MKF), plus a number n of decentralized blocks (DKFs) corresponding to the number of on board sensors (three sensors in this application). Several types of failures were considered ranging from instantaneous biases of different amplitude, to biases entered with ramp transients. Different system's configurations were also studied, from nominal model and system dynamics and noise, to discrepancies in the dynamics and noise statistics, and detailed results can be found in references 37, and 38. The example in figure 19 shows the failure of the Kalman filter scheme to recognize a 2.4 deg/sec bias entering as a ramp as supposed to the NN scheme, when system and filter models have discrepancies in the dynamics and noise statistics.

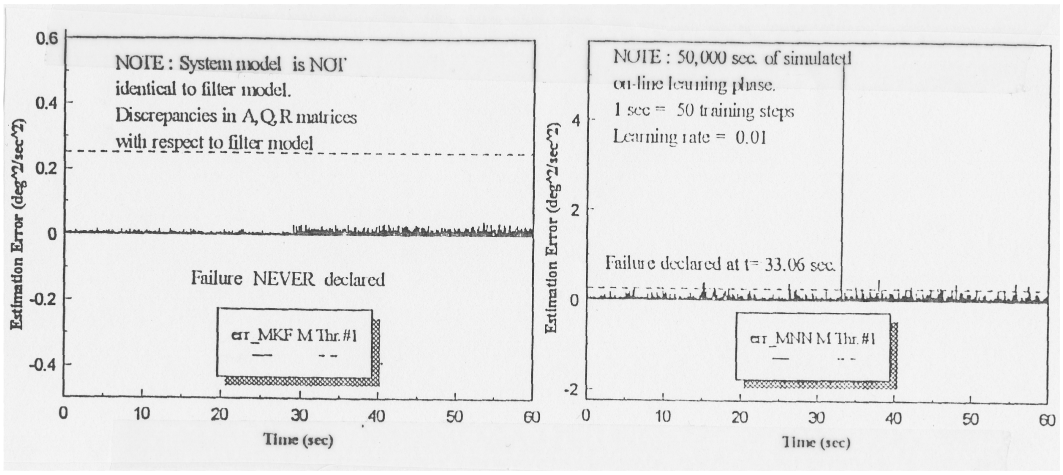


Figure 18. Estimation Error Comparison between KF and NN, with Model Discrepancy

Sensor Validation using Unmanned Air Vehicle Data

This section describes some results of SFDIA research besed on experimental data relative to an unmmanned research aircraft.

Description Of The B777 Model And Its Electronic Payload

The aircraft model shown in Figure 19 is a 1/24th semi-scale B777 research designed, built, assembled, instrumented, and flown at WVU. Scaled research models do not exhibit acceptable flying qualities due to the negative effect of the payload on the dynamic characteristic. Therefore, specific changes were made to the geometric and aerodynamic characteristics of the "perfectly scaled" model to achieve desirable handling qualities. In particular, the design aimed to achieve acceptable values for two specific parameters, that is the Thrust/Weight (T/W) and the Weight/Wing Surface (W/S) ratios. The aircraft was essentially designed around its electronic payload with a simple "2 component: tail/fuselage + wing" architecture. The wing, featuring ailerons, inboard/outboard flaps and housing the fuel tanks, was manufactured with fiberglass, carbon fiber, foam, and lightweight plywood.

Figure 19 - B777 Model in Flight and during Approach

The fuselage was manufactured using carbon fiber as a whole piece along with the horizontal tail and the vertical tail. Additional servos on the fuselage include a nose wheel and a brake servo. The B777 aircraft propulsion system features two ducted fan engines with a nine-rotor blade system with an engine shroud/rotor hub assembly. The units are housed in scaled engine nacelles to add realism to the model. The main characteristics are summarized in the following Table.

Fuselage length	8.75 ft
Wing span	8.92 ft
Wing surface	11.3 ft^2
Weight (with R/C components, 48 oz. Fuel, and payload)	46 lb.
Maximum thrust (approx.)	24.0 lb.
Weight (with R/C components, 48 oz. fuel)	30.2 lb.

Table III – Research Model Characteristics

The onboard computer package, featuring a CPU board, a passive backplane, a 16 MB RAM Disk card, a data acquisition card and a chassis frame, is the core of the aircraft electronic payload since it executes the data acquisition software to acquire the data from all the sensors. The 16 MB RAM disk stores the on-board software and the recorded flight data, which, upon landing, are retrieved through laptop computer via parallel port. The aircraft transmitter/receiver unit features a 10-channel programmable/menu driven system with the capabilities of customizing aircraft controls with a built-in microprocessor. This ground unit allows the pilot to command the conventional aircraft controls with the addition of mixing control surfaces and programming maneuver functions. In terms of the sensor capabilities, the electronic payload features a unit with 3 gyros and 3 accelerometers, an air-data probe with pressure sensors for altitude and airspeed measurements, an aerodynamic vane with potentiometers for the aerodynamic angles, and potentiometers at the hinges of each control surface for the measurement of the surface deflections. Additional components of the payload include batteries and power converters. A total of 33 flights have been performed with 16 flight tests specifically for SFDIA purposes.

Learning-Based SFDIA

Analytical redundancy based methods have been widely discussed in the past. Specific applications have also been developed in the field of flight control systems. In these methods an analytical model of the system is used inside within "observers" to provide estimates of measured variables. This redundancy is then used both to detect a fault and to provide fault tolerance capability to the system.

The most relevant signals that characterize the longitudinal dynamics of an aircraft are:
Angle of attack α [rad];
Pitch rate q [rad/sec];
Normal acceleration a_z [g];
Elevator deflection δ_e [rad].
Velocity relative to the air V [m/s]
Altitude H [m]
The analytical redundancy between these signals is ensured by the equations of motion of the aircraft and in particular by the "normal force equation" below expressed along the body fixed reference frame:

$$ma_z = \rho(H)V^2 SC_z(\alpha,\delta_e,V,a_z/V^2,q/V)/2$$

where m is the aircraft mass, a_z is the normal acceleration measured at the center of gravity, S is the reference surface, ρ is the air density, and Cz is the normal force non-dimensional coefficient (which is a function of α, q/V and δ_e). In this effort it was assumed - without any loss of generality - that a failure can occur only on the sensors measuring α, q, and a_z. The analytical redundancy between the variables involved in the above has been exploited to build up the input-out models required in the SFDIA scheme. In particular, the following estimation models where employed:

$$\hat{\alpha}(k+1) = f_\alpha\left(V(k), H(k), a_z(k)/V^2(k), q(k)/V(k), \delta_e(k)\right)$$
$$\hat{q}(k+1) = f_q\left(H(k), V(k), a_z(k)/V^2(k), \alpha(k), \delta_e(k)\right)$$
$$\hat{a}_z(k+1) = f_{a_z}\left(H(k), V(k), q(k)/V(k), \alpha(k), \delta_e(k)\right)$$

The signals a_z/V^2 and q/V have been used in the above relationships instead of a_z and q because they allow for the non-linearities of the functions to be smoother and, therefore, easier to be identified. In general, since the non-linearities in aircraft dynamics are relevant, the use of non-linear approximators is required.

The purpose of residual generation is to check if the actual measurements fit the process model. This can be accomplished by analysing the trend of the following residual signals $r_i(k)$:

$$r_\alpha(k) = \alpha(k) - \hat{\alpha}(k) + n_\alpha(k) + F_\alpha(k - k_f)$$
$$r_q(k) = q(k) - \hat{q}(k) + n_q(k) + F_q(k - k_f)$$
$$r_{a_z}(k) = a_z(k) - \hat{a}_z(k) + n_{a_z}(k) + F_{a_z}(k - k_f)$$

where $n_i(k)$ is the measurement noise and $F_i(k\text{-}k_f)$ is the additive fault modeling function. Typically this function is different from zero after the occurrence of the fault at the instant $k{=}k_f$. The residual set above has not been designed to exhibit predefined sensitivities to different faults. In other words a single measured quantity should not have impact on a specific residual, based on the structured residuals generation approach. Therefore the models expressed in the two sets of equations can be used as "virtual sensors" or as "residual generators" respectively.

Typically the failure modelling function $F(k\text{-}k_f)$ is described by step and ramp functions representing abrupt and incipient faults (bias or drift) respectively. Within this effort a linear ramp-like function is assumed. Thus, the failure is modelled as follows:

$$F(k - k_f) = \begin{cases} 0 & k < k_f \\ A \cdot (k - k_f)/T_R & k_f \le k < k_f + T_R \\ A & k \ge k_f + T_R \end{cases}$$

where T_R is the duration of the ramp and A is the final value. By varying T_R it is possible to model either hard or soft failures.

The filtered residual vector is processed by an isolation and accommodation logic in order to infer information about the health status of sensors. When the squared filtered residual exceeds a predefined threshold S_{10}, the state of the corresponding sensor is declared suspect (fault *detection*) and a procedure is invoked to decide on the health status of this sensor. At the same time the learning of the NN is stopped to avoid that the NN learns from the "failure-contaminated" measurement. If the filtered residual exceeds another threshold S_{20} the state of the sensor is then declared faulty (fault *declaration*), a failure procedure is enabled, and an accommodated variable $y_a(k)$ is provided as output. Thus, the accommodation procedure substitutes the faulty measurement with the estimation given by the NN. As for any SFDIA approach, the following capabilities are critical:

Failure *detectability* and false alarm rate (the sooner the fault is detected and the least the number of false alarm it is, the better is the SFDI system).
Estimation error (The least is the estimation error, the better is the quality of the accommodation).

The used fault isolation and accommodation scheme is only a step toward the development of a realistic and reliable online scheme, on the other hand, the main purpose of this paper was to investigate, using real flight data, the suitability of different online neural approximators for fault detection purposes rather than focusing on failure identification and accommodation issues.

Neural Networks for On-line Approximation

The two main classes of neural architectures that have emerged in the literature are the Multi Layer Perceptron (MLP) NN and the Radial Basis Function (RBF) Networks NN.

Multi-Layer Perceptron with Extended Back-Propagation (MLP-EBP).

It is a 3-layer architecture NN, whose detailed description has been already done in the paper, with the following sigmoid activation function at each of the neurons:

$$f(net,U,L,T) = \frac{U-L}{1+e^{-net/T}} + L$$

where *U*,*L*, and *T* are the upper bound, the lower bound and the slope at the origin of the activation function respectively. The EBPA updates not only the matrices of the weights between input/hidden layers and hidden/output layers - *W(k)* and *V(k)* respectively - but also the parameters *U,L,T* at the neurons at the hidden and output layers.

"Conventional" Radial Basis Function (RBF)

In the "conventional" RBF NN the estimations $y_s \in \Re^m$ are expressed as a linear combination of *M* Gaussian Basis functions:

$$y_s(x) = We^{-\frac{(x-\mu)^T(x-\mu)}{2\sigma^2}}$$

where $x \in \Re^n$ is the input vector, the parameters μ_j and σ_j are the basis center and width respectively. In the conventional implementation the hidden layer neurons are *a priori* statically allocated on a uniform grid that covers the whole input space and only the weight w_{ij} are updated. This approach requires an *exponentially increasing* number of basis functions versus the dimension of the input space.

Fully Tuned Extended Minimal Resource Allocation Network RBF (EMRAN-RBF).

To avoid the dimensionality problems associated with the conventional RBF, a sequential learning technique for RBF NNs has been introduced in the literature. The resulting architecture was named the Resource Allocating Network (RAN) and has shown to be suitable for online modeling of non-stationary processes with only an incremental growth in model complexity. The RAN learning algorithm proceeds as follows: At each sampling instant, new neurons are added if all the following 3 criteria are met:

Current estimation error criteria: error must be bigger than a threshold:

$$e(k) = y(k) - \hat{y}(k) \geq E_1$$

Novelty criteria: the nearest center distance must be bigger than a threshold:

$$\inf_{j=1}^{M} \left\| x(k) - \mu_j(k) \right\| \geq E_2$$

Windowed mean error criteria: windowed mean error must be bigger than a threshold:

$$\frac{1}{T}\sum_{i=0}^{T}[y(k-T+i) - \hat{y}(k-T+i)] \geq E_3$$

If one (or more) of the above criteria are not met, the existing network parameters (the centers μ_j, the weights w_{ij} and the variances σ_j) are adjusted using the on-line learning algorithm. To avoid an excessive increase of the Network size a *pruning strategy* can also be applied leading to the so-called Minimal RAN (MRAN). Furthermore, the adaptation algorithm is called Extended MRAN (EMRAN) when the parameters are updated following a “winner takes it all” strategy. More precisely, using the EMRAN algorithm only the parameters of the most activated neurons are updated, while all the others are unchanged. This strategy implies a significant reduction of the number of parameters to be updated online with just small performance degradation with respect to the MRAN.

Application to the WVU B777 Flight Data

Four flight data windows of approximately 200 sec each from 4 different flights of the model were used for the SFDIA experiment. The first two sets of flight data were used for off line training purposes while the last two were used for validation purposes. Figure 20 shows a typical time history of *q(k)* and its estimation, during the occurrence of a simulated failure on the pitch gyro at time t_f =150 s using the EMRAN algorithm. To highlight the effects of the failure a “large bias” failure type was selected (A=10 deg/sec T_R=1 sec).

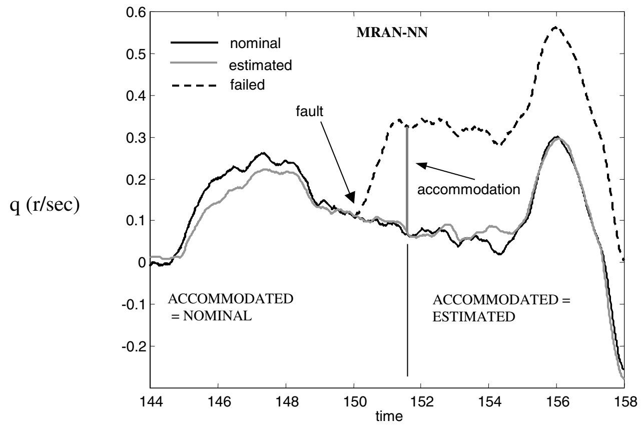


Figure 20 - Failure on the q gyro at t_f = 150 sec

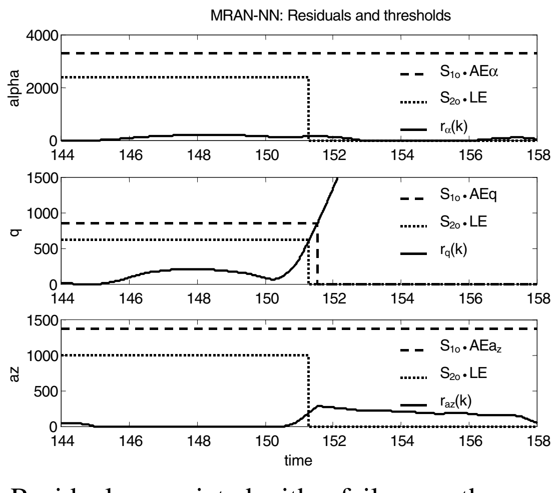


Figure 21 - Residuals associated with a failure on the q gyro at t_f = 150

Figure 21 shows the time histories of the residual signals. The three residuals are generated subtracting the three EMRAN-generated estimations, respectively to q, α, and a_z. After the failure the residuals increase. However, due to the different sensitivities to a fault on the q gyro, only *r_q(k)* exceeds the thresholds. In particular at t=151.2 s the threshold S1o is exceeded; therefore the learning of the three NNs is preventively halted. At t=151.5 s the failure on the sensor *q(k)* is “officially” declared since *r_q(k)* also exceeds the threshold S2o. From this moment on, the on-line neural estimate of *q(k)* is used in lieu of the measurement from the faulty sensor. The failure identification was possible since the residuals *r_az(k)* and *r_α(k)* remain well below their thresholds.

Neural Networks Comparison

The capabilities of detecting small amplitude and slowly drifting bias failures are strictly related to three aspects:

level of noise in the measured signal;
level of accuracy of the estimator;
level of the on-line learning rate.

While the presence of measurement noise is independent of the implemented algorithm, the other two aspects are critically dependent on the performance of the selected estimator. In particular, the selection of the learning rate is reached as the result of a trade-off since it should be high enough to allow the learning of a new aircraft operating condition while, at the same time, small enough to avoid the learning of a faulty measurement from slow drifting fault. On this basis, the performances of the two different NN architectures (EMRAN and MLP-EBP) were compared by varying the level of the learning rate on the data of one validation flight, where at the time $t=t_f=90$ sec a step of 5 deg/sec (7.4% of the maximum value of the pitch rate) was artificially injected on the measured q signal to simulate the effect of a sensor failure.

The comparison of the performance of the two schemes is performed through the evaluation of different SFDIA indicators: TLE and TAE. TLE is the time in which a sensor is declared suspect while TAE is the time at which it is finally declared faulty. To evaluate the performance of the detection/identification system, the detectability ratio (DR) between the main peak of the filtered residual signal during the failure transient ($90 < t < 100$) and the peak of the filtered residual before failure ($0 < t < 90$) is considered. This ratio quantifies the detectability provided by the scheme.

Table IV – EMRAN-RBF; ηo=0.0001

L.R.(η	0	ηo	2ησ	4ηo
TLE	93	93.39	93.71	94.71
TAE	93.89	94.25	94.97	95.95
%LE	0	0	0	25.14737
%AE	0	0	0	25.61053
DR	11.63651	11.49651	12.35879	12.19832
MEE	-1.73646	-2.04047	-2.39899	-3.95824
STDEE	2.452785	2.407548	2.369341	2.202287
POWEE	9.031454	9.959815	11.36895	20.51775

Table V – MLP-EBPA; ηo=0.00006

L.R. (η	0	ηo	2ησ	4ηo
TLE	94.74	94.22	94.32	95.32
TAE	95.82	95.66	95.76	96.44
%LE	3.9	0.726316	0.084211	7.942105
%AE	0	0	0	9.657895
DR	3.588593	4.149321	4.352193	3.53532
MEE	-2.30202	-2.29437	-2.44726	-3.49016
STDEE	2.322125	2.322165	2.320881	2.167919
POWEE	10.69157	10.65659	11.37558	16.88107

Furthermore, the time percentage %LE in which a sensor is declared “suspect” *before* a true failure detection is reported to assess the false detection (false alarm) rate. Similarly the time percentage %AE in which a sensor is declared “faulty” *before* a true failure declaration is evaluated to assess the false accommodation rate. Finally, the parameters *MEE(k), STDEE(k)* and *POWEE(k)* are introduced to evaluate the goodness of the accommodation system. These parameters represent the mean, the variance and the power of the absolute estimation error over the whole data file.

Table IV and V show these results for the two architectures. Analysing the results in Table V, some conclusion can be drawn for the MLP approximator. For moderate values of the learning rate a substantial decrease in false alarms is observed. An increase in the detectability ratio DR can be seen without a corresponding increase in the power of the error (POWEE). The price to pay is generally an increase in the delay for failure declaration (TLE, TAE); however, this effect is relatively modest. These results highlight the benefit of introducing an on-line learning scheme, at least for MLP approximators, to improve the degree of reliability of the detection scheme and to guarantee satisfactory performances in all the working conditions. However the learning rate cannot be increased arbitrarily; in fact, an excessive value increases the possibility that the SFDIA scheme "learns" a faulty pattern before a failure is declared and properly accommodated. This risk is confirmed by the analysis of the last columns in the tables, where high learning rates prevented the detection of the failures. This is also consistent with a significant decrease in the detectability ratio. Furthermore higher learning rates compromise the integrity of the global approximation capability of the NN, which can have adverse effects at post-failure conditions when the neural approximator has to supply reliable estimates using only the learned data. The fact that for low learning rates the post-failure POWEE index takes on small values shows the possibility of using the neural estimator as "virtual sensor" after a failure declaration. The RBF-EMRAN NNs have generally shown better performance. Their approximation and generalization capabilities are confirmed by the absence of false alarms even without learning, by the high values of the detectability ratio, and by the lowest values of the POWEE index.

Fault Tolerance in Flight Management

Management of the aircraft formation can be centralized or decentralized. In the former case, one formation manager exists that acts as a supervisor for all the aircraftand manages the topology of the channels they use to exchange information among themselves. This manager can be one of the aircraft or ground-based. The major advantage of ground-based management is the capability to react to and reach decisions from a possibly higherlevel of "intelligence" than that achievable by on-board computers. In fact, human decision could be added to the automatic system but this surely introduces a delay in the reconfiguration progress. A prompt response is still needed to bring the formation into a safe configurationprior to a ground-based decision. In a decentralized management scheme, on the other hand, each aircraft is given a certain level of decision capability, while the whole formation must be capable of reconfiguring, making decisions, and achieving mission goals. In both cases it is important to establish general rules for the information exchange and graph theory can be used to describe inter-aircraft communication. The aircraft can be thought of as the nodes of a graph. The physical communication channels create the arcs in the graph. These arcs are oriented because, in the most general case, channels are not bi-directional; this is not a limitation, since two nodes can be connected by two opposite direction arcs to model a bi-directional channel. The graph must also be connected because, if two subgraphs exist without any arc connecting them, the aircraft inside the two groups cannot behave like a formation component, yielding in fact two separate formations. The communications graph should be redundant from the standpoint of the capability of propagating information. In the event of failures, this redundancy leaves room for reconfiguration. However, having the capability of using a channel does not mean that it must be used at all times. Optimization of available channels under a cost function constraint can be achieved via graph programming techniques.

Graph Theory Optimization

The problem of mapping a formation structure with graph theory can be configured as a shortest path optimization problem. Suppose that at least one of the aircraft knows the mission reference trajectory. This reference trajectory can be seen as the effective formation leader and can be represented in the graph by a node called Virtual Leader (VL), so that each aircraft that knows the mission trajectory has one communication channel with the VL. If a cycle exists, since the graph must be connected, it must contain all nodes. However because at least one node has the VL as its only reference, and the VL has no incoming arcs by definition, the VL and all nodes using the mission trajectory as reference cannot be part of a cycle. Thus the graph cannot have any cycle, and the VLwill be the root of the solution tree.

To solve the shortest path problem, the Dijkstra algorithm was used. It has polynomial complexity, it guarantees optimality of the solution, and it is deterministic; that is, starting from the same conditions, all runs lead to the same solution. The Dijkstra algorithm can be modified to obtain the optimal redundant channels among those outside theoptimal arcs set. Suppose we need *m* total possible channels for each node; that is, *m-1* redundant channels. The unmodified algorithm can still be used to find the optimal solution. At the end of the optimization procedure, the nodes' potentials are frozen. Then for all nodes *i* in the S set, select *m-1* incoming arcs that have the minimum value of *d'(i) = d(j)+Cji*, where *j* is a possible redundant preceding node, and order for increasing values of *d'(i)*. This modification computes sub-optimal solutions. The arcs chosen with this technique can be considered suboptimal because they are the second-best choices. After removing the incoming arc of a node belonging to the optimal path, the new optimal incoming arc that a new run of the Dijkstra algorithm will produce is the first one of this redundant arcs list. Figure 22 shows the graph of a sample formation. While the VL is connected with all aircraft, not all possible inter-aircraft connections are present. An arc's weight is identified by a positive number. The figure also shows the result of the modified Dijkstra algorithm. The optimal path, in solid line, and each node. In this case, *m=3* but a node might not have the two required redundant channels because it has fewer than m incoming arcs; this is the case for node 1. When node A switches reference and uses a sub-optimal path, its potential increases, and a reconfiguration procedure must be run on all the nodes of the sub-tree that originates from that node. In fact, a new global optimization would be needed, but the potential of all the nodes that do not have any incoming optimal path from A cannot be affected, because these surely have a lower cost path to the VL. Since the detected failure and the new node A potential after reconfiguration must be propagated to the nodes that belong to the optimal A sub-tree path only, the same communications channels used for formation-keeping data can also be used to exchange potential updates. Furthermore, since after all nodes have completed the reconfiguration, the new graph is optimal, this procedure can be repeated in case of successive faults without ever having to reconsider optimization of the whole graph. This means that any number or combination of successive faults can be managed with this sub-tree-based technique without compromising whole-graph optimality.

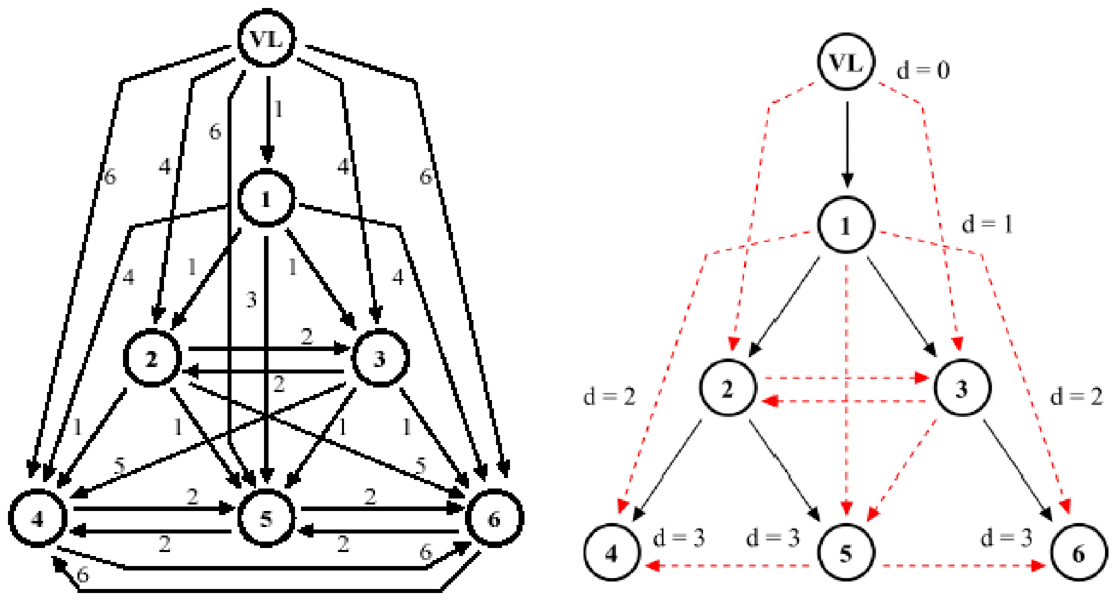


Figure 22. Fault Tolerant Communications Link for Formation Flight

The following simulation presents the reconfiguration of the formation, shown in figure 22, following a fault in the aircraft 3 transmitter. The failure happens during a 180-deg turning maneuver. As shown in the figure, aircraft 6 uses aircraft 3 as the reference for its formation-keeping control system. The first redundant channel for aircraft 6 is that coming from aircraft 5. When aircraft 6 detects a fault, it reconfigures itself to use aircraft 5 as reference. Figure 23 shows the distance of aircraft 6 from its ideal trajectory compared to that of the same simulation without failure. The red line shows that, after the fault at = *50* sec, a fast transient is followed by a reconfiguration that brings the aircraft back into a stable position inside the formation. The two traces of distance error do not have to join, in this case, because the performance of the controller is different.

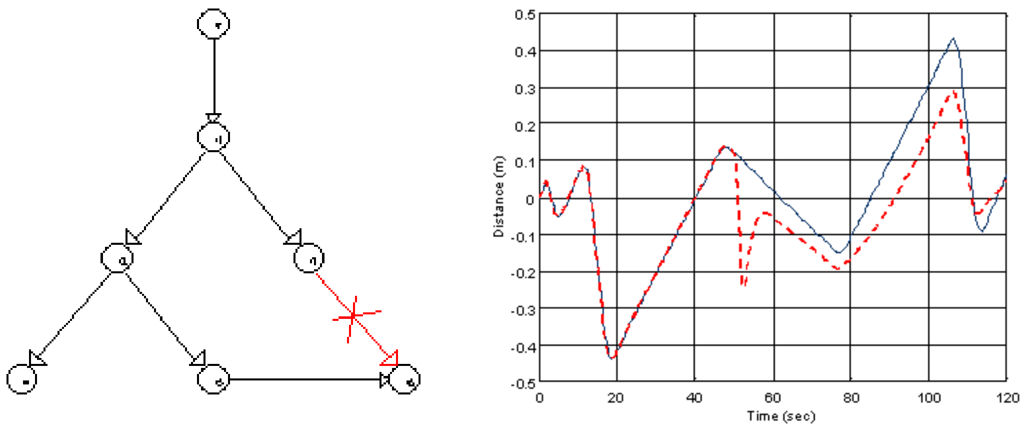


Figure 23. Failure Reconfiguration during Formation Flight

Conclusions

This paper has presented some results of a study undertaken with the goal of showing the potential of online learning neural networks for adaptive control at non-linear conditions. Particularly the paper has shown successful simulations of identification and (identification + control) using an indirect approach for non-linear SISO systems. The paper presented also some issues related to the use of neural architectures in the area of sensor and actuator failure detection and identification. The main focus of the work was the problem of achieving satisfactory performance for on-line learning, and subsequent potential for on-board implementation on flight vehicles having low and/or no physical redundancy. To this end standard gradient-based neural networks were considered and adapted to the above constraints. A NN-based scheme SFDIA has been analyzed using experimental flight data from the WVU B777 research aircraft model. T he scheme was implemented with 2 neural architectures, that is the MLP-EBPA NN and the RBF-EMRAN NN. The scheme has shown to be successful in the detection, isolation and accommodation of failures “injected” on the WVU B777 flight data. Finally some issues on failure accomodation in formation flight management have been addressed using linear programming techniques.

Acknowledgement

This work was supported in part grants from the NASA West Virginia Space Consortium, the Institute for Software Research, and the Italian Ministry of Education and University Research.

References

1. Astrom, K.J, Wittenmark, 1989, "*Adaptive Control*", Addison-Wesley Publishing Co.
2. Barnard, E.,1992 "Optimization for Training Neural Nets", *IEEE Transactions on Neural Networks*, Volume 3, Number 3, pg. 232
3. Casdorph, V.A., 1994 "Detection and Identification of Battle Damage and/or Generic Failure Using On-Line Learning Neural Networks and Cross Correlation Functions", Master Thesis, West Virginia University, USA.
4. Chen, C.L., Nutter, R.S., 1992 "An Extended Back-Propagation Learning Algorithm by Using Heterogeneous Processing Units", IJCNN92, Baltimore, Md, USA.
5. Chen, V.C., Pao, Y.H., 1989, "Learning Control with Neural Networks", Proceedings of the International Conference on Robotics and Automation.
6. Cybenko, G. , 1989, "Approximation by Superposition of Sigmoidal Functions", *Mathematics of Control Signals and Systems*, Vol.2, No.4, pg. 303.
7. De Villiers,J., Barnard, E., 1993, "Back-Propagation Neural Nets with One and Two Hidden Layers", *IEEE Transactions on Neural Networks*, Volume 4, Number 1, pg. 136.
8. Freund, E., 1975, "The Structure of Decoupled Non-Linear Systems", *International Journal of Control*, Vol. 21, pg.651.

9. Hornik, K., Stinchcomb, M., and White, H., 1989, "Multilayer Feedforward Networks Are Universal Approximators", *Neural Networks*, Vol.2, pg.359.
10. Hunt, K.J., Sbarbaro, D., Zbikowski, R., and Gawtrhop, P.J., 1992 "Neural Networks for Control Systems - A Survey", *Automatica*, Vol. 28, No. 6, pg.1083.
11. Isidori, et al., A., 1981, "Non-Linear Decoupling Via Feedback : a Differential Geometric Approach", *IEEE Transactions on Automatic Controls*, Vol. AC-26, pg.341.
12. Kawato, M., Furukawa, F., and Suzuki, R., 1987, "A Hierarchical Neural Network Model for Control and Learning of Voluntary Movement", *Biological Cybernetics*, Vol.57, pg.169.
13. Napolitano, M.R., Chen, C.I., and Naylor, S., 1993, "Aircraft Failure Detection and Identification Using Neural Networks", *AIAA Journal of Guidance, Control and Dynamics*, Volume 16, Number 6, pg. 999.
14. Narendra, K.S., Monopoli, R.V., 1980 "*Applications of Adaptive Control*", New York Academic Press.
15. Narendra, K.S., Partasarathy, K., 1990, "Identification and Control of Dynamical Systems Using Neural Networks", *IEEE Transactions on Neural Networks*, Vol.1, No.1.
16. Naylor, S.M., 1994, "On-Line Learning Non-Linear Neural Controllers for Restructurable Flight Control Systems", Master Thesis, West Virginia University, Morgantown, WV, USA.
18. Neppach, C.D., 1994, "Application of Neural Networks in Sensor Failure Detection, Identification and Accommodation in a System Without Sensor Redundancy", Master Thesis, West Virginia University, Morgantown, WV, USA.
19. Nielsen, R.H., 1991, "*Neurocomputing*", Addison Wesley Publishing Company.
20. Ogata, K., 1987, "*Discrete-Time Control Systems*", Prentice-Hall, Inc.
21. Psaltis, D., Sideris, A., and Yamamura, A., 1988, "A Multilayered Neural Network Controller", *IEEE Control Systems Magazine*, Vol. 4, pg.17.
22. Sastry, S.S., Isidori, A., 1989 "Adaptive Control of Linearizable Systems", *IEEE Transactions on Automatic Controls*, Vol. 34, No.11, pg.1123.
23. Simpson, P.K., 1990, "*Artificial Neural Systems*", Pergamon Press Inc., Fairview Park, NY.
24. Singh, S.N., and Rugh, W.J., 1972, "Decoupling in a Class of Non-Linear Systems by State Variable Feedback", *ASME Transactions*, Vol. 94, pg.323.
25. Rumelhart, D. and McClelland, J., 1986, "*Parallel Distributed Processing*", MIT Press, Cambridge, MA,USA.
26. Venugopal, K.P., et al., 1994 "An Improved Scheme for Direct Adaptive Control of Dynamical Systems Using Backpropagation Neural Networks", accepted for publication to be published in "*Journal of Circuits, Systems and Signal Processing".*
27. Widrow, B., and Stearns, S.D., 1985, "*Adaptive Signal Processing*", Prentice Hall, Inc.
28. NASA, 1993,"Cooperative Agreement Award No. NCCW-0051", with West Virginia University.
ASI, 1995-1997, "Sviluppo di Architetture Neurali Software e Hardware per la Identificazione in Linea di Guasti in Sensori e Attuatori di Veicoli Aerospaziali", Contratto di Ricerca Agenzia Spaziale Italiana-Consorzio Pisa Ricerche.
30. Willsky, A.S., 1976, "A Survey of several Failure Detection Methods", *Automatica*, Vol. 12, No. 6.
31. Guo, T., Nurre, J., 1991, "Sensor Failure Detection Recovery using Neural Networks", *Proc. of the IJCNN '91*.
32. Napolitano, M.R., Innocenti, M., et. al, 1994, "Sensor Failure Detection, Identification, and Accomodation using a Neural Network based Approach", *Proc. of the AIAA Guidance, Navigation, and Control Conference*, Scottsdale, AZ, August.
33. Silvestri, G., 1995, "Implementazione real-time di reti neurali in applicazioni di identificazione e di controllo in presenza di malfunzionamento sui sensori", Ph.D. dissertation, University of Pisa, Italy.
34. Napolitano, M.R., Innocenti, M., et. al., 1995, "Neural-Network-Based Scheme for Sensor Failure Detection, Identification, and Accomodation", AIAA *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 6.
35. Napolitano, M.R., Casdorph, V., Neppach, C., Naylor, S., Innocenti, M., Silvestri, G, 1996, "Online Learning Neural Architectures and Cross-Correlation Analysis for Acutator Failure Detection, and Identification", International Journal of Control, January.
36. Napolitano, M.R., Casdorph, V., Neppach, C., Naylor, S., Innocenti, M., Silvestri, G, 1994, "Implementation of Hardware-Based On-Line Neural Architectures for Sensor Failure Detection, Identification, and Accomodation", submitted for publication to the *AIAA Journal of Guidance, Control, and Dynamics*, October.

37. Napolitano, M.R.m Windon, D., Casanova, J., Innocenti, M., 1996, "A Comparison between Kalman Filter and Neural Network Approaches for Sensor Validation", AIAA-96-3894, Guidance and Control Conference, San Diego, California, August.
38. Del Gobbo, D., 1996, "Tecniche avanzate per la Rilevazione di Malfunzionamenti", Master thesis, Universita' di Pisa, Italy.
39. Patton R.J., Frank P.M., Clark RN. "*Fault diagnosis in dynamic systems: Theory and applications",* Englewood Cliff, NJ, Prentice-Hall, 1989.
40. Baruh H., Choe K. "Sensor-Failure Detection Method for Flexible Structures", *AIAA Journal of Guidance, Control, and Dynamic 1987*; Vol. 10, no 5, 474-482.
41. Kerr T.H. "False Alarm and Correct Detection Probabilities over a Time Interval for Restricted Classes of Failure Detection Algorithms", *IEEE Transactions of Information Theory 1982*; IT-28, No. 4, pp. 619-631.
42. Napolitano M.R., Neppach C.D., Casdorph V., Naylor S., Innocenti M., Silvestri G. "A Neural Network Based Scheme for Sensor Failure Detection, Identification and Accommodation", *AIAA Journal of Guidance Control and Dynamics 1995;* 18(6) 1280-1286.
43. Hunt K.J., Sbardato D., Zbikowski R., Gawthrop P.J. "Neural Networks for Control Systems: a Survey" *Automatica 1992;* 28(6), pp. 1083-1112.
44. Napolitano M.R., An Y., Seanor B. "A Fault Tolerant Flight Control System for Sensor and Actuator Failure using Neural Networks", *Aircraft Design 2000;* Vol. 3, pp. 103-128.
45. Vemuri A., Polycarpou M., Diakourtis S. "Neural Network Based Fault Detection and Accommodation in Robotic Manipulators", *IEEE Transaction on Robotics and Automation* 1998; Vol. 14, no. 2, pp. 342-348.
Napolitano M.R. "*Design of the B777 Model Flight Control System*". WVU Internal Technical Report, March 1999.
46. Isermann R., Balle' P. "Trends in the Application of Model-Based Fault Detection and Diagnosis of Technical Processes", *Control Engineering Practice 1997;* Vol. 5, no. 5, pp. 709-719.
47. Lu Y., Sundararajan N., Saratchandran P. "Analysis of Minimal Radial Basis Function Network Algorithm for Real-Time Identification of Non-linear Dynamic Systems". *IEEE Proceedings on Control Theory and Application 2000;* Vol. 4, no. 147, pp. 476.

Airfoil Design Using Navier-Stokes Equations and Hybrid Evolutionary Optimization Techniques

Domenico Quagliarella[‡], **Domenic D'Ambrosio**[⋆], **Angelo Iollo**[⋆]
[‡] Centro Italiano Ricerche Aerospaziali
Via Maiorise, 81043 Capua, Italy
[⋆]Dipartimento di Ingegneria Aeronautica e Spaziale, Politecnico di Torino
C.so Duca degli Abruzzi 24, 10129 Torino, Italy
d.quagliarella@cira.it, domenic@athena.polito.it, iollo@polito.it

1 Introduction

An application of hybrid optimization techniques to airfoil design with Navier-Stokes equations is here presented and discussed. The greatest obstacle in the use of Navier-Stokes equations with evolutionary optimization procedures is the huge computational effort required. To overcome this limit, several approaches have been suggested, that range form parallel genetic algorithms [1, 2] to approximated fitness evaluation [3, 4] to hybrid techniques [5].

These three approaches can be profitably combined and an example will be here given related to transonic airfoil design.

The lecture is organized as follows: an outline of hybrid optimization techniques will be given then the parallel hybrid genetic algorithm will be presented. Thereafter an application example will be reported, and finally some ongoing developments about the hybrid approach based on gradient computation through the adjoint to Navier-Stokes equations will be described.

2 Hybrid genetic algorithms

Hybridization has been one of the first techniques adopted for improving genetic algorithm performance, while keeping the desirable flexibility features of genetic algorithms [6].

In the optimization context, hybridization can be defined as the mixing of two or more search techniques with, possibly, complementary features.

The genetic algorithm itself, even in its simplest implementation, can be considered as a hybrid optimization technique, where selection, mutation and crossover cooperate in the same optimization process.

Figures 1 and 2 show how the genetic algorithm can be extended through the addition of a hill climbing operator. In the first scheme the hill climber receives in input the whole population belonging to the current generation, while in the second one an intermediate selection process chooses a subset of elements that will be refined through hill-climbing.

A hybrid algorithm requires care in balancing the various components of the search procedure. Indeed, if high-rated solutions are injected in the population at an early evolution stage, this may adversely affect population diversity and force the evolution in wrong directions. The way to avoid this depends, in general, on the particular optimization technique introduced. If a gradient based technique is used as a solution refinement operator, it may be useful to balance the improvement rate of both techniques by stopping the hill climber after a few iterations. However, the iteration number choice is mostly a question of practice and experience, and it should take into account both optimization algorithm and problem features.

Another point that requires special care when using hybrid techniques is encoding. Often, indeed, a specialized optimization technique relies on a particular encoding of the problem variables that may play a key role in the efficiency of the method. Therefore, whenever possible, it is worthwhile to extend to the genetic algorithm

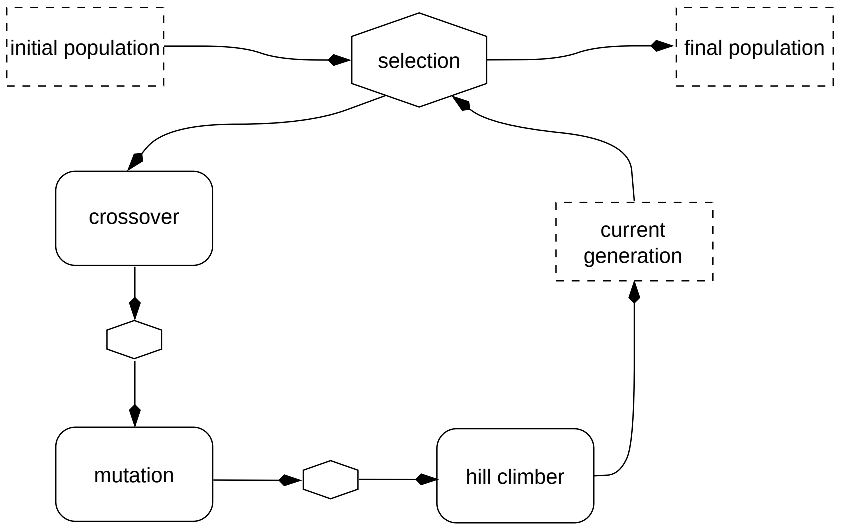


Figure 1: A simple hybridization scheme between a genetic algorithm and a hill climber.

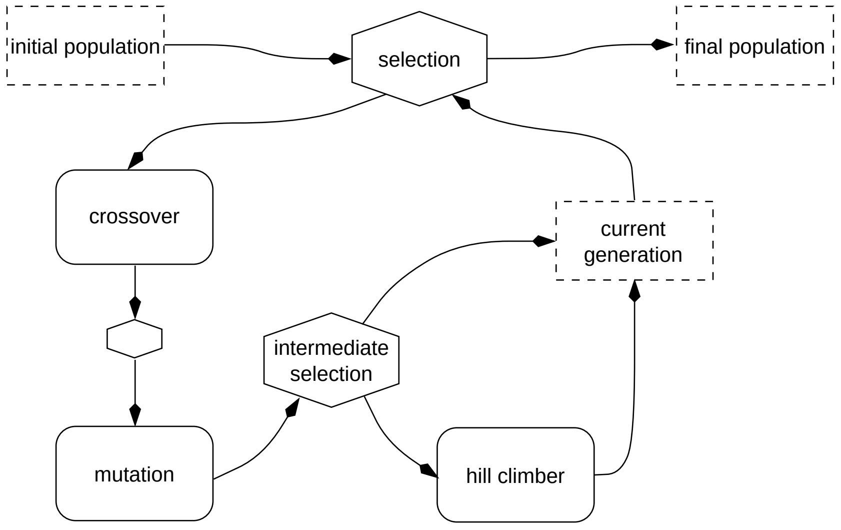


Figure 2: A more complex genetic-hill climber coupling scheme.

the encoding of the specialized technique. On the other hand, the simple binary encoding used by many genetic algorithms may cause problems to the specialized algorithms if the conversion of data is not carefully handled. For example, if a binary genetic algorithm does not use enough bits for variable encoding, the improvement obtained using a gradient based hill climber may be lost when re-encoding the variables in binary form.

The hill climber operator adopted here is a gradient search based on BFGS algorithm [7]. Indeed, the fitness functions of the presented applications are differentiable, for which gradient based techniques are much more efficient to locally improve a given solution. The genetic algorithm developed adopts a bit string codification of the design variables; anyway, this does not prevent the use of operators requiring real number list encoding, such as extended intermediate crossover and word level mutation [8]. In this cases, the binary string is decoded into a real numbers list, the operator is applied and the modified variables set is encoded back into a bit string. This scheme allows the use of a free mix of different type of operators. The hybrid genetic algorithm operates as follows: through the application of the selection, crossover and mutation operators, an intermediate generation is created from the current one; afterwards, if the hybrid option is activated, some randomly chosen individuals are fed into the BFGS based operator to be improved, and then introduced into the new generation.

3 Parallel genetic algorithm

The hybrid genetic algorithm here described has two points that can be executed in parallel: the evaluation of the new population members after the mutation and recombination phase and the evaluation of the gradient through finite differences in the gradient based hybrid operator. The parallel programming model adopted relies on shared memory multiprocessing and the parallelism is implemented at the thread level using the standard POSIX thread interface.

The same code base works on a SGI POWER CHALLENGE system with 16 R-10000 processors and on a Linux PC-cluster with eight processors and the MOSIX clustering software [9].

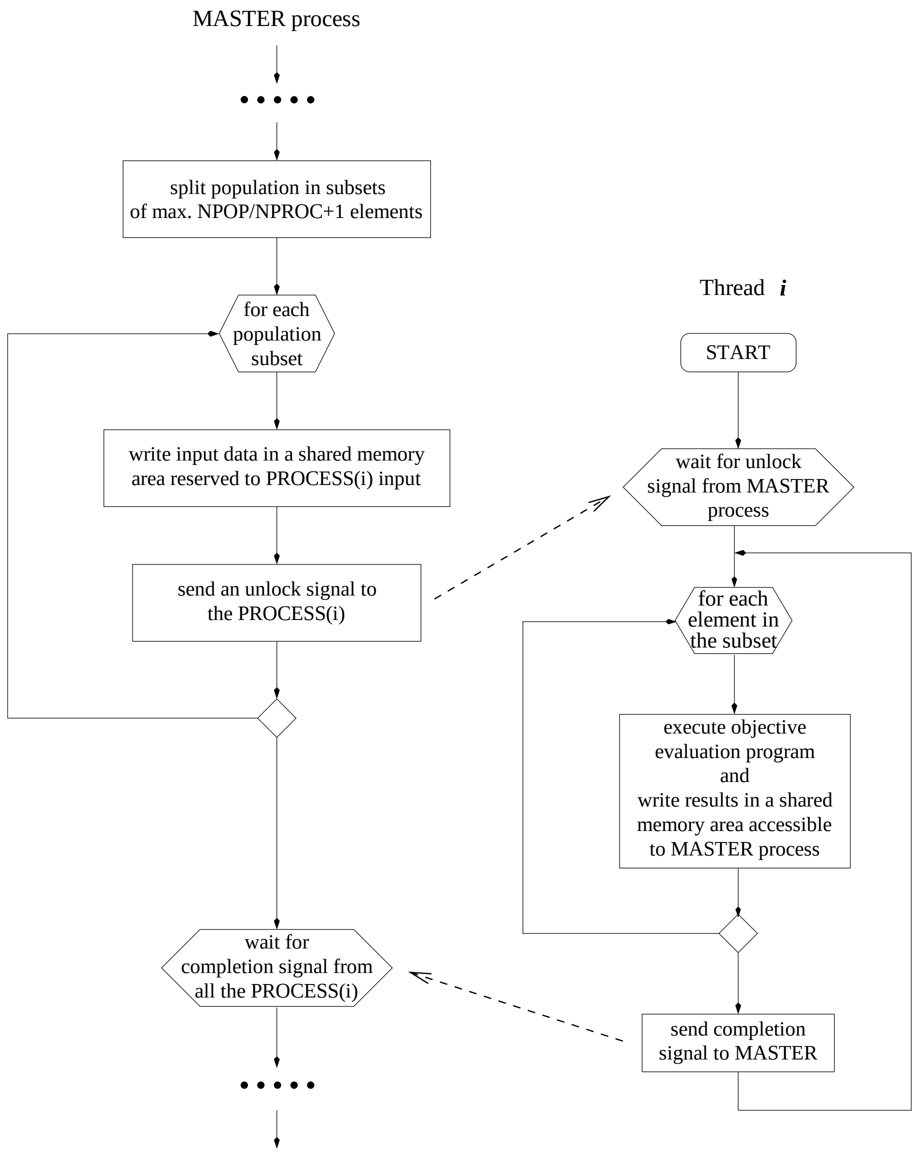

MASTER process
split population in subsets of max. NPOP/NPROC+1 elements
for each population subset
write input data in a shared memory area reserved to PROCESS(i) input
send an unlock signal to the PROCESS(i)
wait for completion signal from all the PROCESS(i)
Thread *i*
START
wait for unlock signal from MASTER process
for each element in the subset
execute objective evaluation program and write results in a shared memory area accessible to MASTER process
send completion signal to MASTER

Figure 3: Parallel evaluation loop.

The algorithm is organized following the master-slave paradigm. Figure 3 shows the architecture of the parallel evaluation loop.

In the initialization phase that precedes the first execution of the evaluation loop, the master process creates a pool containing a number of processes, equal to the maximum number of threads available for the computation (NPROC). The child threads are immediately put in a wait state, and they will remain in such a state until they receive a “go ahead” signal from the master to start the computation. This choice avoids the inefficiency of creating a child process every time a computation is needed and killing it at the end.

When the master process enters the evaluation loop, it splits the population, of size NPOP, in slices of maximum NPOP/NPROC+1 elements and then each slice is assigned to one of the child threads. Afterwards, a signal is sent to the child, through a standard POSIX semaphore, so that it begin to evaluate its slice. When the child terminates its computation, it sends a completion signal to the master (using another POSIX semaphore). The master waits for the completion of all child processes in a synchronization point. The child processes are terminated at the very end of the program, when all the evaluation loops related to each generation and gradient evaluation loop have been completed.

This architecture has maximum efficiency when each thread has an even computational charge. If this is not the case, the computation process can loose efficiency because the master has to wait at the synchronization point for the completion of the slowest process. A partial solution to this problem consists in activating more threads than available processors (e.g. 8 threads on a four processor machine). If this is not possible, then the algorithm should be modified in order to redistribute dynamically the computational charge to the child threads.

4 Aerodynamic flow solvers

Two flow solvers with different level of accuracy have been adopted for the objective function computation of the presented design exercises. The first one is the in-home developed ZEN Navier-Stokes solver [10] with Baldwin Lomax turbulence model [11]. The second one is Drela's MSES code [12] that is based on a finite-volume discretization of the Euler equations on a streamlined grid. The viscous region is computed using an integral boundary layer based on a multi-layer velocity profile representation. The inviscid and viscous regions are coupled using displacement thickness.

The boundary layer code is used both as a low-fidelity solver, and as a helper for ZEN code. Indeed, when constant lift coefficient (c_l) is required, a suitable angle of attack for ZEN is guessed using MSES; after a first computation with ZEN, the angle of attack is corrected using the $\partial c_l/\partial \alpha$ computed with MSES.

5 RAE 2822 airfoil optimization

The optimization problem requires the minimization of the drag coefficient $obj = c_d$, with control on lift coefficient and maximum thickness. The design point is fixed at: $M = 0.68$, $Re = 5.7 \times 10^6$, $c_l = 0.56$. The starting geometry is the RAE 2822 airfoil. Transition is fixed to $x/c = 0.01$.

The airfoil shape is defined as a linear combination of an initial geometry $y_0(x)$ and some modification functions, $f_i(x), i = 1, \ldots, n$:

$$y(x) = y_0(x) + \sum_{i=1}^{n} w_i f_i(x) \tag{1}$$

where w_i are the design variables.

The modification functions used here are reported in Table 1 with $\xi = x/c \in [0, 1]$.

5.1 Optimization using simple GA and Euler+BL

A first optimization run has been performed using the genetic algorithm without hybrid operators, and the MSES solver only. The geometry was represented using 20 design variables (10 for the upper surface and 10 for the lower) chosen among the polynomial, Hicks-Henne and Wagner functions. Three subsequent runs of the parallel GA were performed, and 10 threads were active in each run. The GA parameters are reported below:

Variables encoding	32 bit
Selection	3 step R. W.
Crossover	one-point, $p_c = 1$
Mutation	bit level, $p_m = 0.01$
Population size	40
Generations	40

Hicks-Henne	Legendre
$0.888\,(1-\xi)\,\sqrt{\xi}e^{-13.28\xi}$	$0.42\,(1-\xi)^3\,\sqrt{\xi}$
$0.57\,(1-\xi)\,\sqrt{\xi}e^{-5\xi}$	$0.946\,(1-\xi)^3\,\xi$
$0.1\sin^3\left(\pi\xi^{0.431}\right)$	$0.136\,(1-\xi)^3\,\left(12\xi-10\xi^2\right)$
$0.1\sin^3\left(\pi\xi^{0.757}\right)$	$(1-\xi)^3\,\left(225\xi^5-630\xi^4+560\xi^3-220\xi^2+30\xi\right)$
$0.1\sin^3\left(\pi\xi^{1.357}\right)$	
$0.1\sin^3\left(\pi\xi^{3.106}\right)$	
Linear	**Wagner**
0.2ξ	$0.87\left(\frac{2\arcsin(\sqrt{\xi})+\sin(2\arcsin(\sqrt{\xi}))}{\pi}-\xi\right)$
	$0.24\left(\frac{\sin(2k\arcsin(\sqrt{\xi}))}{k\pi}+\frac{\sin(2(k-1)\arcsin(\sqrt{\xi}))}{\pi}\right)\ k=2,\ldots,6$
Polynomial	**Rear loading**
$0.52\,\left(0.5\xi^3-1.5\xi^2+\xi\right)$	$6625000\,(1-\xi)\,\xi^{15}e^{1/5-20\xi}$
$0.4\,\left(\xi-\xi^2\right)$	$17500000\,(1-\xi)\,\xi^{18}e^{1/4-20\xi}$
$0.52\,\left(0.5\xi-0.5\xi^3\right)$	$44440000\,(1-\xi)\,\xi^{22.66}e^{7/20-20\xi}$
	$90000000\,(1-\xi)\,\xi^{30}e^{1/2-20\xi}$

Table 1: Modification functions used in the design examples.

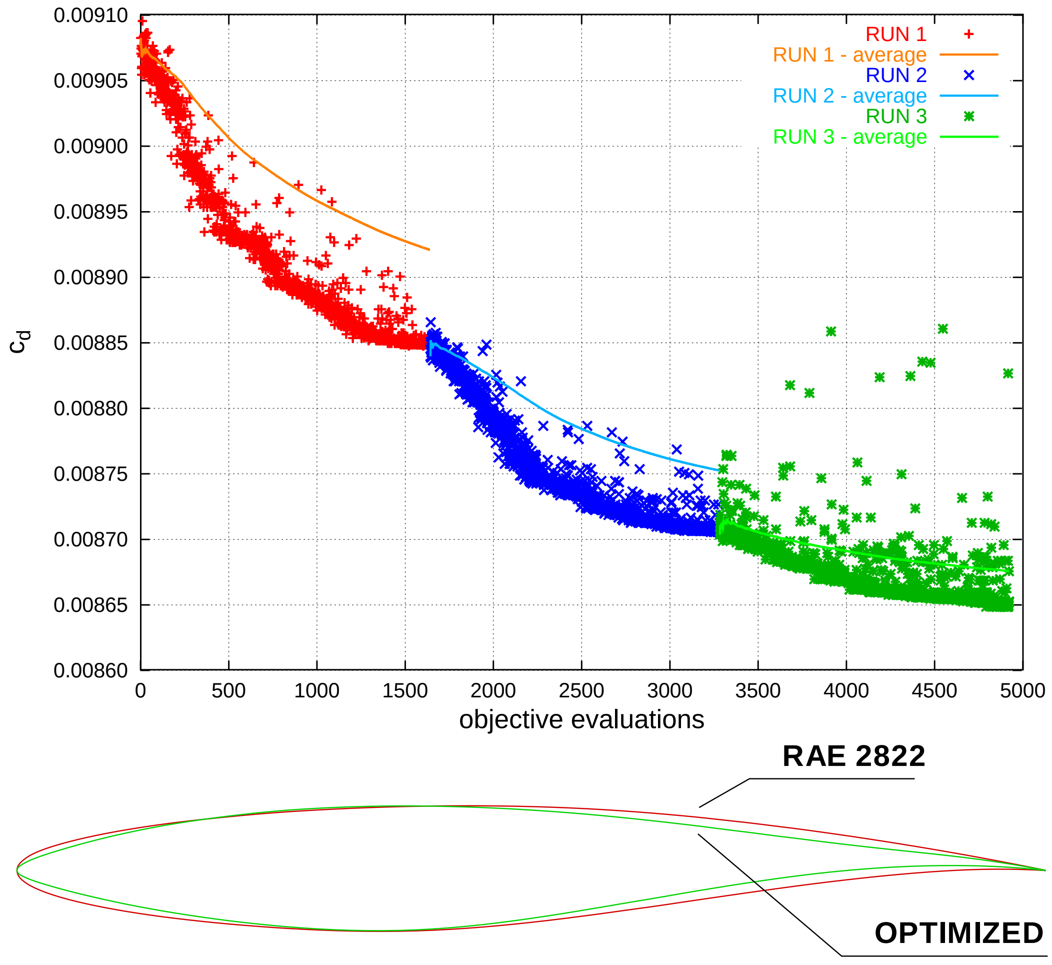


Figure 4: Optimization history and obtained shapes in the Euler+Bl optimization runs.

The optimization history, and the initial and final shapes are reported in figure 4. The initial drag coefficient was $c_d = 0.009070$ at $\alpha = 1.9216°$, while the final one resulted to be $c_d = 0.008645$ at $\alpha = 1.5619°$. A subsequent analysis with the ZEN NS flow solver on the same configurations, however, showed different trends, and there was no appreciable difference between the drag coefficients of the two configurations. Therefore, it was decided to use the initial RAE 2822 as starting point for the subsequent optimization pass with the Navier-Stokes solver.

5.2 Optimization using hybrid GA and NS

The result of the previous run were used to select the design variables that had the strongest effect on the objective function.

A 256×56 C-shaped grid was used for each Navier-Stokes run. A three level multigrid acceleration strategy was used.

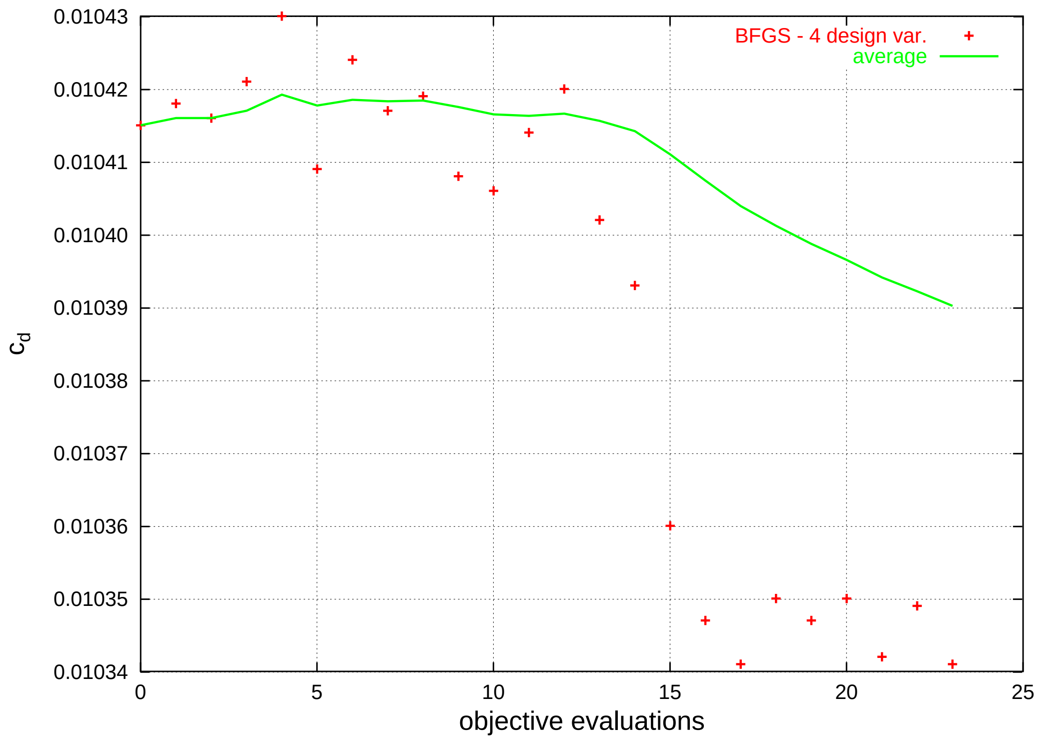


Figure 5: Optimization history with BFGS algorithm, Navier-Stokes equations for drag evaluation and four design variables.

A first set of 4 design variables was identified and a simple BFGS run was performed. The maximum number of BFGS steps was fixed to 10. The optimization history is reported in figure 5. The optimization process terminated when the convergence criteria were meet, after three gradient evaluations and a total of 23 objective function evaluations. As can be observed, a small drag reduction was obtained (0.7%). It is worth to note that in figure 5, as well as in all the following evolution histories reported, all the objective function evaluations required by the optimization algorithm are reported, including those used to compute the gradient by finite differences.

The number of design variables was then extended to eight. Here three different run were performed, The first one was a simple GA with 8 population members that ran for 15 generations. The second one was a hybrid GA that ran for 4 generation, with a population size of two elements. The BFGS was applied to each element, and the number of BFGS steps was fixed to 10. This produced a total of 62 objective function evaluations. The last hybrid GA was instead characterized by a population of eight elements and the BFGS activation probability was set to 6% with three descent steps allowed. After 7 generation the optimizer required 82 objective calculations. The evolution history for these three runs is reported in figure 6. In run 3 the best solution had a drag coefficient equal to 0.010295 while the starting one was $c_d = 0.010415$. The c_p distribution of the original and the optimized airfoil are reported in figure 7, the c_f comparison is reported in figure 8, and, finally, the Mach field around the airfoil is reported in figure 9.

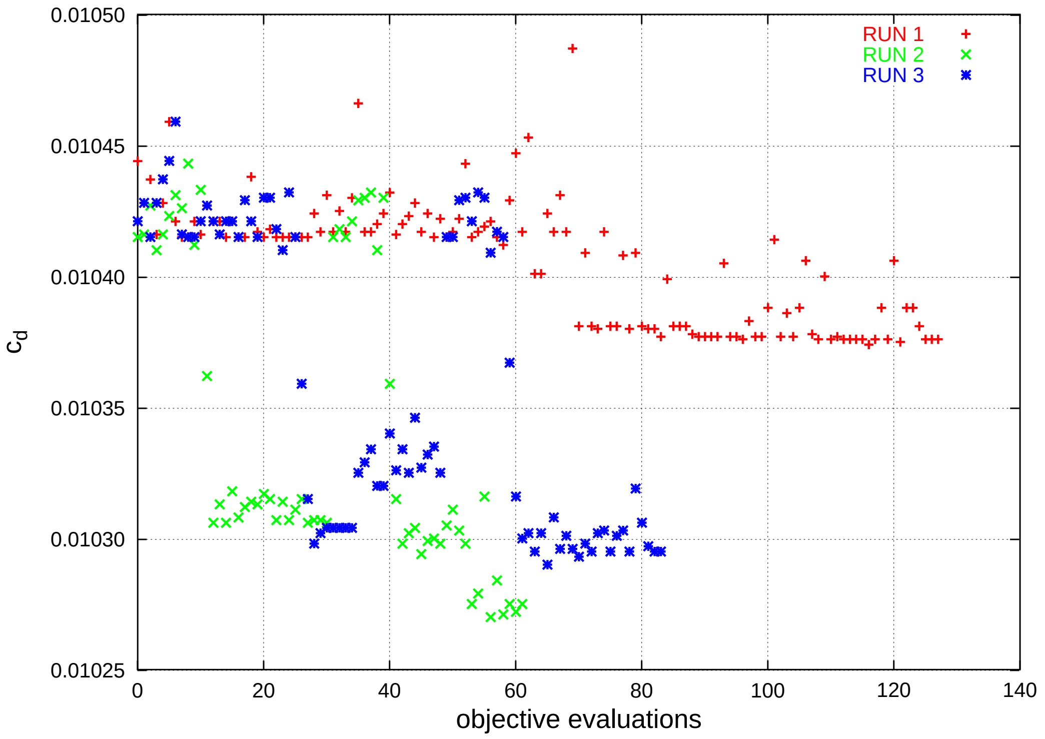


Figure 6: Evolution history of the three runs with eight design variables and Navier-Sokes solver.

6 Use of the adjoint method for gradient computation

A planned improvement of the hybrid genetic algorithm is the introduction of the adjoint method in gradient computation. The major advantage of this approach is that there a higher computational efficiency is obtained. Indeed, by solving the adjoint equations, one obtains all of the gradient components.

A multi-block continuous Navier-Stokes adjoint equation solver for two-dimensional fields was implemented. The numerical solution of the adjoint equations is obtained by using a first-order time-dependent technique based on a finite volume discretization. The solver computes the fluxes at cell interfaces using a flux-vector splitting technique. In a similar way, the boundary conditions are imposed on the numerical fluxes at the computational field edges.

At the moment the adjoint based procedure is being developed as a stand-alone code, whose scheme is reported in figure 10. The gradient computed using the adjoint equation set is then used by a conjugate gradient optimization routine. After the validation phase, the developed solver will be used as core of a new hybrid operator of the genetic algorithm.

The functional to be minimized is:

$$\mathcal{L} = \omega_1 D + \omega_2 \frac{(L - L^*)^2}{2}$$

where D is the drag L is the lift, L^* is the desired lift and the ω_i are weights.

The first case considereds corresponds to the the case of fixed lift without constraints on the geometry. The absence of constraints on shape will lead, as will be evident in the first example reported, to a clear tendency to thickness reduction and to rear loading the airfoil.

Figure 11 reports some results related to the design problem previously defined (RAE2822, $M = 0.68$, $Re = 5.7 \times 10^6$, $c_l = 0.56$) when no control on maximum thickness is imposed. In particular, drag, penalty

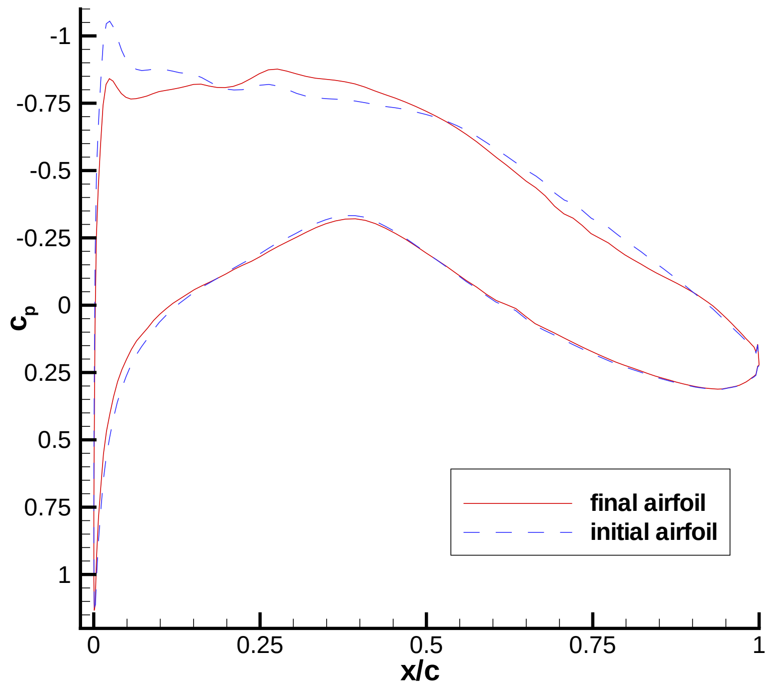


Figure 7: Comparison between initial and final c_p distributions in NS run 3.

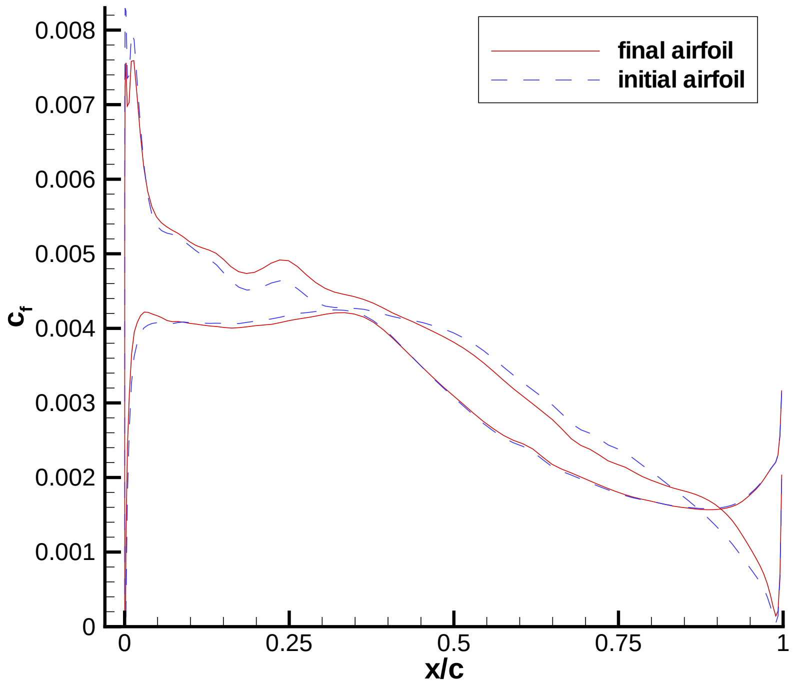


Figure 8: Skin friction coefficient comparison in NS run 3.

on lift, and initial and final airfoil shapes are reported. The gradient is computed point-wise, i.e. each grid point

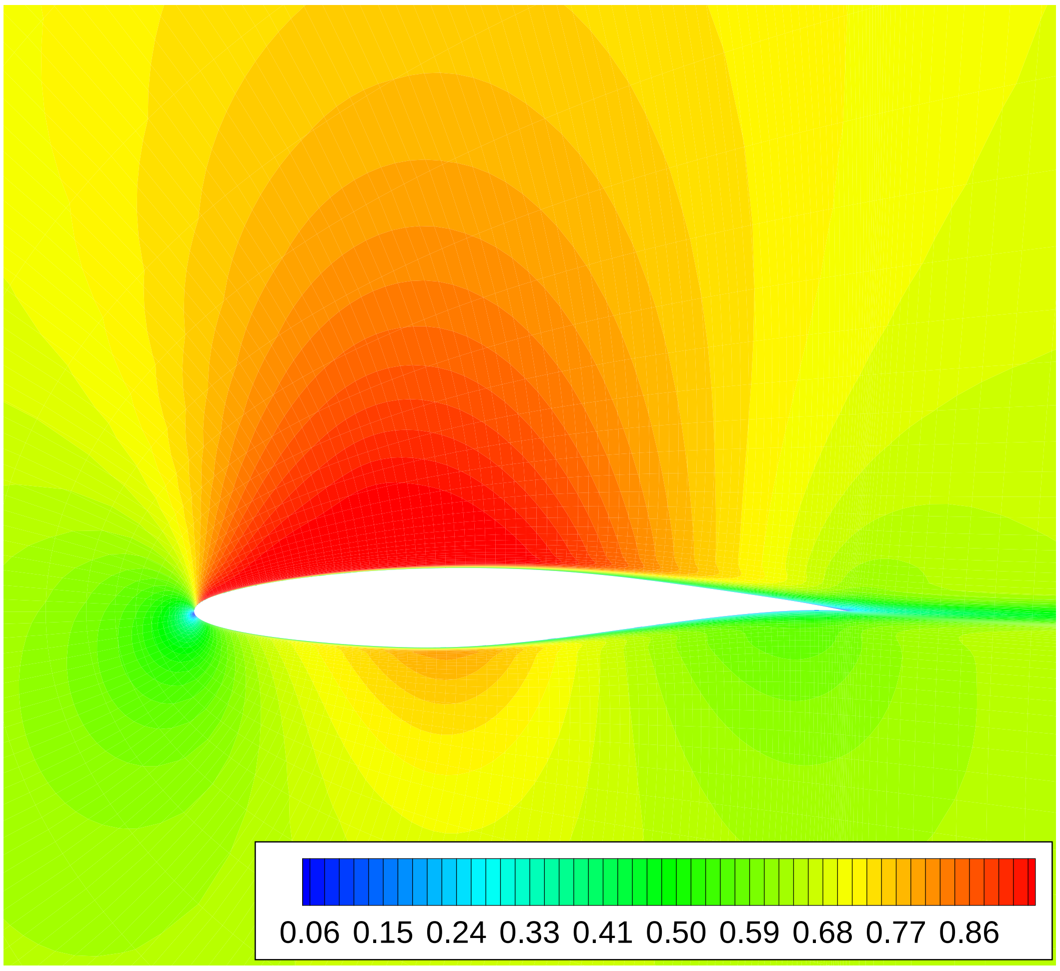


Figure 9: Mach field around the optimized airfoil of run 3.

on the airfoil boundary is a design variable. The gradient analytic expression involves second as well as third derivatives of the flow variables. As the flow field is only second order accurate in space, it can occur that the computation of such derivatives is not reliable in regions of abrupt flow changes, as for example near the front stagnation and close to the rear separation. For this reason the point-wise gradient was smoothed using a Fourier filter. After such filtering the conjugate gradient method is able to nicely decrease the functional $\mathcal{L}$.

Figure 12, instead, reports the result obtained when an a-posteriori control on maximum thickness is imposed. This is obtained projecting, at each iteration, the airfoil modification vector along a direction in which the modification in the maximum thickness section is zero. It is worth to note that while in the first run over 400 field computations were allowed, in this second one only about 160 evaluations were allowed in order to save computational resources.

7 Conclusions

Navier-Stokes flow solvers can be profitably used for aerodynamic shape design with evolutionary optimizers, provided that attention is paid to computational efficiency. In particular, a pre-analysis of the design space with lower fidelity tools is highly recommended to avoid waste of computational resources.

Furthermore, parallel computing and advanced approaches like gradient computation through adjoint, pave the way for challenging industrial applications.

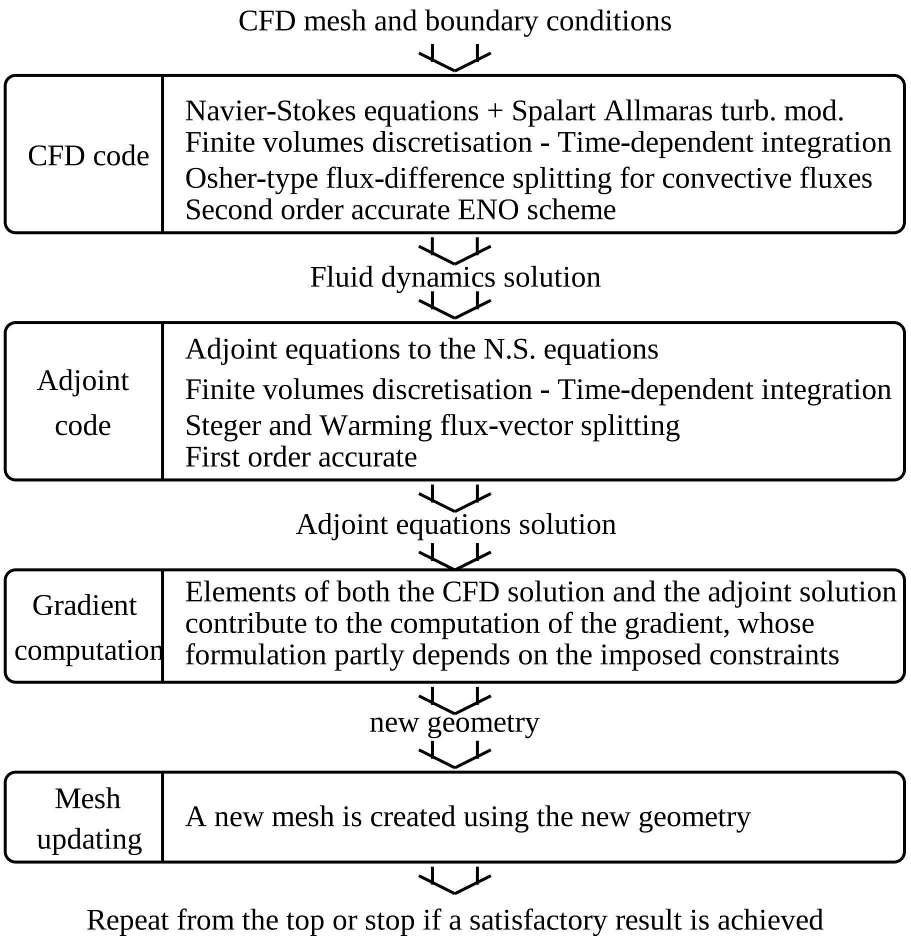

CFD mesh and boundary conditions
CFD code
Navier-Stokes equations + Spalart Allmaras turb. mod.
Finite volumes discretisation - Time-dependent integration
Osher-type flux-difference splitting for convective fluxes
Second order accurate ENO scheme
Fluid dynamics solution
Adjoint code
Adjoint equations to the N.S. equations
Finite volumes discretisation - Time-dependent integration
Steger and Warming flux-vector splitting
First order accurate
Adjoint equations solution
Gradient computation
Elements of both the CFD solution and the adjoint solution contribute to the computation of the gradient, whose formulation partly depends on the imposed constraints
new geometry
Mesh updating
A new mesh is created using the new geometry
Repeat from the top or stop if a satisfactory result is achieved

Figure 10: Adjoint optimizer scheme.

Acknowledgments

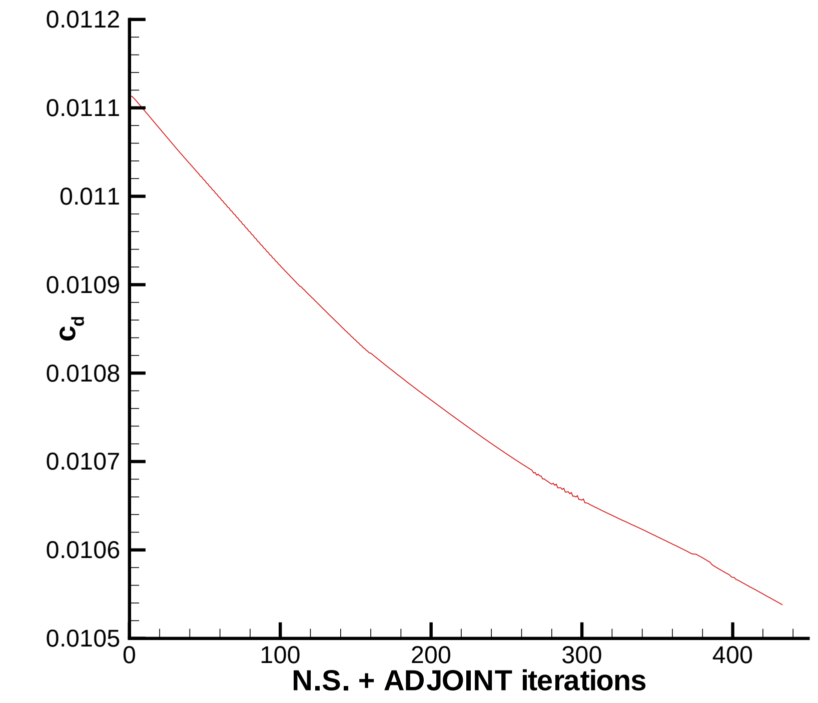
Thanks are due to prof. Mark Drela for allowing the use of his MSES code in our research work, and to dr. Pietro Catalano and dr. Luigi Paparone of the Italian Center for Aerospace Research (CIRA) for their precious help on the use of ZEN flow solver.

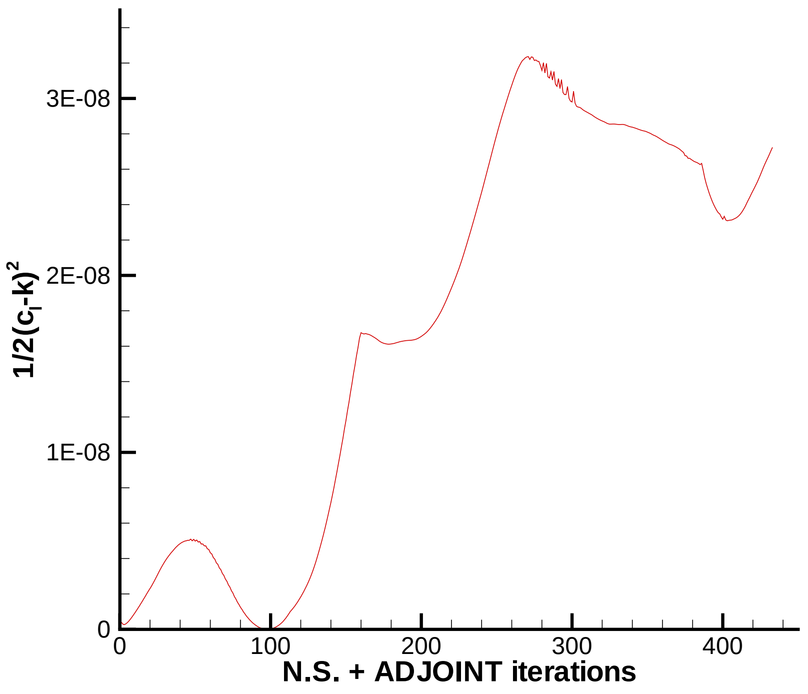
Part of the work here presented was carried out within the framework of the AEROSHAPE project. This project (Multi-Point Aerodynamic Shape Optimization) is a collaboration between Aerospatiale Matra Airbus, Alenia Aeronautica (Coordinator), Daimler Crysler Airbus, EADS-M, Dassault Aviation, SAAB, SENER, SYNAPS, CIRA, DERA, DLR, FFA, INRIA, HCSA, NLR, ONERA, and NuTech Solutions. The project is funded by the European Commission, DG Research, under the GROWTH initiative (Project Ref.: GRD1-1999-10752).

References

[1] Shigheru Obayashi. Pareto genetic algorithm for aerodynamic design using the navier-stokes equations. In Domenico Quagliarella, Jacques Périaux, Carlo Poloni, and Gabriel Winter, editors, *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science*, pages 245–266, England, November 1997. John Wiley & Sons Ltd.

[2] N. Marco, S. Lanteri, J. A. Desideri, and Jacques Periaux. A parallel genetic algorithm for multi-objective optimization in computational fluid dynamics. In Kaisa Miettinen, Marko M. Makela, Pekka Neittaanmaki, and Jacques Periaux, editors, *Evolutionary Algorithms in Engineering and Computer Science*, pages 445–455, England, 1999. John Wiley & Sons Ltd.





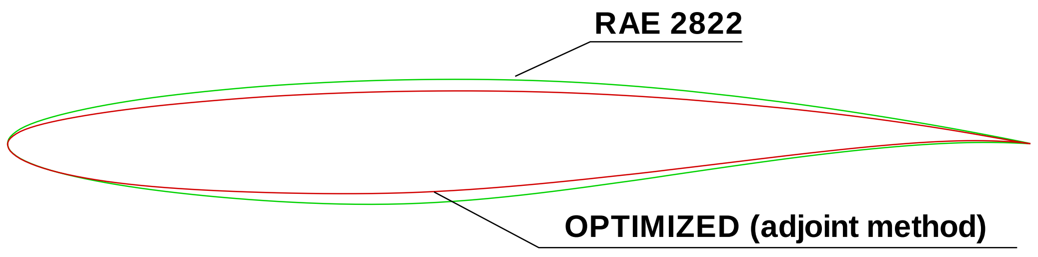
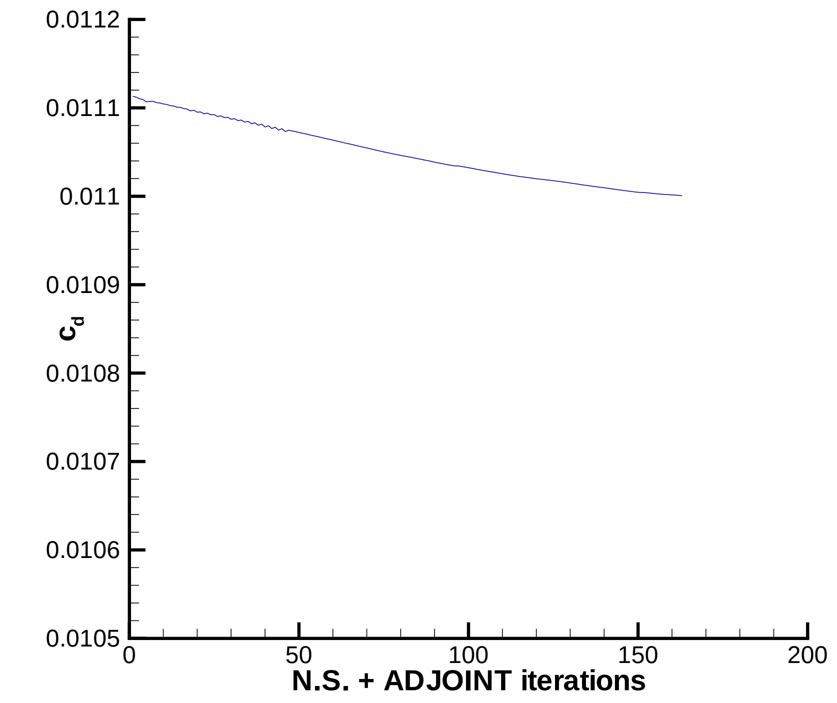
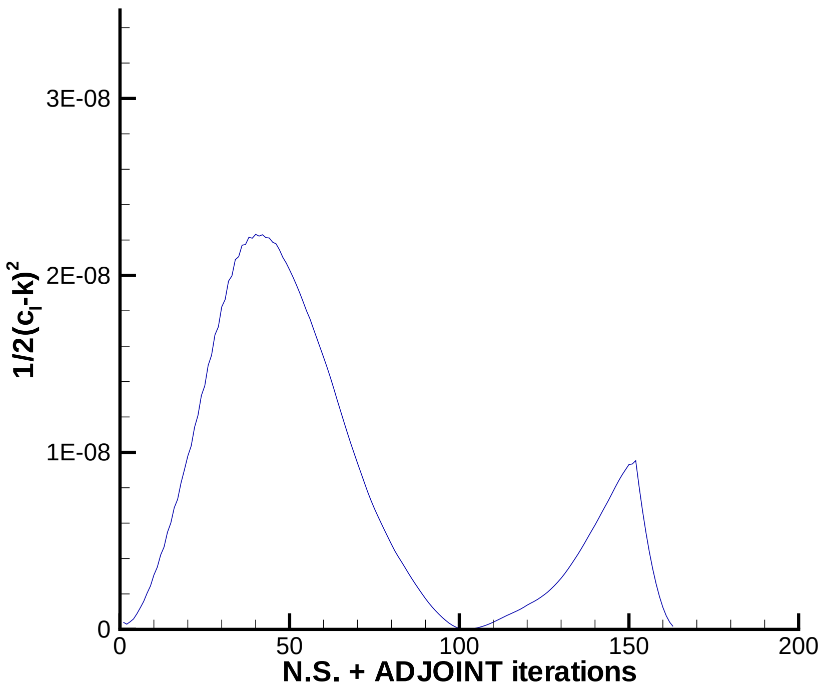


Figure 11: Drag, lift penalty constraint for the functional $\mathcal{L}$, and airfoil shapes comparison for the case without constraint on maximum thickness.





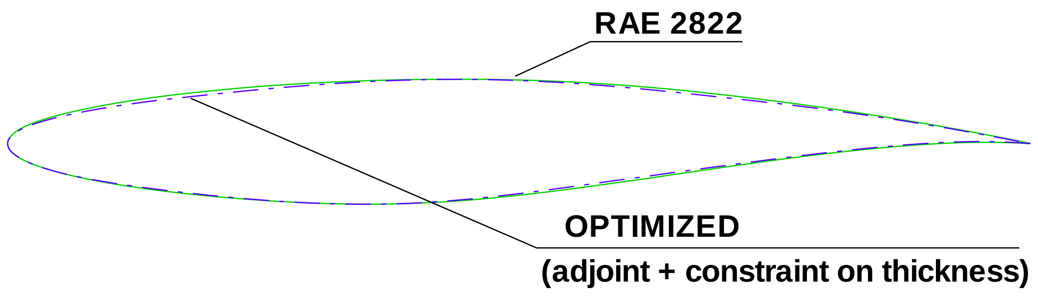


Figure 12: Drag, lift penalty constraint for the functional $\mathcal{L}$, and airfoil shapes comparison for the case with constrained maximum thickness.

[3] A. P. Giotis, K. C. Giannakoglou, and Jacques Périaux. A reduced-cost multi-objective optimization method based on the pareto front technique, neural networks and pvm. In *Proceedings of ECCOMAS 2000 Conference*, Barcelona, Spain, September 11–14 2000.

[4] Carlo Poloni and Valentino Pediroda. Ga coupled with computationally expensive simulation: Tools to improve efficiency. In Domenico Quagliarella, Jacques Périaux, Carlo Poloni, and Gabriel Winter, editors, *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science*, pages 289–309, England, November 1997. John Wiley & Sons Ltd.

[5] Domenico Quagliarella and Alessandro Vicini. Coupling genetic algorithms and gradient based optimization techniques. In Domenico Quagliarella, Jacques Périaux, Carlo Poloni, and Gabriel Winter, editors, *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science*, pages 289–309, England, November 1997. John Wiley & Sons Ltd.

[6] L. Davis. *Handbook of genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

[7] G. N. Vanderplaats. *Numerical Optimization Techniques for Engineering Design: with Applications*. Mc Graw–Hill, 1984.

[8] Alessandro Vicini and Domenico Quagliarella. Inverse and direct airfoil design using a multiobjective genetic algorithm. *AIAA Journal*, 35(9):1499–1505, September 1997.

[9] A. Barak and O. La'adan. The mosix multicomputer operating system for high performance cluster computing. *Journal of Future Generation Computer Systems*, 13:361–372, March 1998.

[10] J. C. Kok, M. Amato, and I. W. Boerstoel. Mathematical-Physical modeling for multi-block NaS/Euler simulation. Technical report, Centro Italiano Ricerche Aerospaziali, Capua, Italy. CIRA-DLCEST-TR183, and NLR-CR91-235L.

[11] B. S. Baldwin and H. Lomax. Thin layer approximation and algebraic model for separated turbulent flows. American Institute of Aeronautics and Astronautics (AIAA), 1978. AIAA Paper 78-257.

[12] Mark Drela. Newton solution of coupled viscous/inviscid multielement airfoil flows. In *AIAA, Fluid Dynamics, Plasma Dynamics and Lasers Conference*, Seattle, WA, June 1990. American Institute of Aeronautics and Astronautics (AIAA). AIAA Paper 90-1470.

This page has been deliberately left blank

Page intentionnellement blanche

Conceptual Missile Design Using Genetic Algorithms

Murray B. Anderson
Sverdrup Technology Inc./TEAS Group
308 West D. Avenue
Building 260
Eglin Air Force Base, FL 32542, USA

Executive Summary

This paper was written to show the potential of using an intelligent systems tool to rapidly conduct conceptual missile design studies. Rather than having separate teams of engineers developing/proposing discipline-specific designs (i.e. aerodynamics, propulsion, flight control system), a computer can quickly learn how to design a missile as an overall "system" to effectively defeat difficult targets. This paper was an invited paper presented at a joint NATO/Von-Karmen-Institute Workshop on Intelligent System held in Brussels, Belgium during May of 2002.

Abstract

For these preliminary design studies, a pareto genetic algorithm was used to manipulate a solid rocket design code, an aerodynamic design code, and a three-loop autopilot to produce interceptor designs capable of defeating two different target scenarios. The first scenario is a ground-launched interceptor engaging a high-speed/high-altitude target like a re-entry vehicle. The second scenario is an air-to-air engagement against a highly maneuverable target. Twenty-nine design variables were required to define the optimization problem, and two primary goals were established to access the performance of the interceptor designs. Design goals included minimizing both miss distance and intercept time. The genetic algorithm was able to quickly develop several basic types of designs that performed very well for the individual target scenarios.

Introduction

With the addition of guidance, an autopilot and an airframe with movable control surfaces, basic rocketry, expands into a more lethal and much more precise means of waging war. Rather than increasing the size of the warhead being delivered (to make-up for a loss in delivery accuracy), modern weaponeering has tended to use a small warhead coupled with an accurate control system. For ground launched systems, ideas such as "smart rocks" and "brilliant pebbles" that sprang from early Strategic Defense Initiative (SDI) research were based on the belief that the energy delivered by a small, fast moving projectile without an explosive could be as lethal as a less accurate system with an explosive warhead. Similarly for air-launched ground attack weapons, Vietnam proved that delivery accuracy was paramount to defeating tough targets. The Air Force subsequently devoted billions of dollars to the development of precise warhead delivery systems. The Persian Gulf War showed the benefit of these highly accurate weapon systems that were developed. While it is difficult to argue with the success of modern weapons, these systems share a common design philosophy that is somewhat lacking. When a system has a guidance system and autopilot, there is a tendency to compensate for less than stellar aerodynamic designs by shifting more and more of the delivery problems over to the autopilot. As a result, autopilots are typically very good and very robust, but the airframe and aerodynamics of the overall system are almost an afterthought. Overall system performance and system capability, therefore, suffers because of the over-reliance on the autopilot to compensate for weaknesses in the aerodynamic performance of the weapon system. The goal of this research is to let an artificial intelligence tool, a genetic algorithm, design the aerodynamic shape while, at the same time, designing the propulsion system and determining key autopilot variables. This all-at-once approach to missile design is intended to provide a system capable of producing good aerodynamic shapes in addition to the good performance expected from an autopilot.

Following a discussion of the system components being manipulated by the genetic algorithm in this research, the operating modes of the genetic algorithm are discussed. Lastly, the results of the rapid design studies for the two target scenarios are presented.

Guidance Algorithm

Guidance is the first key component to accurately commanding a missile to a target. The guidance algorithm used in this study is standard proportional navigation (called ProNav). A ProNav system commands accelerations normal to line of sight between the missile and the target, proportional to the closing velocity (V_c) and the line of site rate ($\dot{\lambda}$). In equation form, this relationship is

$$\eta_c = N^{'} V_c \dot{\lambda} \qquad \text{Equation 1}$$

where N' is the effective navigation ratio or gain. The closing velocity and line of sight rate are typically determined by a Doppler radar and seeker, respectively. For this research, it is assumed that there is a perfect seeker and a perfect radar system so that the target position and velocity are known exactly. For preliminary design studies, these two assumptions are appropriate. However, to make the autopilot performance variables (like damping ratio) more meaningful, the target position/velocity model was run at a slightly slower time step than the autopilot itself.

Before discussing the impact of the effective navigation ratio on the guidance system, it is appropriate to discuss how the commanded accelerations make their way into the missile control system. As a two-dimensional example, consider the yaw guidance plane shown in Figure 1. As this figure shows, from an inertial coordinate system viewpoint, the acceleration commands generated by ProNav are in the Earth-centered coordinate system (X,Y,Z) rotated through the line-of-sight angle. The commanded accelerations in the inertial axis system are then

$$\begin{aligned} a_X &= \eta_c \cos(\lambda) \\ a_Y &= -\eta_c \sin(\lambda) \end{aligned}$$

These accelerations, and the ones from the corresponding X-Z plane, must be transformed into the missile body axis system before they can be used to move control surface to point the missile toward the target.

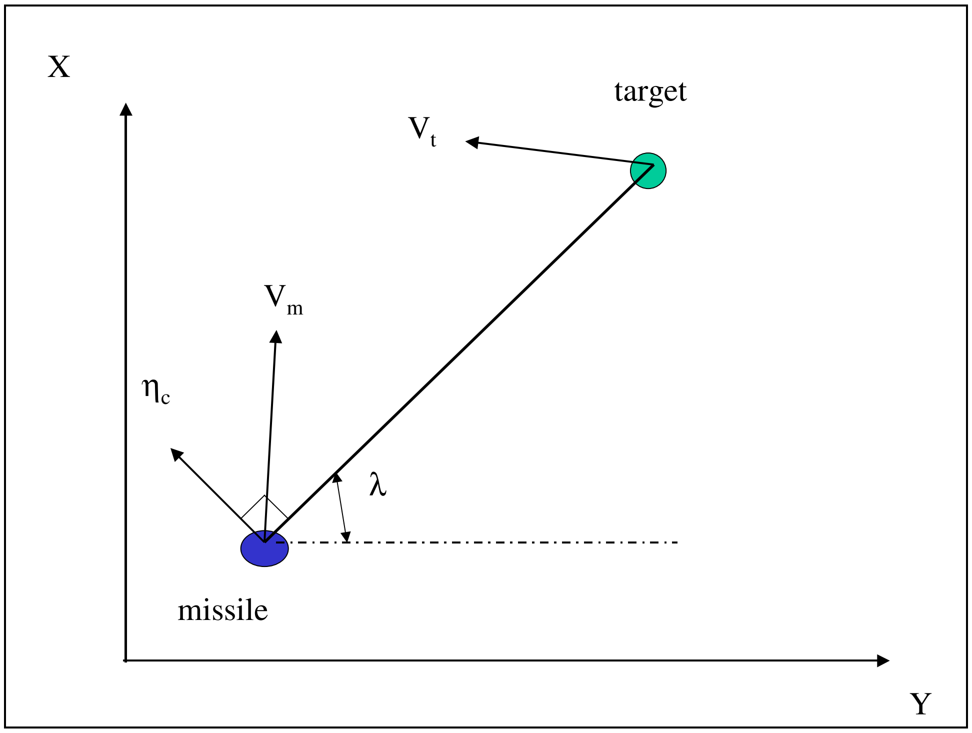


Figure 1. Guidance Command Schematic

The effective navigation ratio controls the behavior of the autopilot throughout the flight of the missile. Based on equation 1, it is obvious that, for a fixed closing velocity and fixed line-of-sight rate, higher effective navigation ratios mean higher commanded accelerations. Since it is the goal of the ProNav guidance law to eliminate the line-of-sight rate, higher effective navigation ratios translate into more active maneuvers early in flight in order to minimize the maneuvers near the end-game. For non-maneuvering targets, there is an obvious advantage to using a high effective navigation ratio. Typical ranges for this parameter are 3 to 5 (unitless) according to Zarchan[1] for tactical weapon systems. So, given this variation, the effective navigation ratio is a variable the genetic algorithm should determine.

As the missile enters the endgame maneuvers near the target, the line-of-sight rate will approach infinity as the missile passes (or passes through) the target and, therefore, the commanded accelerations will also approach infinity. It is common to limit the total acceleration commands (circular total acceleration commands) for flight systems and, for this study, the total acceleration was limited to 90 G's. Lateral accelerations greater than 90 G's would likely cause failure of the missile electronic components, if not failure of the structure itself.

Autopilot

The autopilot chosen for this study is the so-called three-loop pitch/yaw autopilot. This autopilot design was chosen because of its simplicity, because it is actively being used in several existing weapon systems, and because it is possible to analytically calculate the proper system gains (for all flight conditions) based on a few specified autopilot performance parameters. Since only the pitch and yaw channels are being modeled, the airframe is free to roll, and does so through induced roll rates as the missile pitches and yaws during intercept. This approach does reduce the turning efficiency of the missile but, as long as the roll angles are tracked accurately, the autopilot is able to compensate. Often in preliminary design studies[2] the roll autopilot is modeled as a perfect stabilizer and is, therefore, neglected entirely. In this current research, inertial acceleration commands from guidance are simply rotated to the missile body axis system and the tail control surfaces respond with appropriate combined pitch/yaw commanded deflections to steer the missile to the target, regardless of the roll angle. The three loop autopilot (see Figure 2 for a schematic of one channel) takes acceleration commands from guidance and outputs the elevator/rudder commanded deflection angles that should be required in order to achieve the commanded accelerations. These tail control surface deflections produce a response through the airframe as a pitch/yaw rate and as a longitudinal/lateral acceleration. The rates and accelerations are fed back through the autopilot to dampen the system. For this study, it was assumed that the Inertial Measurement Unit (IMU)was perfect, meaning that the actual and measured accelerations and rates are identical. For simplicity, it is also assumed that the "perfectly measured" accelerations include appropriate translations to compensate for center-of-gravity movement as fuel is expended. In real systems, there will be lag and measurement error in the IMU, but for this preliminary design study, these simplifications are appropriate. The actuator was modeled as a 2nd order system[1] with a damping of 0.7 and a natural frequency of 125 rad/sec. These simplifications mean that the system is 5th order if the airframe is considered to be 2nd order response.

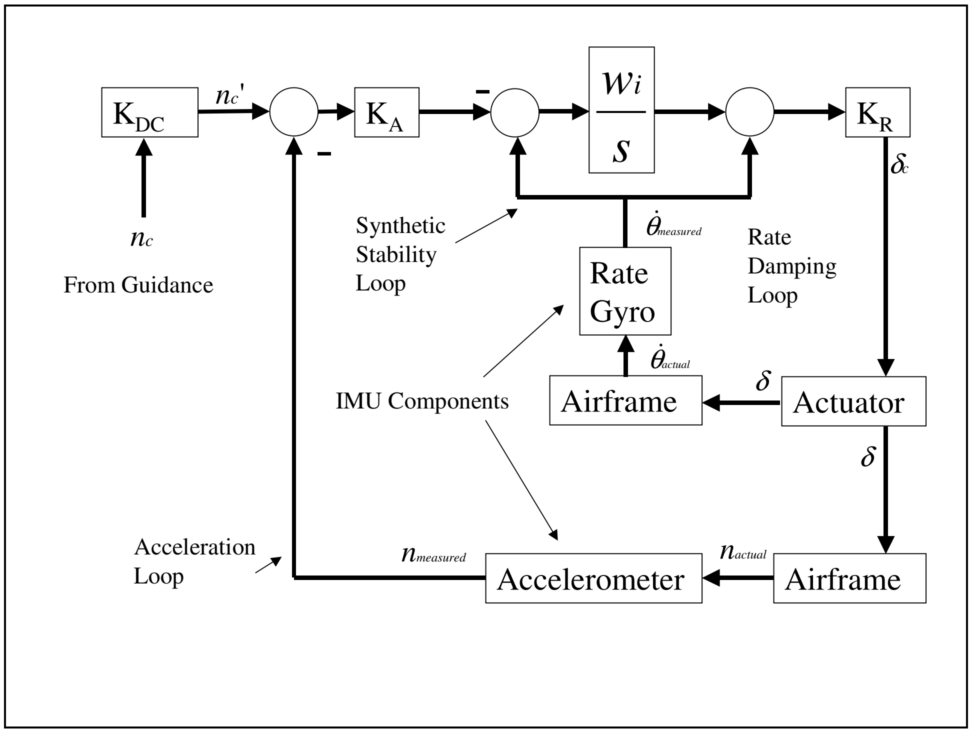


Figure 2. Basic Three-Loop Autopilot

As Figure 2 shows, there are plenty of feedback opportunities in the system to help maintain a robust controllable system for very diverse missile aerodynamic characteristics. The autopilot should be designed to provide a fast, well-damped response to acceleration commands. This task may seem easy at first, but consider what happens from the time a missile is launched to the time all the fuel is burned. It is very likely that the system center of gravity will have moved forward (stabilizing movement toward the nose), changing the airframe from a statically unstable system (center of pressure forward of the center of gravity) to a stable system. For ground-launched missiles, the velocity of the missile may also have changed from 0 ft/sec to over 4,000 ft/sec and the altitude may have changed from sea level to over 100,000 feet. Throughout all these changes, the autopilot must adapt or the missile will be uncontrollable.

The three loops shown in Figure 2 serve specific purposes. The rate damping and synthetic stability loops provide proportional plus integral compensation, effectively damping the airframe poles[3]. The synthetic stability loop was originally added to compensate for statically unstable airframes. The acceleration loop controls the lateral acceleration of the missile. The autopilot gains are K_{DC}, K_A, ω_i, and K_R. All that needs to be done is to calculate the appropriate values for these gains. One of the reasons for choosing the three-loop autopilot is that these gains can be determined analytically from the aerodynamics of the airframe and from some specified performance parameters for the autopilot. These performance parameters are the damping ratio (ζ), time constant (τ), and crossover frequency (ω_{cr}). The system-damping ratio governs the sensitivity of the system (i.e. small heading errors should not produce large elevator/rudder deflections) and limits overshooting the commanded acceleration to help protect the structural integrity of the system and the electronics. The time constant is a measure of how fast the system responds to acceleration commands. For example, a 0.3 second time constant means that 63 percent of the commanded acceleration should be achieved in the first 0.3 seconds. The crossover frequency (when gain falls below 0.0Db) is a measure of autopilot robustness when higher order dynamics are not modeled. Since the system analyzed here is a 5th order system, the crossover frequency essentially determines how fast the autopilot responds during homing. Higher crossover frequency values mean a faster autopilot but, if the autopilot is too fast, it can cause instability in the actuator. A good rule of thumb[1] is that the crossover frequency should not exceed 1/3 of the bandwidth of the actuator. In this study, the actuator was assumed to have a natural frequency of 125 rad/sec and a damping of 0.7, so the highest crossover frequency that can be safely used should be around 41.66 rad/sec to maintain stability. This particular instability is the reason why, in preliminary design studies, determining a good estimate of the crossover frequency can only really be done with a 5th order system or higher. Evaluation of 3rd order systems (perfect actuator) is usually a first step to analyzing a 5th order system (actuator included), followed by an 11th order system that includes actuator dynamics, IMU response, and structural dynamics[2].

The equations that determine the four autopilot gains (at a single flight condition) from the aerodynamics and autopilot performance parameters are omitted in this discussion. These equations can readily be obtained from multiple sources. , However, it should be noted that Nesline[2] provides general equations that work well even for very unstable missiles, while Zarchan[1] does not. The basic premise of these equations is that the aerodynamic derivatives (like $C_{m\alpha}$ and $C_{m\delta}$) and expected aerodynamic damping provide a means of determining the linear response that a system is capable of delivering at a given dynamic pressure. If the system response can be estimated, appropriate gain schedules can be developed to achieve the desired autopilot performance levels.

It should also be noted that, for the ground launch case, the autopilot is not activated until 0.4 seconds into the flight to give the system time to generate enough speed to make fin commands meaningful.

Aerodynamics Code

Washington[4] developed AeroDesign, the aerodynamic prediction methodology used as the basis for this study. AeroDesign was modified to include two axial force considerations that were not part of the original software. First, the fineness ratio of the nose of the missile is compared to a Sears-Haack[5] body and, if the nose is not slender enough, a drag penalty proportional to the nose bluntness is added to the baseline axial force coefficient. The second axial force coefficient correction was implemented to correct for cases where the rocket nozzle exit diameter actually exceeds the diameter of the body.

AeroDesign was further modified to provide aerodynamic damping derivative estimates and linear aerodynamic coefficient contributions for deflected control surfaces in the pitch and yaw planes in a format and flight condition range compatible with a guided six-degree-of-freedom simulation.

Rocket Performance Code

The solid rocket performance software used in this study is an erosive burning star grain design program that is suitable for preliminary design studies. There are some fundamental assumptions made in the formulation of the software that are suitable for rapid evaluation of preliminary design. The major assumptions are:

1. The pressure varies throughout the chamber, however, the pressure is calculated only at the head end (P_1) and at the grain end (P_2). The chamber pressure (P_{CH}) is then defined as the average of these two pressures.
2. The burn rate of the propellant also varies over the entire surface and is subject to erosive burning. The burning rate at any point can be defined as $r = aP_{CH}{}^{n}\left(1 + k \cdot V\right)$, where k, a, and n are burning rate constants. Since the chamber pressure is calculated only at the head end where V_1 equals 0, and at the aft end of the grain, where V equals V_2, the velocity is averaged to calculate an average erosive burning rate.
3. The grain burns on the edges normal to the centerline of the rocket only (i.e. no end-burning so x_{gl} is constant) at the average burning rate.
4. The flow is isentropic between the aft end of the grain and the throat.
5. The flow obeys the perfect gas law.
6. The chamber pressure varies with time, but is essentially constant during the discharge of a single particle.
7. The flow is one-dimensional and steady.
8. There is no deformation of the propellant due to acceleration, pressure, or viscous forces.
9. The temperature is uniform throughout the grain, but the grain is temperature sensitive (this is a fuel characteristic).

The rocket motor to be designed by the genetic algorithm has certain definable characteristics such as the strength of the combustion chamber material, which should be known before the design process begins. For this study, the following rocket characteristics were used:

1. Propellant is ammonium perchlorate (80%).
2. Initial temperature of the propellant is 20°C.
3. The design chamber pressure (i.e. maximum chamber pressure for case structural design) is 3000 psi. Chamber case thickness is determined by using a factor of safety of 1.5 given this maximum chamber pressure.
4. The allowable stress in the case is 195,000 psi.
5. The factor of safety is 1.5.
6. The case is made of a steel alloy with a density of 0.28 lbm/in^3.

7. The nozzle is made of an aluminum alloy with a density of 0.19 lbm/in^3.

For this study, ammonium perchlorate (80%) was chosen but there is no reason why the propellant choice could not be another design parameter if there is a database of propellant burning characteristics available. For ammonium perchlorate, the erosive burning rate constant k is 1.0E-4 sec/ft, and the burning rate constants a and n are 0.15 and 0.4, respectively.

Six-Degree-of-Freedom (6-DOF) Simulation

The equations of motion used in this 6-DOF were obtained from Etkin[6] assuming that (1) the missile structure is essentially rigid, (2) there are no rotating masses internal or external to the missile, and (3) there are no significant cross products of inertia. Using Etkin's notation, longitude is measured positive from west to east, which is opposite from conventional world maps. The simulation was modifed so that even though Etkin's coordinate system is used exactly, the output of the simulation is converted to standard world map definitions to avoid confusion. Also consistent with Etkin, the standard Eulerian definitions of Φ, Θ, and Ψ were used. To avoid gimbal lock (θ=90 deg) concerns, quaternions were used instead of the standard Euler angles during the integration process. However, all output was converted from quaternions to Euler angles for output.

The conventional body-fixed acceleration and moment equations were modified to include aerodynamic damping terms and contributions due to control surface deflections. For this study, a tail-control missile was assumed but there is no reason why a canard-control system could not also be analyzed. The definitions used for the control surface contributions to the aerodynamic coefficients were as follows:

1. positive elevator commands produce positive normal force contributions and negative pitching moment contributions
2. positive rudder commands produce positive side force contributions and negative pitching moment contributions

For a 5th order system, 19 differential equations must be solved simultaneously (15 for the quaternion system of equations and 4 for the elevator and rudder).

Link to GA: GA with Aerodynamics, Propulsion, Six-DOF, Guidance, Autopilot, and Target

Linking the separate software codes to the genetic algorithm was done in a modular fashion so that other modules could be later substituted for the ones used in this study. The genetic algorithm passes all design variables down to the Six-DOF via one subroutine call statement (single line of interface). The Six-DOF then calls the other components, including the mass properties routine that calculates the component inertias and the center of gravity for the system. For this study, the payload was assumed to weigh 50 lbf and the electronic components/actuators weighed an additional 50 lbf.

Variables Governing Design

There are nine variables that govern the solid rocket motor design, fourteen variables that govern the external shape of the vehicle, two variables that control the launch angle (verticality and heading), three variables that define the autopilot performance, and one variable to set the effective navigation ratio or gain. Figure 3 shows the external geometry variables, with the exception that the nozzle exit radius is actually determined from the expansion ratio (A_e/A^*), which is one of the solid rocket motor design variables. Though the nozzle is shown, it is merely for visualization purposes. The nozzle actually resides within the total length of the missile. The nozzle exit radius is not, however, free from external aerodynamic considerations since there is a substantial drag penalty that can be incurred if the nozzle exit radius exceeds the body radius. Hopefully, the genetic algorithm will learn to design the rocket motor and external shape cooperatively so that good thrust levels are obtained without incurring a drag penalty. Basically then all outer body dimensions are controlled by the genetic algorithm, from the nose length to the nozzle exit radius. Figure 4 and Figure 5 show the nine solid rocket motor design variables that are also part of the design process, with (A_e/A^*) being one of those variables.

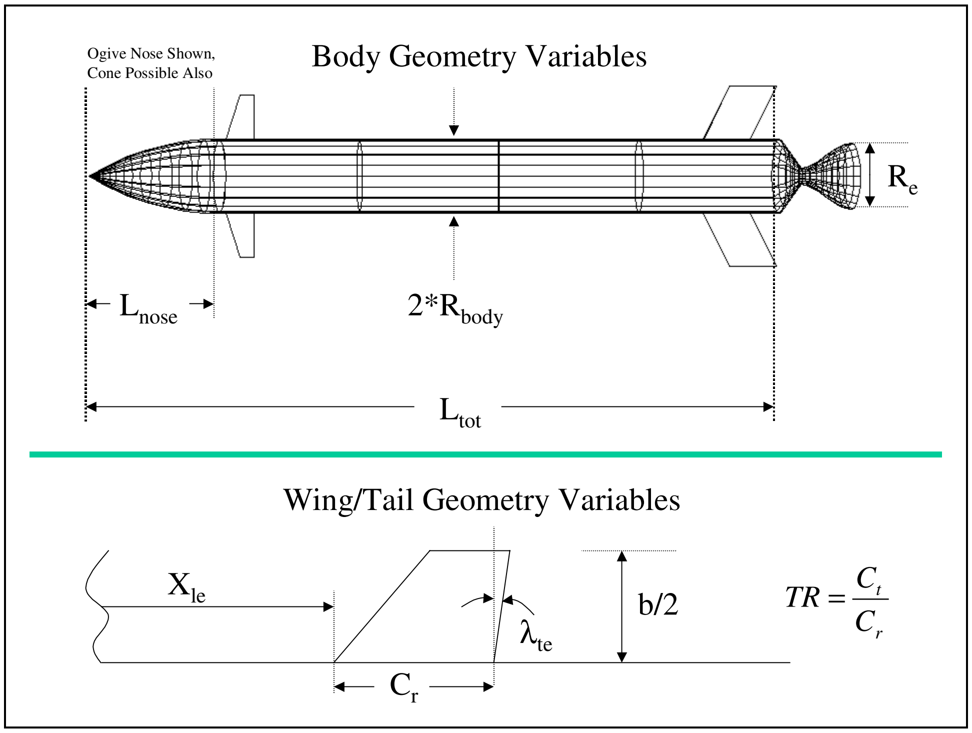


Figure 3. External Shape Design Variables

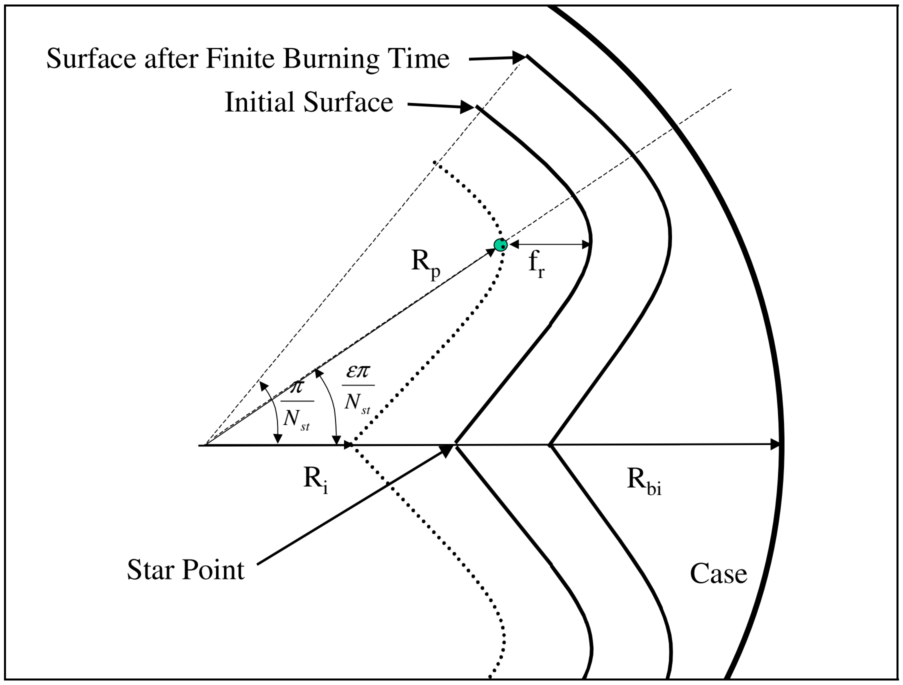


Figure 4. Solid Rocket Motor Grain Design Variables

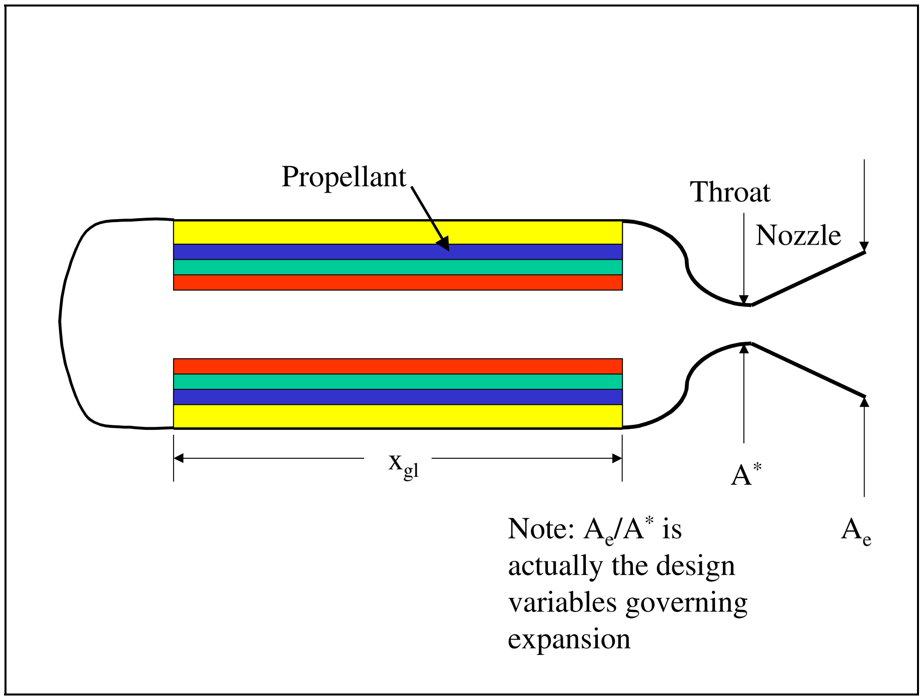

Propellant
Throat
Nozzle
x_{gl}
A^*
A_e
Note: A_e/A^* is actually the design variables governing expansion

Figure 5. Solid Rocket Motor Grain Length, Throat Diameter, and Expansion Ratio

Table 1 formally defines each design variable and Table 2 shows the minimum, maximum, and resolution that is desired for each variable. The maximum, minimum, and resolution dictate the size of the optimization space. In genetic algorithm terms, the number of genes in each chromosome is defined as:

$$number\ of\ genes = \sum_{n=1}^{\substack{number\\ of\\ parameters}} \left(INTEGER\left[\frac{LN\left(\frac{\max_n - \min_n}{resolution_n} \right)}{LN(2)} \right] + 1 \right)$$

The genetic algorithm requires parameter bounds and resolutions only and, from this table, it is obvious that a very broad range of designs is possible. In fact, the specified bounds and resolutions means that 2^{175} possible designs exist. The size of this problem is tremendous, especially when the number of atoms in the universe is estimated at 2^{266}.

Table 1. Design Variables for Guided Systems

Variable Name	Definition (units)
R_{bi}	Grain outer radius, also the rocket motor case inner radius (inches)
R_p	Outer star radius (inches)
R_i	Inner star radius (inches)
x_{gl}	Grain Length (inches)
N_{st}	Number of star points
f_r	Fillet radius (inches)
ε	Angular fraction (rad)
D^*	Diameter of the throat (inches)
R_{exp}	Nozzle expansion ratio (area of throat/area of exit)
Nose	1 – Ogive, 2 – Cone
L_{nose}	Nose Length (inches)
L_{tot}	Total Body Length excluding nozzle (inches)
R_{body}	Body radius (inches)
b_w	Exposed semi-span of wing (inches)
C_{rw}	Wing Root chord (inches)
λ_{tew}	Wing Trailing Edge Sweep Angle (deg)
TR_w	Wing Taper ratio (C_t/C_r)
X_{lew}	Distance from nose tip to wing leading edge (inches)
b_t	Exposed semi-span of tail (inches)
C_{rt}	Tail Root chord (inches)
λ_{tet}	Tail Trailing Edge Sweep Angle (deg)
TR_t	Tail Taper ratio (C_t/C_r)
X_{let}	Distance from nose tip to tail leading edge (inches)
θ	Euler Vertical Launch Angle (θ=90 would be vertical (deg))
Ψ	Euler Launch Heading Angle (Ψ=90 would be East (deg), 0 would be North)
ζ	Damping Ratio
τ	Time Constant
ω_{cr}	Crossover Frequency
N'	Effective Navigation Ratio or Gain

Table 2. Maximum, Minimum, and Resolution of Variables for Guided Systems

Parameter	Minimum	Maximum	Resolution	Number of Genes
R_{bi}	2.0	10.0	0.02	9
R_p	0.2	9.9	0.02	9
R_i	0.1	9.5	0.02	9
x_{gl}	50.0	200.0	1.0	8
N_{st}	3	10	1	3
f_r	0.05	1.0	0.01	7
ε	0.25	1.0	0.01	7
D^*	0.1	9.0	0.01	10
R_{exp}	1.0	20.0	0.2	7
Nose	0	1	1	1
L_{nose}	20.0	90.0	5.0	4
L_{tot}	50.0	450.0	10.0	6
R_{body}	3.0	20.0	1.0	5
b_w	0.0	80.0	1.0	7
C_{rw}	0.0	80.0	1.0	7
TR_w	0.0	1.0	0.1	4
λ_{tew}	0.0	44.0	2.0	5
X_{lew}	20.0	200.0	5.0	6
b_t	0.0	80.0	1.0	7
C_{rt}	0.0	80.0	1.0	7
TR_t	0.0	1.0	0.1	4
λ_{tet}	0.0	44.0	1.0	5
X_{let}	200.0	400.0	5.0	6
θ	5.0	90.0	1.0	7
ψ	0.0	180.0	1.0	8
ζ	0.2	1.0	0.05	4
τ	0.1	0.9	0.1	3
ω_{cr}	10.0	100.0	5.0	5
N'	2.0	5.0	0.1	5

Mode of Operation of Genetic Algorithm

The controls for the genetic algorithm used in this study were fairly common. The number of population members was limited to 150 because of computer run time considerations. Generally, the more population members the better, but when running a 6-DOF thousands of times it was necessary to limit the population size to less than the number of genes. Historically, pareto GA's have been run with more population members than genes to help maintain good diversity in the presence of competing goals.

The following table lists all the genetic algorithm operating modes/parameters that were used for this study.

Table 3. Genetic Algorithm Controls for Guided Missile Cases

Mode/Variable	Value
Number of Goals	2
Steady State Algorithm	False
Elitist	True
Creep Mutation	True
Remove Duplicates	True
Number of Members of the Population	150
Crossover Rate	90%
Mutation Rate	0.2%
Creep Rate	2%
Selection Process	Tournament
Crossover	Single Point

Design Conflict Checking

Some obvious geometrical checks were used to keep the genetic algorithm from expending computational resources on designs that were not practical. Seven separate checks were made as follows:

1. Outer rocket motor case radius cannot exceed body radius
2. Rocket motor grain length cannot exceed body length
3. Tail control surfaces cannot be coincident with, or in front of, wing
4. Tail control surfaces cannot overhand the aft end of the missile
5. Wing cannot overhand nose portion
6. Based on the specified payload and electronic weights and densities, as well as the rocket motor size, the total volume of the missile must be able to house these components
7. Tail control surfaces must be located such that the actuator hinge line (assumed to be at the 50% location of the tail root chord) can be placed at, or very near, the rocket motor throat. Since the actuators take up a considerable volume, it is logical that they would be placed near the throat of the nozzle.

If any of these conflicts occur, the genetic algorithm sends back extremely poor performance values in each goal area, so that it will learn not to try these designs in the future.

Thermal and Structural Considerations

The 6-DOF software calculates stagnation temperature at each time step based on Mach number and altitude (standard atmosphere is assumed). If the stagnation temperature ever exceeds 2500 degrees Rankine, then the system is assumed to fail because of thermal loads.

The structural considerations are manifested in the strength of the wings and tails since these are obvious weak points. Wing and tail loads during flight are used to calculate root bending moments and bending stresses. If the bending stresses ever exceed 185,000 psi (typical for stainless steel), the wing or tail surfaces fail and the flight is terminated at that point. The wing joints are assumed to be rigidly connected to the missile body along the entire root chord. Each actuator-controlled tail surface is assumed to be mounted on a 1.25-inch diameter stainless steel rod. The genetic algorithm must learn to design systems that will not fail either thermally or structurally.

The Target

The two targets specified for this research are vastly different. The first is a fast point-mass ground-attack re-entry vehicle like a SCUD missile. The second is a highly maneuverable air-to-air target capable of random 5-15G's maneuvers every 0.1 seconds. The target subroutine defines initial positions, velocities, and accelerations for the target and performs simple Euler integration on the equations of motion to update the target's position and velocity in time. For the random maneuvering target scenario, each missile design was flown against 10 different random targets and the miss distances and intercept times were averaged.

Launch Conditions

The launch velocity (for the assumed launch aircraft) was specified to be 700 ft/sec at 5000 feet altitude. An ideal launch was assumed so there were no initial angular rates imparted to the missile. The launch aircraft heading and pitch attitude were variables to allow the genetic algorithm find the best way to design, as well as launch, the missile.

Goals for Guided Interceptor

The goals against both targets were to minimize both miss distance and intercept time. Given the pareto genetic algorithm, to win the tournament selection process, a competing solution must be better than a competitor in both goal areas simultaneously.

Results: Ground Launched Interceptor

With the design problem and parameters completely defined, the pareto genetic algorithm was executed until satisfactory missile performance was obtained. Figure 6 shows the convergence history for each goal. As this figure shows, it took the genetic algorithm 12 generations (1800 attempts) before it found a design that was capable of even lifting off the ground. Designs that could not produce thrust were given a miss distance error of 1E+5 so that the genetic algorithm would learn that these were very bad designs. Once the genetic algorithm learned how to produce thrust, the miss distance fell to 298,000 feet at generation 12. This large miss distance was due to tail fin failure just after lift-off. Luckily, within two more generations (generation 14), the genetic algorithm found a design that would not fail structurally or thermally. This design flew to within 30 feet of the target. Such a large change in performance is rather unusual and highlights what can happen if the right crossover or mutation occurs at the right place and at the right time. This is not to say that the best design of generation 12 actually produced the design at generation 14, but that the surviving genes from generation 12 combined with other survivors and produced a 2nd generation descendant with substantially improved performance. These two designs are substantially different (see Figure 7). The generation 12 design had a radius of 8.5 inches and a takeoff weight of over 2500 pounds. The generation 14 design had a radius of 6.3 inches and a takeoff weight of 1700 pounds. The design at generation 14 had much smaller tail surfaces, so these surfaces were able to maintain their structural integrity. The generation 12 tail surfaces failed within 1/2 second of autopilot initiation.

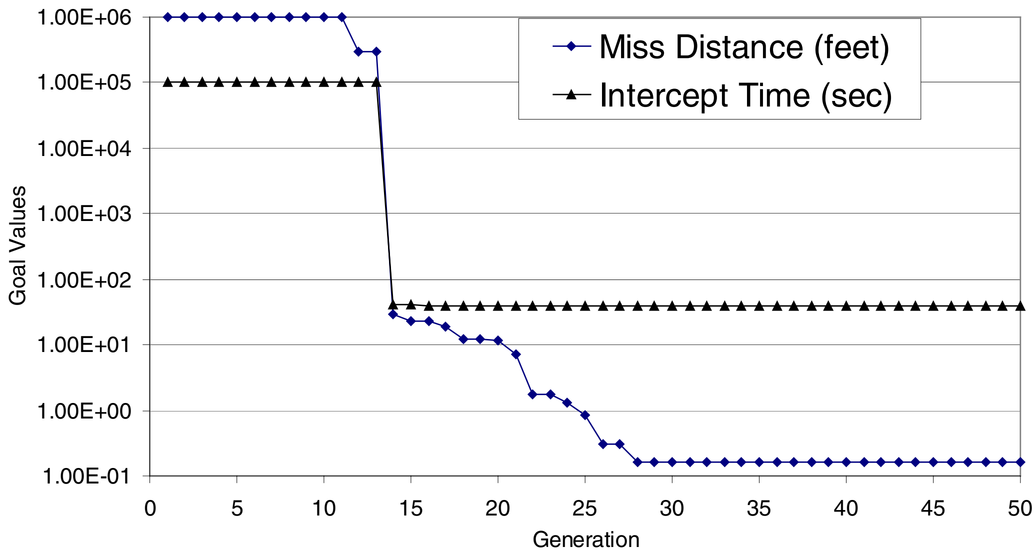


Figure 6. Convergence History for Guided Interceptor Goals

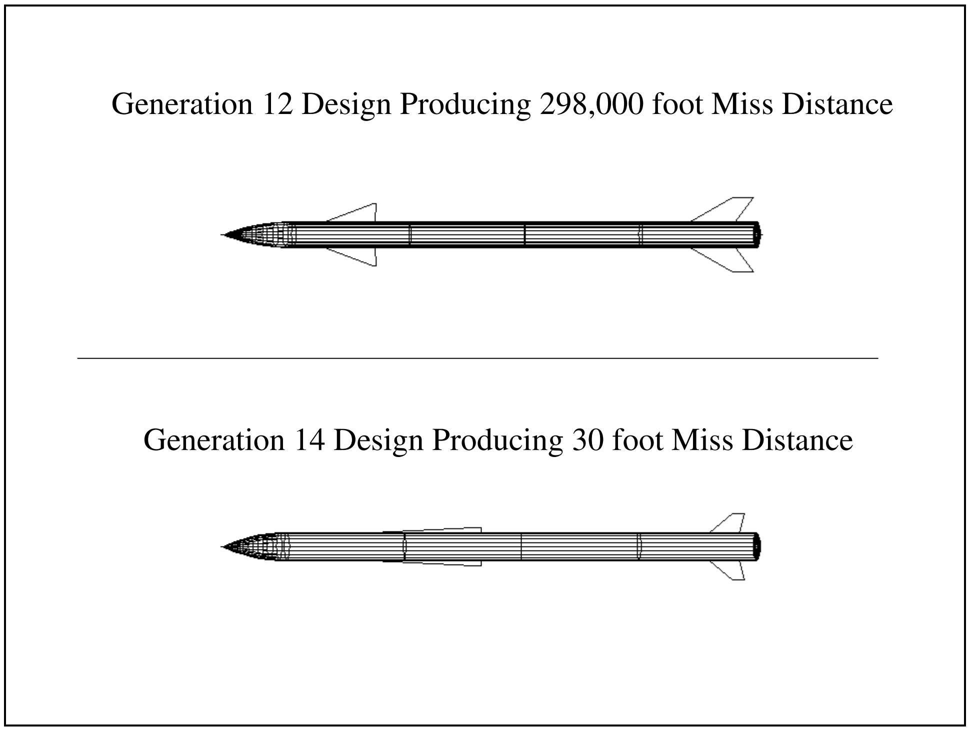

Generation 12 Design Producing 298,000 foot Miss Distance
Generation 14 Design Producing 30 foot Miss Distance

Figure 7. Design Changes from Generation 12 to Generation 14

The minimum miss distance continued to improve from generation 14 to generation 28, falling to within 0.16 feet. Given the time step that was used for the last one-second of the engagement, the minimum miss distance could only be 0.1 feet at best. So, it is fair to say that the accuracy of the system was near a maximum by generation 28. Prior to the last one-second of flight, the time step was such that the system accuracy could be no better than approximately 5 feet. A larger time step saves considerable computer run time and, for preliminary design efforts, saving computer time is important. No further improvement in the minimum miss distance was seen between generation 28 and generation 50.

Though difficult to see in Figure 6 because of the scale, the minimum intercept time fell from 41.6 seconds to 38.5 seconds between generations 14 and 50, and the takeoff weight fell from over 2500 pounds to less than 700 pounds. The convergence figure does not imply that the 700 pound rockets actually hit the targets, nor does it imply that the designs that pulled less than 0.2 G's hit the target. Rather, this figure merely shows that, within the 150 members of each population, the lightest rocket capable of lift-off weighed less than 700 pounds and at least one rocket never pulled more than 0.2 G's during its flight. Analysis of these particular systems showed that the light rockets did not fly very far and that the low-G rockets barely lifted off.

Since there were multiple goals involved in the design process, it is appropriate to examine the history of each goal area through the generations. Figure 8 shows miss distance and intercept time for several generations. It is clear that more and more members of the population maneuver closer and closer to the target as the generations progress. In generation 14, 36 members of the population (36 out of 150) had a miss distance within 100 feet. By generation 50, 124 members of the population reached within 50 feet of the target. By inspection it is also clear from this figure that intercept time falls appreciably between generations 30 and 50, even though the two populations have similar miss distance distributions. Of solutions with a 5-foot miss distance, intercept times varied between 38.8 seconds and 42.3 seconds.

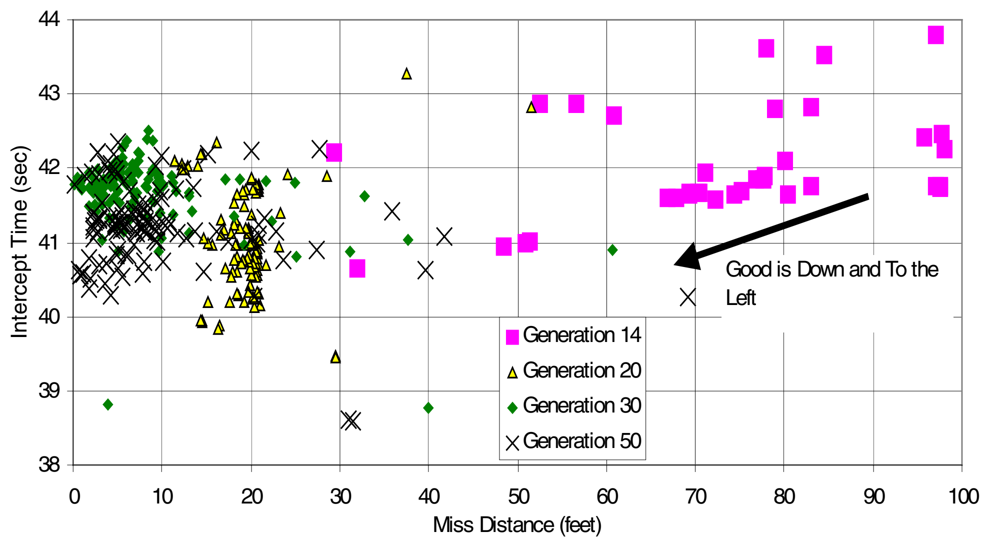


Figure 8. Miss Distance and Intercept Time: Generations 14, 20, 30, and 50

Figure 9 shows the prevalent time constants and damping ratios that dominated generation 50. There are clearly not 150 individual points (representing members of the population) on this figure because, by generation 50, many members of the population were using exactly the same damping ratios and time constants. High system damping is obviously preferred.

This result should be expected since overshooting acceleration commands is not a desirable missile flight characteristic because it wastes energy. The preferred time constants were in the 0.5 to 0.6 second range, which is very reasonable since the target is not conducting evasive maneuvers to escape the interceptor. Missiles that are designed to intercept high-G (9-G's is typical) maneuvering targets have time constants near 0.2 to 0.3 seconds so that they can quickly respond to target evasive tactics.

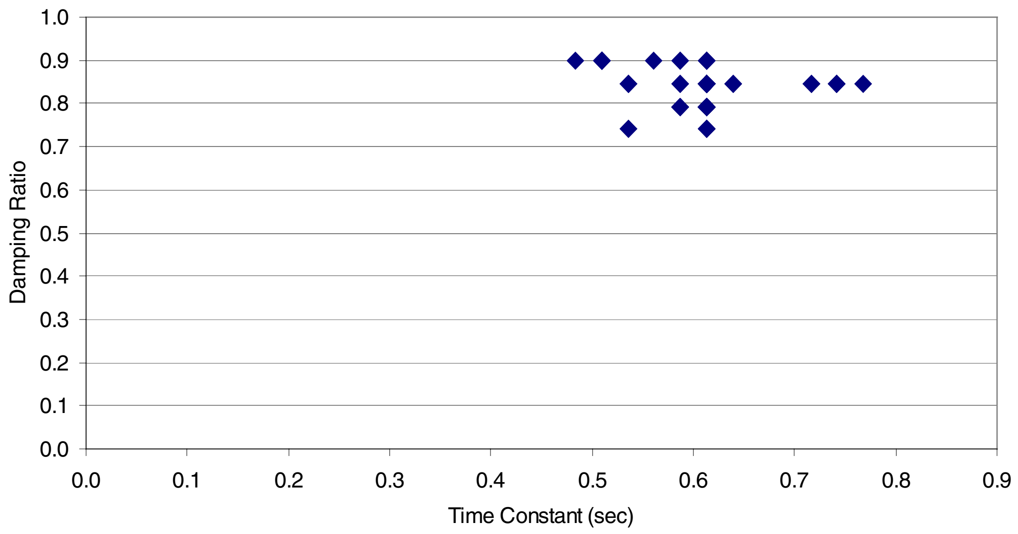


Figure 9. Generation 50 Time Constants and Damping Ratios

Figure 10 shows the proportional navigation gains and crossover frequencies that dominated the population at generation 50. Fairly high navigation gains (4.7-5.0) dominated the population, which means that system quickly tried to minimize heading errors. Low values of the navigation gain, in the 2.0 to 3.5 range, would tend to delay the correction of heading errors. For high altitude intercept missions, it makes sense to take out heading errors early in the flight, rather than waiting until the altitude is such that system responsiveness suffers from the lack of air density (i.e. dynamic pressure). The dominant crossover frequencies were between 20 and 30 rad/sec. This result is not too surprising since the highest value that could safely be used[1] was roughly 41.66 rad/sec, which corresponds to 1/3 of the bandwidth of the actuator used in this study.

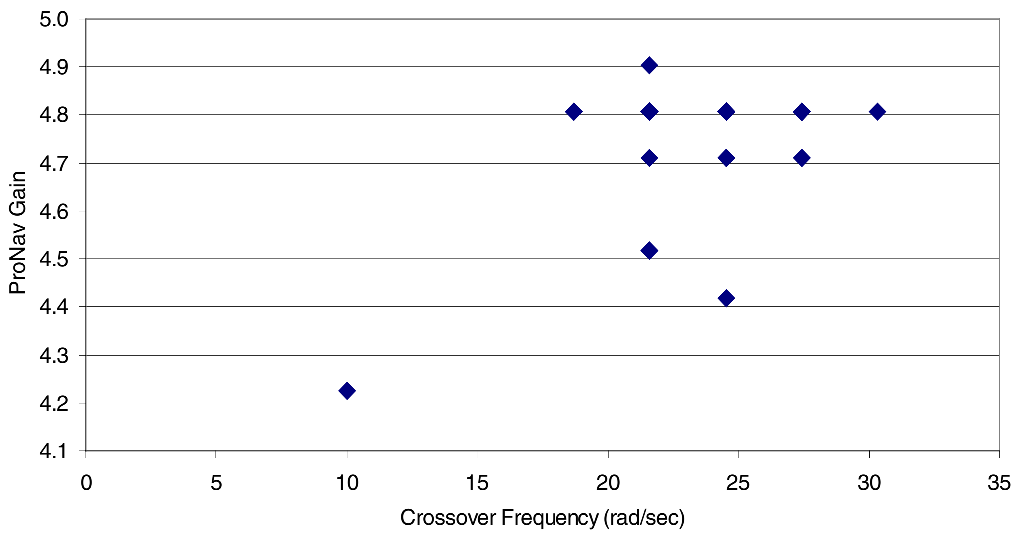


Figure 10. Generation 50 Proportional Navigation Gains and Crossover Frequencies

Although neither the launch angles nor the autopilot performance parameters show great variation among the members of the population, the actual missile designs produced during the solution process are quite diverse. Figure 11 through Figure 14 show the diversity that exists in a population. The first generation 30 design that is shown happens to be the design that intercepted the target substantially faster that it's contemporaries (recall Figure 8). This nine-pointed star design has fairly small wings and the tail control surfaces are well ahead of the aft end of the rocket. The nose is also fairly blunt. This missile is, however, able to fly to within four feet of the target more quickly than the others. The time constant, damping ratio, and crossover frequency for this design was 0.69 seconds, 0.85, and 21.6 rad/sec, respectively. Certainly such a long time constant could help explain some of the miss distance. Smaller time constants help make the system more responsive. The navigation gain was 4.52, which will tend to make the system minimize heading errors as early in the flight as possible when there is sufficient dynamic pressure to make steering easier. The designs shown in Figure 11 include forward and aft wing placements, low and high wing taper ratios, swept and unswept wing/tail trailing edges, and cone and ogival noses that have both high and low fineness ratios. The solid rocket motors themselves are fairly consistent, having a large initial burning area (9 and 10 point star grains) and large combustion chamber volumes.

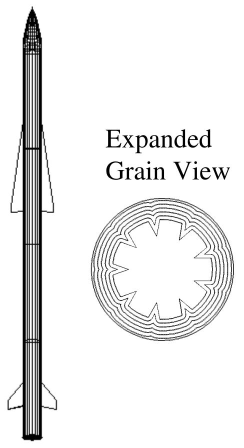
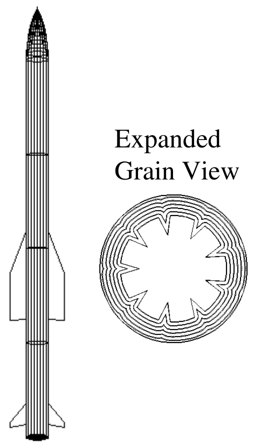
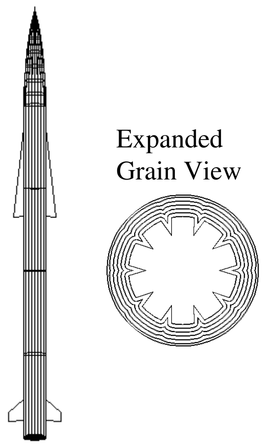
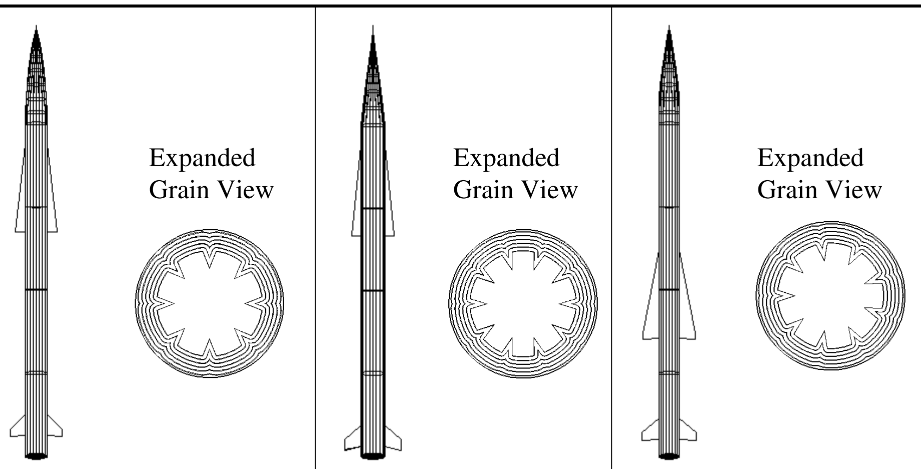
		
Generation 30 Miss Distance: 3.9 ft Intercept Time: 38.8 sec Takeoff Weight: 1729 lbf G-Load: 44.6G's Length: 367.4 inches Radius: 8.5 inches	Generation 30 Miss Distance: 3.2 ft Intercept Time: 41.0 sec Takeoff Weight: 1706 lbf G-Load: 43.7G's Length: 354.7 inches Radius: 9.6 inches	Generation 40 Miss Distance: 3.2 ft Intercept Time: 40.9 sec Takeoff Weight: 1599 lbf G-Load: 47.4G's Length: 354.7 inches Radius: 9.6 inches

Figure 11. Sample Designs from Generations 30 and 40

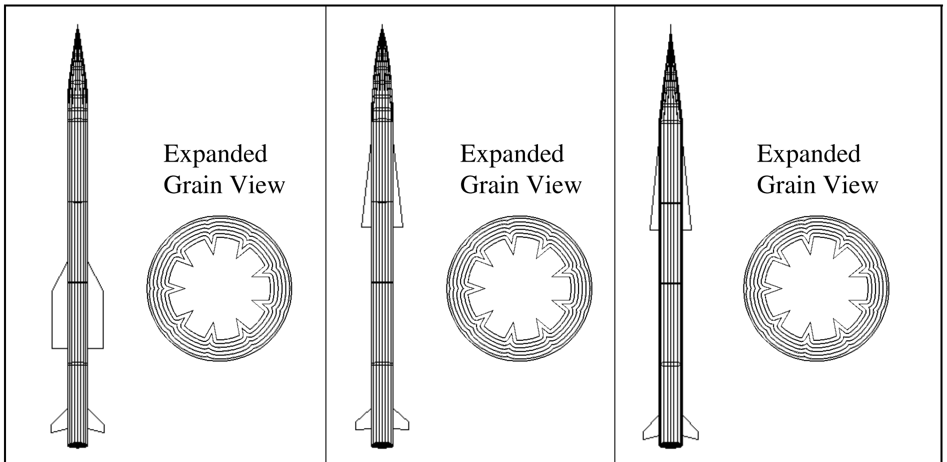
The designs shown in Figure 12, Figure 13, and Figure 14 all come from generation 50, the final generation, and are segmented into designs that have miss distances less than 3 feet, 2 feet, and 1 foot respectively. Not all the designs that meet these accuracy criteria are shown. These particular designs are merely representative of some of the more accurate designs in the population. By generation 50, the genetic algorithm has found two basic classes of external designs that work fairly well: highly tapered, small semi-span, forward-placed wings and moderately tapered, moderate semi-span, aft-placed wings. The genetic algorithm found that large semi-span designs produce large bending moments (which tend to cause structural failure), so wing areas were held at reasonable levels by increasing the root chord. The tail surfaces are fairly similar, though the placement of the surfaces varies slightly as does tail size. The placement of the control surfaces dictates where the throat of the rocket motor is placed (to make room for the actuators within the missile body), so the length of the nozzle expansion region varies. In no case, however, did the actual exit area exceed the diameter of the missile body. Designs yielding nozzle exit radii exceeding the radius of the missile body would produce excess drag, and the genetic algorithm learned to avoid these types of designs. There is a chance, therefore, that the rocket motor exhaust pressure would not match local ambient conditions very well during the boost phase of the interceptor, and this question will be examined in more detail later. First though, the rocket motor grain designs appear to be very similar, but Figure 12 shows examples of 8, 9, and 10 pointed star grains. A common feature in generation 50 was also present in generation 30 and generation 40, namely large initial burning areas and large combustion chambers.



Generation 50	Generation 50	Generation 50
Member: 105	Member: 134	Member: 88
Miss Distance: 2.2 ft	Miss Distance: 2.1 ft	Miss Distance: 2.1 ft
Intercept Time: 41.4 sec	Intercept Time: 41.9 sec	Intercept Time: 40.5 sec
Takeoff Weight: 1606 lbf	Takeoff Weight: 1608 lbf	Takeoff Weight: 1613 lbf
G-Load: 39.7G's	G-Load: 42.7G's	G-Load: 40.8G's
Length: 354.7 inches	Length: 348.4 inches	Length: 354.7 inches
Radius: 9.6 inches	Radius: 10.1 inches	Radius: 9.0 inches

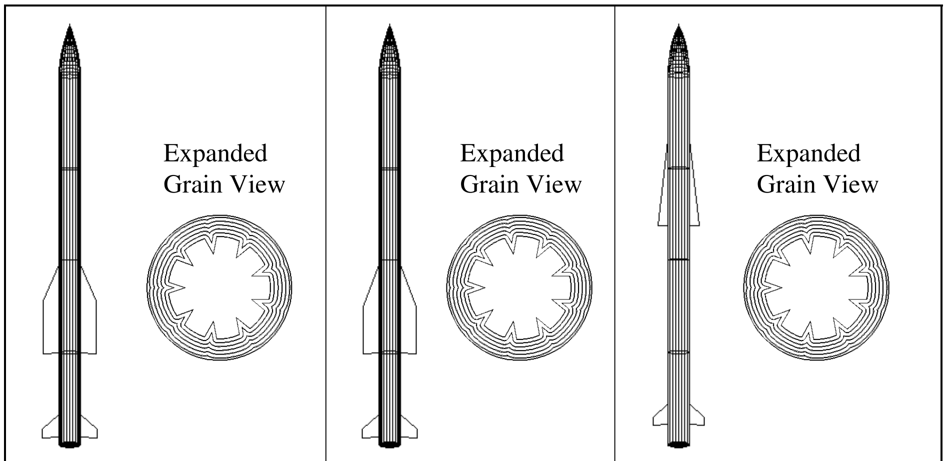
Figure 12. Three Generation 50 Designs with Miss Distances Less Than Three Feet

Nose shapes also continue to vary between ogive's and cone's, although it appears that the ogive nose exists in the more accurate examples. Physical sizes and takeoff weights of the interceptors are very similar, as are intercept times and maximum G-loads. The three best designs (in terms of miss distance only) all had 9-pointed star grains, but the wing sizes and locations still vary significantly. It is also interesting to note that these designs all had nose shapes that were fairly blunt compared to the other designs that were slightly less accurate. These nose shapes were not blunt enough to incur a drag coefficient penalty larger than 0.012 based on a Sears-Haack body, so an examination of the aerodynamic data for these shapes revealed that the net effect of the change in the nose length was to move the center of pressure farther forward very slightly (an average of approximately 1.5 inches) over all flight conditions. This center of pressure movement helped reduce the static margin at rocket motor burnout, thereby increasing maneuverability at the coast condition without seriously impacting maneuverability during rocket motor burn.



Generation 50	Generation 50	Generation 50
Member: 56	Member: 71	Member:94
Miss Distance: 1.8 ft	Miss Distance: 1.7 ft	Miss Distance: 1.3 ft
Intercept Time: 40.4 sec	Intercept Time: 40.7 sec	Intercept Time: 41.9 sec
Takeoff Weight: 1665 lbf	Takeoff Weight: 1614 lbf	Takeoff Weight: 1619 lbf
G-Load: 39.7G's	G-Load: 44.4G's	G-Load: 38.9G's
Length: 354.7 inches	Length: 354.7 inches	Length: 348.4 inches
Radius: 9.0 inches	Radius: 9.6 inches	Radius: 10.1 inches

Figure 13. Three Generation 50 Designs with Miss Distances Less Than Two Feet



Generation 50	Generation 50	Generation 50
Member: 132	Member: 36	Member: 1
Miss Distance: 0.8 ft	Miss Distance: 0.6 ft	Miss Distance: 0.16 ft
Intercept Time: 40.5 sec	Intercept Time: 40.6 sec	Intercept Time: 41.7 sec
Takeoff Weight: 1667 lbf	Takeoff Weight: 1655 lbf	Takeoff Weight: 1602 lbf
G-Load: 39.5G's	G-Load: 39.8G's	G-Load: 39.5G's
Length: 348.4 inches	Length: 348.4 inches	Length: 354.7 inches
Radius: 9.0 inches	Radius: 9.0 inches	Radius: 9.6 inches

Figure 14. Three Generation 50 Designs with Miss Distances Less Than One Foot

It is obvious from Figure 15 why the interceptor is able to build up Mach 4 speeds so fast. The initial take-off thrust is nearly 64,000 lbf and, as the sharp star points burn off, the thrust reduces to approximately 37,000 lbf before increasing again as the burning area begins to increase during the final burning phase. The chamber pressure is nearly 2,000 psi initially, which is well within the 3,000 psi limit of the rocket motor case. Higher chamber pressures may be more efficient in terms of utilizing the case weight that is present (less rocket case steel could have been used had thickness been a design variable), but given the fact that the missile nearly reaches the thermal limit, this may be a case where not violating the thermal barrier was more important than a few extra pounds of weight.

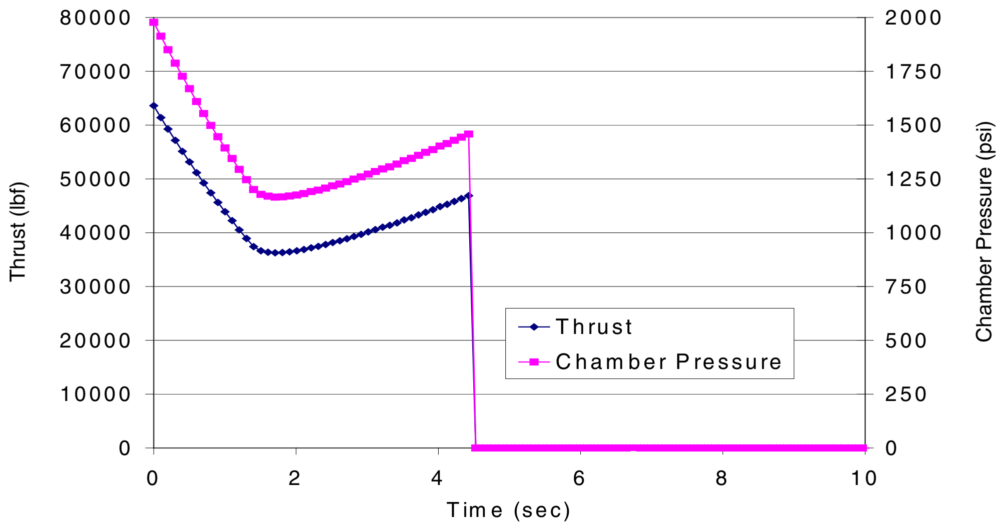


Figure 15. Thrust and Chamber Pressure History

One clear measure of thrust efficiency, however, is the ratio of the nozzle exit pressure to the local ambient pressure (P_e/P_a). A ratio of 1.0 is the most efficient, and Figure 16 shows that the genetic algorithm did a good job of compromising the exit pressure ratio to be both above and below the ideal ratio during the burn. Since a variable nozzle exit area was not possible with this system, the ideal ratio cannot be maintained. It is remarkable, however, that the genetic algorithm was able to design the rocket motor throat/exit expansion ratio so that a reasonable pressure ratio was obtained while keeping the nozzle exit radius within the body case radius to avoid a drag penalty. The ability of the genetic algorithm to simultaneously work on multiple goals while under multiple constraints/penalties makes it a robust "hands-off" design tool. The fact that the genetic algorithm also seems to yield good results for implicit (not-stated) performance goals, such as maintaining a nozzle exit pressure ratio near 1.0, also highlights the value of the technique.

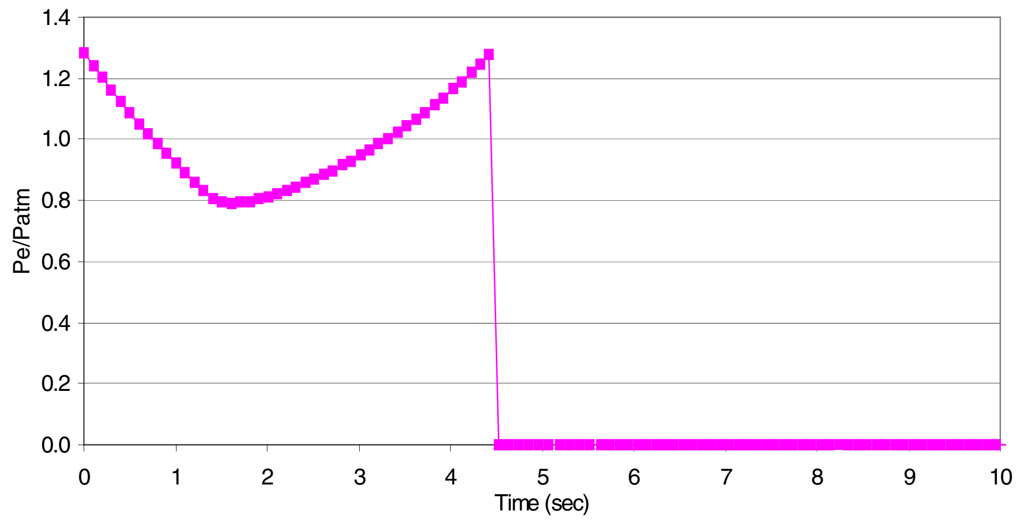


Figure 16. Exit Pressure Ratio History

Results: Air-to-Air Interceptor

Figure 17 shows the convergence history of the best performer in the two goal areas. There was a major performance improvement at generation 108, and continued learning until generation 150. Recall that each potential design was flown against 10 separate random targets, so it should be expected that there would be some noise in the miss distance goal even though elitism was chosen.

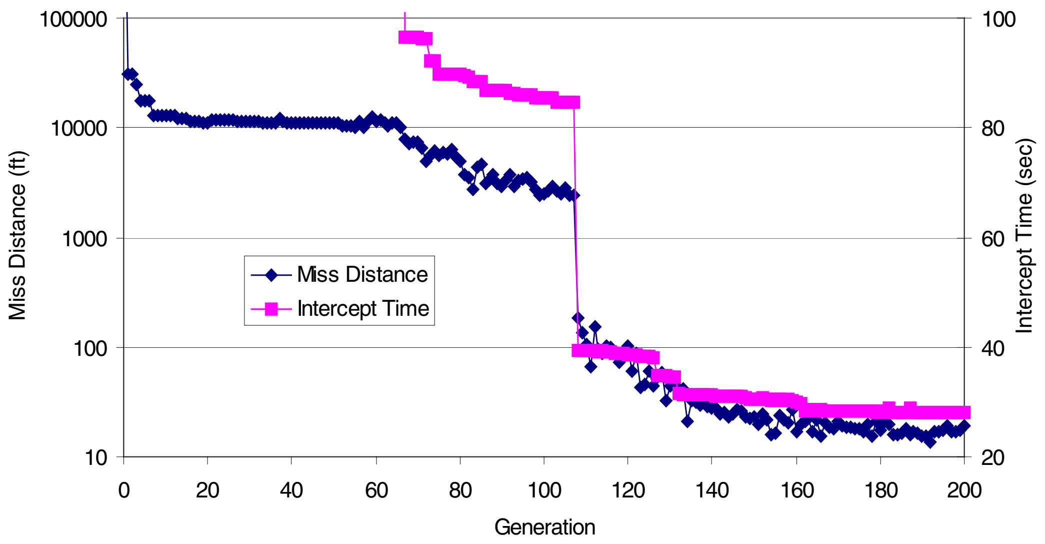


Figure 17. Convergence History of Best Performer in the Two Goals

Figure 18 shows population diversity in the two goals at several generations. There is very clear performance improvement in both goals from generation 120 to generation 150. More and more of the population members began to take on the successful characteristics of their ancestors, and, therefore, the number of competitive solutions began to increase. This is most obvious when looking at the sheer number of solutions that are competitive in generation 150 versus generation 120.

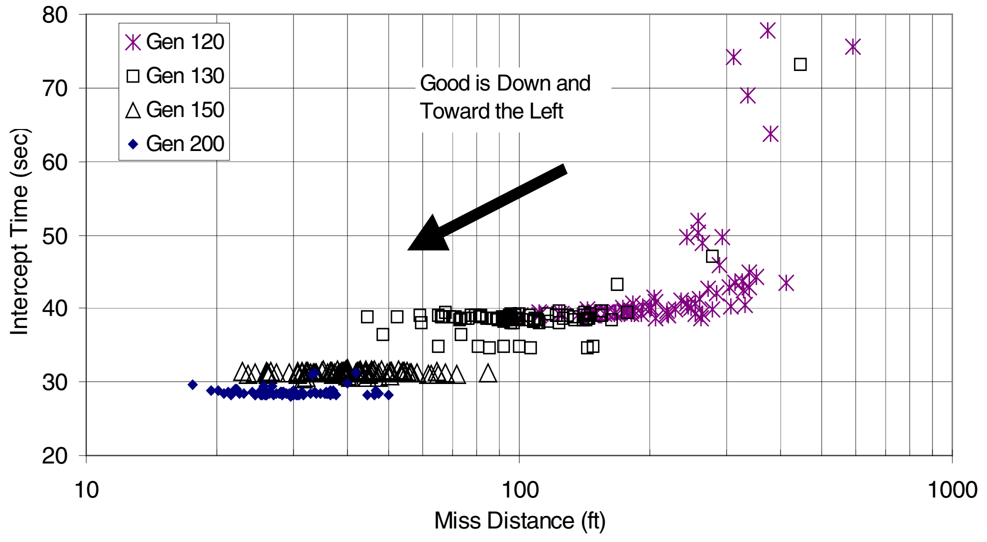


Figure 18. Intercept Time and Miss Distance Diversity

Some of the more competitive missile designs from generation 200 are shown in Figure 19. The most obvious feature of these designs is the large wing/tail surfaces. For high maneuverability, large surfaces are needed since thrust vectoring is not currently part of the design process. Wing placement is also very far forward in order to enhance turning ability. As the rocket motor burns the center of gravity of the missile moves forward, so a forward wing placement helps offset some of the inherent stability due to center of gravity motion. It is interesting to note that a circular cross section was preferred for the solid grain, meaning a progressive thrust profile. One reason for choosing a progressive thrust profile could be that, since the launch point was at 5,000 feet (altitude) and the target was at 20,000 feet, it might be beneficial from a thermal viewpoint to delay achieving maximum Mach number until reaching an altitude with a lower static temperature.

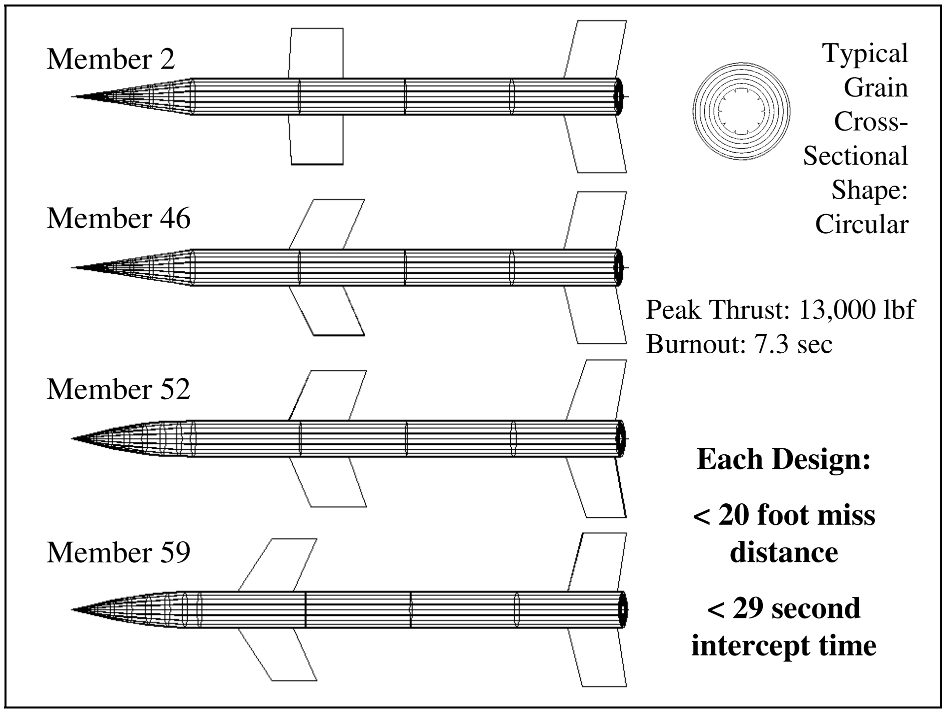


Figure 19. Various Missile Designs from Generation 200

Figure 20 shows the Mach number and temperature profile for one of the flights for the first missile (member 2) shown above. As you can see, the genetic algorithm was pushing the thermal limit in order to minimize the intercept time, so having a progressive thrust profile probably helped keep the missile from exceeding the thermal limits. Prior to burnout, the missile had climbed to approximately 10,000 feet and had a Mach number near 4.5. Had the missile been at 5,000 feet at Mach 4.5, it would have exceeded the thermal limit of 2500 degrees Rankine by over 25 degrees.

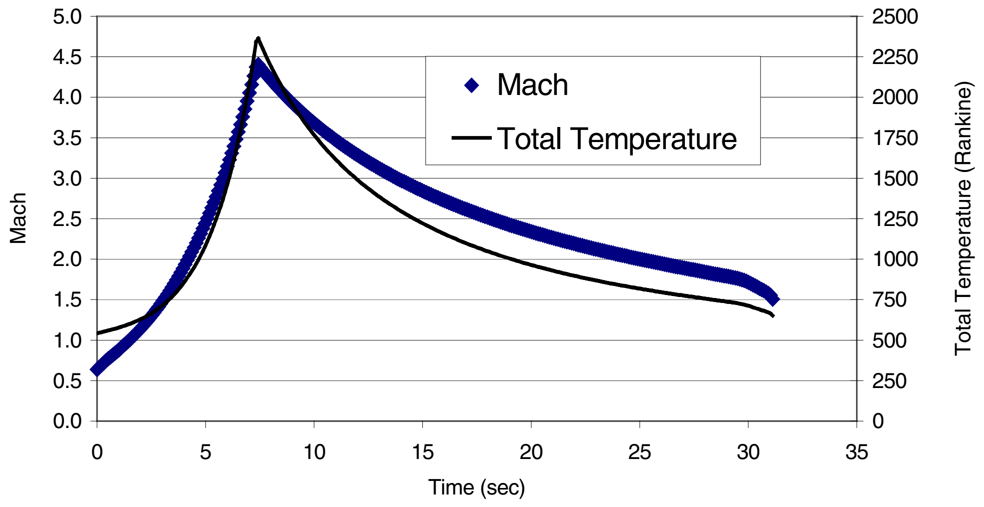


Figure 20. Mach Number and Total Temperature Profile for a Rapid Intercept Case

These acceleration histories were achieved with the types of damping ratios, time constants, navigation gains, and crossover frequencies shown in Figure 21 and Figure 22. The time constants concentrated around 0.37-0.47 seconds. Recall that there are 100 actual points on these figures, though it appears that there are less because many points are the same. In fact, of the 100 solutions at generation 150, over 80 of them used a time constant between 0.37 and 0.42 seconds. The favorite damping range was from 0.63 to 0.88, with a fairly even distribution of solutions in this range. This is moderate damping.

The Navigation gains are fairly conventional for air-to-air interceptors (4.5-4.9), but many of the specified crossover frequencies are very high. There is a cluster of solutions within the conventional air-to-air missile range, between 47-54 rad/sec. But there is another cluster between 85 and 94 rad/sec that is close to the actuator natural frequency of 125 rad/sec. Analyses of these high crossover frequency cases shows that, although they are only marginally stable at certain flight conditions, they are (on average) just as accurate as the solutions having lower crossover frequencies. They do, however, have lower stability margins and are more prone to excitation at certain airspeeds. Since there was no goal established to assess stability per se, it is natural that these types of solutions would survive since their performance in the two stated goals was comparable to the more stable solutions.

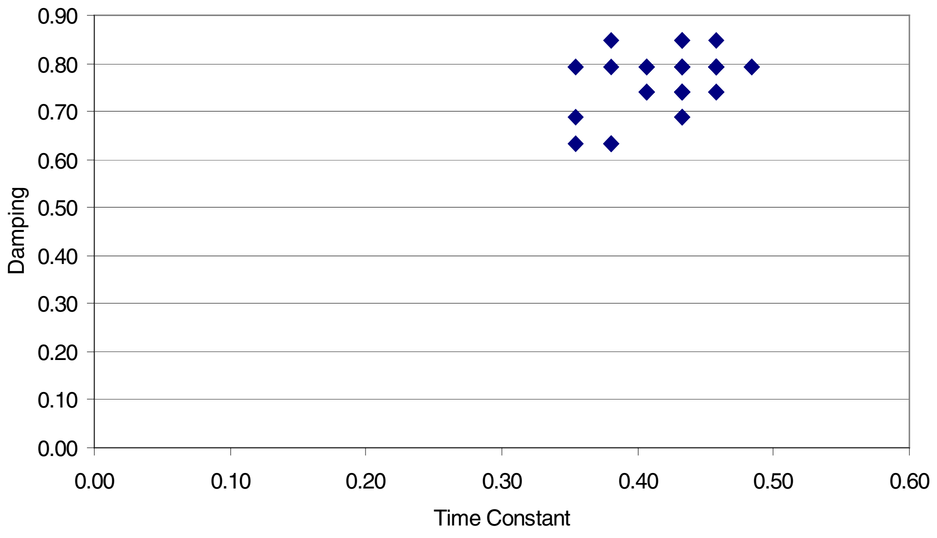


Figure 21. Generation 100 Time Constants and Damping Ratios

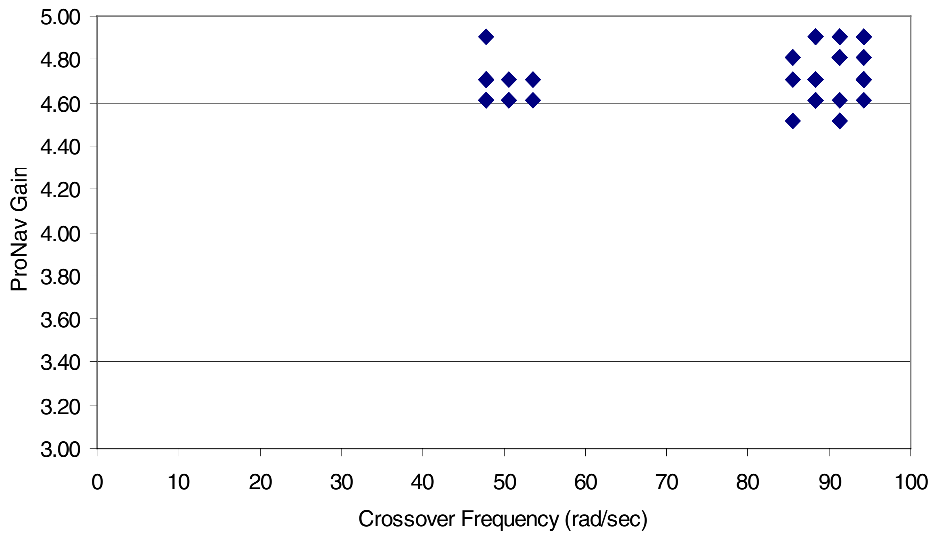


Figure 22. Generation 100 Navigation Gains and Crossover Frequencies

Computer Run Time

The 200 generations presented here required 10 days of CPU time on a Silicon Graphics R-10000 processor. The R-10000 is approximately the same speed as a 400 MHz Pentium II processor.

Conclusions

The pareto genetic algorithm is well suited to designing complete interceptor systems consisting of propulsion, aerodynamics, and autopilot modules. This work has shown that an all-at-once design process, controlled by the proper artificial intelligence tool, is not only possible, but much faster than iteratively designing each subsystem component into a workable package. Simple constraints, such as the thermal limits and structural integrity calculations used in this work, provide a means of injecting real-world considerations into the design process. Even with diverse performance modules and diverse goals, the genetic algorithm was able to learn how to design around the constraints while achieving good performance in overall system goals. In both the difficult high-altitude/high-speed engagement scenario and the highly maneuverable air-to-air target scenario, the genetic algorithm developed multiple designs capable of close intercept.

The basic three-loop autopilot is ideal for preliminary design studies. Improvements such as thrust compensation could, and should, be included when the design process moves beyond the preliminary stage. The analytic determination of gains, and gain schedules based on Mach number and altitude, have made three-loop autopilots very popular in current missile systems. The basic proportional navigation guidance algorithm used here worked well, but it is likely that performance could be improved by using multiple algorithms during different phases of flight to correct, for example, the adverse acceleration commands at low launch speeds.

References

[1] Zarchan, Paul, *Tactical and Strategic Missile Guidance*, 3rd Edition, Volume 176, American Institute of Aeronautics and Astronautics, 1997.
[2] Nesline, F.W. and Nesline, M.L., "How Autopilot Requirements Constrain the Aerodynamic Design of Homing Missiles," Conference Volume of 1984 American Control Conference, San Diego, CA, June 6-8, 1984.
[3] Stallard, D.V., "An Approach to Autopilot Design for Homing Interceptor Missiles," AIAA- 91-2612-CP, AIAA Guidance and Control Conference, 1991.
[4] Washington, W. D., "Missile Aerodynamic Design Program," 1980, 1990.
[5] Ashley, H., and Landahl, M., *Aerodynamics of Wings and Bodies*, Dover Publications Inc., New York, New York, 1965.
[6] Etkin, Bernard. *Dynamics of Atmospheric Flight*, John Wiley and Sons, 1972.

Analyzing Rocket Plume Spectral Data with Neural Networks for Condition Monitoring

Kevin W. Whitaker
Associate Dean for Academic Programs
College of Engineering
The University of Alabama
P.O. Box 870200
Tuscaloosa, Alabama 35487-0154
USA

Abstract

Space Shuttle Main Engine fault detection systems typically rely on sensor data analysis via redundant rule-based expert systems along with visual observations for the real-time assessment of engine health. A novel alternative to the traditional health monitoring approach is predicated upon the acquisition and subsequent neural network processing of electromagnetic plume emissions. Spectrometric examination of an emission spectrum provides a means for the identification and quantification of metallic species indigenous to the main engine plume flow. Knowledge of the metallic species eroding could pinpoint the specific location of component degradation within the engine as well as identify serious component failures at an early stage. Such an approach is advantageous because it allows for the detection of numerous internal failures that would otherwise go unnoticed by traditional monitoring methods. This paper details a radial basis function neural network architecture that is capable of inferring metallic state from a given plume spectrum. Specifically, a comprehensive discussion of the methodologies necessary for the development and implementation of the neural network approach are provided. The resulting neural networks are validated with actual test-stand data from an actual Space Shuttle Main Engine at NASA's Stennis Space Center.

Nomenclature

y	=	desired function for neural approximation
$\boldsymbol{x}$	=	neural network input feature pattern
$\boldsymbol{g}_j$	=	Gaussian kernel function
μ_j	=	kernel function center position
σ_j	=	kernel function spread constant
$\boldsymbol{w}$	=	optimal array of network weighting coefficients
$\mathbf{G}$	=	array of Gaussian kernel responses to input patterns
$\mathbf{I}_i$	=	vector of spectral intensities, (Watts•cm^{-2}•str^{-1}•nm^{-1})

Introduction

Current Space Shuttle Main Engine (SSME) operating health assessment procedures employ an expert-rule based platform with redlining capability. This approach is very effective at identifying potentially catastrophic engine failures but it cannot quantify the degree of internal engine component degradation experienced during an engine's cycling period. For example, the degree of erosion of a pre-burner faceplate or nozzle side-wall is assessed by a costly post-test, or post-flight, visual inspection. In many instances, SSME inspectors do not have prior knowledge of what engine components to check and isolate as potentially dangerous for future engine operations. This paper offers an attractive alternative to the traditional health monitoring approach. It is a method that will allow for the isolation of SSME problem areas and also provide crucial real-time information that can be used to assess the internal health of the SSME. Specifically, radial basis function neural networks are used to approximate the underlying functional mapping between plume emission spectra and metallic species content. The resulting neural platform is capable of providing the metallic concentrations contained in the SSME plume.

For years, researchers at NASA's Marshall Space Flight Center (MSFC) have been filming developmental tests of the Space Shuttle Main Engine. Subsequent analysis of film that involved major engine failure incidents revealed a consistent feature common to most of the breakdowns. In 8 of 27 failure events observed, a visible discharge of some substance was seen in the plume prior to the engine failure.[1-3] These discharges ranged from extreme flashes to small regional streaks. The discovery suggested an interesting approach to SSME health assessment, namely, if it is possible to visually detect anomalous events with the naked eye, then perhaps it would be possible to discover anomalous events below the "visible threshold" (infrared, ultraviolet, etc.) that would be indicative of an imminent failure.[1,2] This would be tantamount to real-time engine health monitoring via the quantitative analysis of the SSME exhaust plume spectral data.

The idea of extracting chemical data from the analysis of the electromagnetic (EM) spectrum is not new. Holding a copper wire in a sufficiently hot flame produces a characteristic green region in the flame. The copper atoms are excited to such a high energy state that they emit electromagnetic radiation at several wavelengths, with green light being dominant. The atomic structure determines the wavelengths of EM emitted, and since all elements are unique, no two elements will emit EM at exactly the same wavelengths. Thus, each element has its own unique spectral signature. For example, in the same flame, the element nickel will emit EM at wavelengths different than copper. A plot of radiant intensity versus wavelength is called the electromagnetic spectrum and an example of a typical spectrum for nickel is shown in Fig. 1. Since this spectrum is unique, a spectrometric detector some distance from the flame would allow a user to determine the presence of nickel, or any other material for that matter, if its spectrum was known.

Individual spectra for other elements vary in complexity, some having a few atomic transitions (peaks), others having many. Three germane points of importance result from considering Fig. 1: 1) *Every element has its own "spectral signature,"* 2) *The emission will contain atomic transitions at wavelengths which may not be part of the visible spectrum, and* 3) *The* ***intensity*** *of the emission is a function of the* ***quantity*** *of emitting matter present in addition to the system temperature and other quantum variables.*

Rocket plumes are emissive events subject to the same physics (with more complications of course) as burning nickel or copper in an open flame. The Optical Plume Anomaly Detection (or OPAD) program was initiated by researchers at MSFC as an effort to take advantage of the wealth of information contained in the exhaust plume of a rocket engine. The initial idea was to identify anomalous spectral events which were consistent with known mechanical failures and then use them as templates in the health monitoring of future engine tests, ground or in-flight. These spectral templates could then be coupled with the anomalous events found in the vibrational and other sensor data to determine the overall state of the engine.[1]

The Technology Test Bed (TTB) at MSFC and the A1 Test Stand at NASA's Stennis Space Center (SSC) were the first to receive an OPAD instrument pack developed by combustion phenomenologists at the U. S. Air Force Arnold Engineering Development Center (AEDC). Soon thereafter, the process of building a cumulative database of the spectral templates began. Researchers wanted to catalogue the various spectral forms associated with changes in SSME operation (*i.e.*, changes in oxygen/fuel ratio, engine startup, etc.) so that a baseline of "expected" spectral signatures would be established. The "template idea," however, soon gave way to even more ambitious goals as a result of some initial findings in the TTB experimental program.[1,3] Specifically, health monitoring now involved the simultaneous tasks of anomaly detection and metallic species quantification (see Fig. 2). This meant that the free atom densities of all the metals of interest within the engine would have to be inferred for every spectral scan taken by the instruments. The quantification process would essentially give metal concentration versus time and thereby highlight any anomalous events indicative of a major failure.

Simply stated, the capability to detect anomalies and quantify metal erosions would be an enormous benefit to the propulsion engineering community. It could reduce costly engine servicing by specifying only those times that it would be needed. Moreover, it could pinpoint the specific engine component that needs attention (anomaly isolation). For example, post-test analysis of OPAD spectral data from one particular TTB firing revealed the erosion of a metal indigenous to the pre-burner faceplate; indeed, a subsequent analysis of the pre-burner by the engine group at MSFC found that the faceplate was in serious need of replacement.[1] Finally, the OPAD system could assist in the real-time health monitoring of launch vehicles. If an engine is operating in an "off nominal" condition, potentially jeopardizing the success of the mission or safety of the crew, the OPAD detection system could order an engine shutdown or mission termination.

The cursory overview of the OPAD capabilities given above says nothing of the complexity of the subsystems necessary to implement it. Specifically, metal quantification from an electromagnetic spectrum requires a computational model of the exhaust plume emission physics. Spectrometer hardware for the test stand, as well as the in-flight environment, had to be developed for the spectra acquisition. Lastly, methods for implementing the spectral model to allow metal determination would need to be integrated as well as validated. The central theme of the work presented here, however, will be the development of artificial intelligent systems that will perform real-time determination of the metallic species concentrations present in the exhaust plume. Specific objectives include: 1) Establishing appropriate techniques for feature extraction from a spectrum, 2) Identifying robust data transformation methods for accurate neural network performance, 3) Creating a neural architecture capable of quantifying the metallic content of the plume flow, and 4) Creating a neural platform that can perform the spectral mappings in "real-time."

Spectral Investigation of Emissions

Any spectrum obtained with the OPAD instrumentation is composed of three components: 1) a dominant OH component which arises from the burning of dissociated hydrogen radicals, 2) a background noise component caused by the scattering of background light, and 3) a metallic component, if indeed there is one, which would be indicative of a metal erosion. Thus, the quantification of a metal erosion and the subsequent identification of any anomalies requires a spectral "cleaning" procedure followed by an evaluation of the plume metallic state. In other words, methods for removing the OH and background components of the spectrum would need to be employed so that the underlying metallic component could be seen. Then the metal quantities would have to be ascertained from the remaining metallic component in the spectrum.

For a given spectrum, ascertaining the metallic quantity could only be done through two methods: 1) by comparing the spectrum to past spectra obtained from plume seeding tests, or 2) using a theoretical model that emulates the emissive nature of the plume. The first option is plagued by an inability to precisely measure the erosion and survivability rates of the inserted species. Thus, there would exist plume spectra to compare to, but the metallic content associated with the spectra would be in error. Moreover, it would not be cost effective to run the SSME through all the possible metallic seeding combinations. For these reasons, option two was selected.

Spectral models have been in use since the late 1960's at AEDC. Most of these models measured flow-field species concentrations using techniques such as emission resonance absorption, electron beam impact excitation, and most recently, Laser Induced Fluorescence (LIF). There are many excellent validated radiative transfer codes available for specific molecular species, but none have been developed that can handle the comprehensive analysis of atomic spectral data. To meet this need, researchers at Vanderbilt University and AEDC developed a theoretical plume model known as SPECTRA.[4,5] The SPECTRA model can provide quantitative plume spectral analysis for emission, absorption, and LIF configurations installed at the rocket nozzle exit plane. Validation of this code is still ongoing, but carefully controlled SSME plume seedings at the TTB have shown the model to be consistent.[4]

The forward operation of the SPECTRA code involves the calculation of a theoretical plume spectrum from a pre-defined set of metal concentrations and flow parameters. The reverse operation of SPECTRA, namely, obtaining the metallic components which made up the spectrum, cannot be written in a numerically convenient form because of the insurmountable mathematics involved. This, therefore, mandates that the SPECTRA code be applied in an iterative manner until it converges with the spectrum obtained from the OPAD instrumentation. The set of SPECTRA input parameters which produced this convergence would then specify the current metallic state of the plume. Accordingly, higher order numerical fitting routines have been developed for such a purpose and the resulting spectral fits obtained using these codes are very accurate.[6,7] However, the time necessary to process a single spectral scan is on the order of hours while the demands of real-time OPAD require processing a scan every half second.[7] For this reason, neural network techniques have been investigated which can provide an initial estimate of metal concentrations in the rocket plume. These estimates could then be used for evaluative purposes in a real-time environment or as a starting point in post-test analysis for more accurate fitting routines, thereby significantly reducing the amount of computational effort.

The OPAD Neural Network Approach

A classic use of neural networks is function approximation. Specifically, for the OPAD case, the function would be the inverse of the theoretical SPECTRA model. Using large amounts of spectral data generated from SPECTRA, it should be possible to train a neural network to learn this inverse mapping. Subsequently, when presented with a spectrometer scan, the neural network could estimate the metallic content of the plume in real-time. (Predictions in under 0.5 sec was set as a goal for this study.)

Figure 3 supplies an overview of the multiple neural network architecture developed in this investigation. Notice that there is one neural network for each metallic species being monitored. With this setup, the steps in assessing the SSME internal health are as follows: 1) Acquire a spectrum from the plume emission, 2) Extract information about the spectral signatures of the individual metallic species being monitored, 3) Give this information to the appropriate metal neural network for a prediction of the metal's number density, 4) Monitor all the metallic erosions in real-time and identify anomalous emissions, and finally 5) From the list of eroding metals, identify the failing engine component and severity of the erosion.

There can be found numerous types of neural network architectures in the technical literature. For this problem, however, the radial basis function neural network (RBFNN) was chosen because of its ability to approximate highly nonlinear functions (precisely what the inverse of SPECTRA is) very quickly. The means by which these RBFNN's are trained and developed are the topics of the next section.

Radial Basis Function Neural Networks

The RBFNN can estimate a function $y(x)$ after training with a set of representative input/output pairings. The pairings in the present case would be sets of spectra and their corresponding metallic states. From a neural network point of view, this training data could come from either the SPECTRA code or emissions obtained during a test-stand firing.

With examples of input/output patterns, the RBFNN can adjust its internal parameters so that it approximates the functional mapping to within some degree of error. Qualitatively, the RBFNN does this by forming *localized* "bumps" or response regions within the input space. The superposition of the these local response regions forms a response surface that spans the space covered by the input training patterns.

By definition, a radial basis function (RBF) is one which decreases (or increases) monotonically away from a central point thereby giving it an inherent "bump" form. Classic functions that exhibit this propensity are the Gaussian, Cauchy, and the Inverse Multiquadric.[8] As an example, consider the Gaussian function given by Eq. (1):

$$g_j(\vec{x}) = exp\left[\frac{-(\vec{x}-\vec{\mu}_j)^2}{2\sigma_j^2}\right] \quad (1)$$

The RBFNN positions a collection of these RBFs throughout the space covered by the input training patterns. The parameter μ_j specifies the location of a single RBF within the input space (μ_j has the same dimension as the input vector $\vec{x}$) and the parameter σ_j determines the width of the local function. Thus, a given RBF will be centered at μ_j within the input space and have a "receptive field" which is proportional to σ_j. Moreover, it will give a maximum response for input vectors, $\vec{x}$, which are nearest the RBF center, μ_j.

With a developed approximation surface, the RBFNN estimates an output for an incoming input case by first evaluating each of the RBFs (in other words, determining where the input vector lies on the approximation surface) and then forming a weighted linear summation of their responses. The difficulty arises not from the logical evaluation of an input but rather the establishment of the network parameters, namely center positions (μ_j), RBF widths (σ_j), and output layer weighting coefficients.

The construction of an RBFNN is accomplished in a two part learning scheme known as hybrid learning (see Fig. 4). The initial forward connections of the network contain the RBF centers, μ_j, obtained through unsupervised assimilation, followed by an output layer of weighting parameters, w_j, formed through supervised instruction. Training in the unsupervised mode is done without a pre-defined learning goal; input categorization and learning must be done using correlations within the input training data in contrast to feedback from a teacher or critic. For the RBFNN, the learning scheme essentially clusters the inputs and specifies where to position the RBF centers so that the desired response coverage is obtained. Thus, via unsupervised learning, the RBF center positions (the μ_j) are chosen *a priori* and remain fixed throughout the output layer training. Methods for determining these positions as well as the RBF widths have been previously investigated.[9] The rearward connections, composing the output layer of the network in Fig. 4, specify the weighting (or regression) coefficients which are trained in a supervised fashion. Supervised means that learning is based on comparison of the network output with the known "correct" answers.

For an RBFNN with a single layer of Gaussians, given that the RBF centers (μ_j) are fixed, the optimal weight array for the output connections which gives the best functional mapping can be found using the least squares normal equation developed in multiple linear regression theory. Details of this procedure can be found in Ref. 8; the results are simply stated here. For a set of training input vectors, $\vec{x}$, with corresponding RBF centers μ, there will be an array of Gaussian neuron responses, **G**. Given this, the optimal weight array can be stated as:

$$\boldsymbol{w} = [\mathbf{G}^T\mathbf{G}]^{-1}\mathbf{G}^T\boldsymbol{y} \quad (2)$$

With the weights, centers, and widths set, the fundamental mapping can then be written as:

$$f(\vec{x}) \approx \sum_{i=1}^{m} w_i g_i(\vec{x}) \quad (3)$$

Thus, for an input vector $\vec{x}$, the solution $f(\vec{x})$ is a weighted linear summation of each RBF's response to $\vec{x}$. RBFs that are within the region of $\vec{x}$ will give the largest responses, whereas those farthest away will give negligible contributions to the series solution. Moreover, the RBF neuron responses, $g_i(\vec{x})$ will be bounded between 0 and 1 with the assigned weights (w_i) specifying the neurons heights.

A quick glance at Eq. (3) reveals that the RBFNN is a weighted summation of basis functions that are "tuned" using the training data. This is a technique which is frequently used in asymptotics and series representations of functions.[10]

Extracting Features From the Plume Spectrum

The overall goal here is to quantify, in real-time, the amount of metal erosion as well as the plume combustion temperature. The only available information given to the RBFNN is a spectral plot procured from the OPAD instrumentation. Given this, what should the neural network inputs be? In other words, what features within the data supply the most information about the state of a particular metal? The answer to such a question is one of the fundamental discriminants that ultimately determines the success of this neural network application.

Developing neural network inputs most often requires the researcher's in-depth knowledge of both the systems involved and the underlying physical phenomena occurring. There are times, however, when a bad choice of input can lead to a new form of insight into the detailed processes of the physical problem. For example, some inputs may not be as sensitive to changes in the output as was once thought, thereby forcing an inquiry as to why and ultimately leading to some form of insight. This has certainly happened throughout the years of the OPAD neural network development.[9]

Choosing network inputs starts with a firm understanding of all the possible error sources contained within the data. For the OPAD system, the effects of the background light and OH continuum on the observed emission are two such examples. Additional errors could also be caused by spectral interference; that is, some metals may emit at nearly identical wavelengths causing interference in the actual intensities obtained at those wavelengths. It is important to note that these are external error sources and they do not include the internal errors incurred from the failure of the linear regression model to fit all the RBFNN training points. The proper choice of an input structure can never eliminate the external errors, but it can help alleviate their effects on the RBFNN performance.

With the foregoing thoughts in mind, a study was initiated to find input data transformations that would assist in countering the effects of errors associated with the instrumentation and external radiation. Consider a typical OPAD spectrum shown in Fig. 5. The training of a neural network to predict properties of element "A" is desired. Assume that all the peaks (and regions) of element "A" that are not overshadowed or significantly affected by other elements and OH emissions have been located. Further suppose that a total of 200 sampling points which comprise these regions can be extracted and formed into vectors (like $\vec{I}_c$ for example). Based on the foregoing discussions, three things need to be accomplished: 1) use the smallest number of inputs to reduce computational cost, 2) apply a transformation to the inputs that will reduce sensitivity to errors, and 3) scale the data so that it has relatively the same variance.

Step 1 is accomplished by selecting a few emission regions for a metal which are most responsive to changes in concentration, temperature, etc., and relatively non-interfered with by other metals or the OH continuum. Traditionally, step 3 is handled with a logarithmic transformation and this has been found to be sufficient for the analysis here as well. For step 2, three transformation methods were selected for use in pilot studies. These transformations are given in Table 1. The appropriateness, and ultimate selection for use, of the three transformation methods was determined by a detailed empirical study.[9,11] The study isolated a metal and trained three RBFNNs to analyze the metal's spectrum. Each of the three networks trained used a different data transformation. The results found that the subinterval energy method gave the best performing network in light of all the external error sources.

Results and Discussion

The first step in the development of a RBFNN involves the creation of *n*-sets of input/output pairings. To do this, the three parameters for a chosen metal are randomly varied over specified ranges within the SPECTRA code and the resulting spectra are stored. The parameter ranges were set by combustion phenomenologists as the most likely to be observed during a firing of the SSME.[5] Table 2 details these parameters and their typical ranges appropriate for the SSME.

Transformation Method	Operator
Area Input	$\int \left(\vec{I}_n \right) d\lambda$
Maximum Intensity	$\max\left(\vec{I}_n \right)$
Subinterval Energy Method	$\left\Vert \vec{I}_n \right\Vert$

Table 1 Possible transformations for a given input vector.

Flow Field Parameter	Range
Mach Disc Temperature [K]	2500 - 3200
Number Density	1.0E09 - 5.0E14
Broadening Parameter	0.0 - 1.2

Table 2 SPECTRA flow field parameter ranges.

Keeping track of these combinations and the associated spectra produced provides the RBFNN with the necessary sequence of input/output pairings. The number of training sets to generate varies depending on the specific application. For the OPAD project, RBFNNs were trained with 200, 400, and 800 training sets. During training, it was observed that significant increases (on the order of hours) in training time and network complexity accompanied the use of large training sets. The problem was further augmented by the SPECTRA costs incurred: the creation and storage of training data. The constraints of the OPAD project demand real-time, low computational cost software that can be developed in a short time. Furthermore, adding additional training sets (such as the 600 and 800 runs) did not increase the accuracy of the predictions. For these reasons, only moderately large (on the order of 200) training sets were used in the development of RBFNNs. If these are uniformly generated across the parameter ranges, then enough information is provided to allow the RBFNNs to interpolate.

Once the RBFNNs are trained for the respective metals, the question then becomes, how good are the networks at accomplishing the task of predicting the number density, broadening parameter, and temperature of a particular metal from a given test-stand spectrum? Additional sets of test data can be created with the SPECTRA code and used to validate the RBFNNs. The data is easily obtained and can be quite useful for discovering trends and testing new ideas. As a validation agent, however, it does not provide a definitive statement as to the performance of a network within the "real world." To accomplish this, the SPECTRA code would require a complete modeling of the absorption phenomena, the OH continuum, variable wavelength dispersion effects and many other unforeseen physical influences. To use actual test-stand data for validation, however, would require knowledge of what is in the flow, (something that is unknown), so that comparisons can be made with the network's predictions. However, test-stand data can provide validation by other means as will be seen.

In January of 1996, the engine group at SSC was testing the new Pratt and Whitney SSME turbo-pump when an observer noticed a visible object being ejected from the engine accompanied by numerous significant emissive events. Immediate shutdown was ordered to preclude a possible engine failure. Post-test analysis found that the turbo-pump had failed. Further analysis of all the video tape footage, vibrational data, and other instrumentation identified five specific moments during the test where major anomalous events had occurred. Had these anomalies been detected ahead of time, the engine could have been shutdown and the destroyed engine components salvaged.

With this information, OPAD researchers at MSFC attempted to identify the same anomalies within the spectral data. Using the SPECTRA code, the researchers *iteratively* fit many of the spectral scans obtained from the SSC test stand and found major anomalous events in the spectra at precisely the same points in time as indicated by the other data sources.[7] It took the SPECTRA fitting routines two hours to fit a *single* spectral scan taken in time. There were over a thousand scans for the 500 second SSME test, thereby making the job extremely arduous.

As a validation of the RBFNNs, seven metals were selected by analysts at MSFC as the most important to monitor in terms of engine component makeup and erosive materials. They are chromium, aluminum, nickel, manganese, silver, cobalt, and iron. All the scans taken during the SSME failure at SSC were then given to RBFNNs trained to monitor the selected metals. It took the RBFNNs less than two minutes on a Silicon Graphics Indigo to make predictions on a thousand scans for the seven metals. The improvement of computational time was significant in comparison to the two hours required for the iterative fitting of a single scan using the SPECTRA routines.

Figure 6 gives the RBFNN predicted number density trace for each metal. For comparison, the major anomalous events found by the engine group test groups at MSFC and the Stennis Space Center via the alternate data sources (primarily vibrational and visual data) occurred at the test times given in Table 3. Figure 6 demonstrates that at every one of the points listed in Table 3, the RBFNN discovered an anomalous erosive event. (These events are annotated on Fig. 6). The 531 sec event marked the total failure of the SSME turbo-pump and the metallic emissions where much higher than those found in Fig. 6. Therefore, the plots were truncated prior to 531 seconds so the anomalies leading up to the failure could be viewed.

Event 1	**Event 2**	**Event 3**	**Event 4**	**Turbo-pump Failure**
130-131 [sec]	276 [sec]	283 [sec]	404-405 [sec]	531-554 [sec]

Table 3 Test times where the most erosive anomalous events occurred for the January 1996 test of an SSME at the Stennis Space Center.

The predictions of the RBFNNs duplicated previous conclusions drawn from all the other data sources and this was accomplished in under two minutes. Recall that SSME health monitoring via plume spectral assessment is predicated upon the following tripartite evaluation scheme: anomaly identification, quantification, and isolation. The results illustrated by Fig. 6 explicitly *validate* the neural network architecture developed herein as a real-time anomaly identifier.

The question of the RBFNN's ability to *quantify* the amount of metal in the spectrum will be addressed momentarily. As for anomaly isolation, such as identifying a failed component, this task would not be one for the RBFNN alone. Isolation requires the simultaneous use of the network predicted quantities and knowledge of the distribution of the various metals in the components throughout the SSME. To date, a database such as this has yet to be established.

In an attempt to validate the anomaly quantification ability of the RBFNNs, a formal investigation was undertaken. There are three main sources of error in the RBFNN estimations: 1) Uncertainties in the calibration of the spectrometers, 2) Uncertainties caused by the inability of the SPECTRA model to completely account for all the flow physics, and 3) Uncertainties in the inability of the RBFNN model to completely approximate the underlying physical function. The study is ongoing, but the initial results indicate approximate errors of 2% for the temperature prediction, 8% for the number density prediction, and 35% for the broadening parameter prediction.[12]

Further qualitative conclusions can be drawn by examining the spectral "fits" produced by the RBFNN predictions. To illustrate this, predictions for the seven metals of interest were taken from the 130 sec and 276 sec spectral events. There is not a preference of one event time over another; these two were merely selected to provide a qualitative measure of the results. Indeed, the results given here are representative of spectral fits found at other erosion events. RBFNN predictions were fed back into the SPECTRA code to obtain all the individual metal contributions to the spectrum. The resulting spectra were then compared against the *original* test-stand data scans and the wavelength regions indigenous to the various metals were isolated. Typical results of these actions are shown in Figs. 7-10. Additional spectral fits can be found in Ref. 9.

Looking at Figs. 7-10, it can be concluded that the RBFNNs are capable of quantifying metallic content in the plume. (Note that an "exact fit" would be when the predicted and test-stand data curves are identical.) The fits for nickel, iron, and chromium seem to be the best. Cobalt is nestled in-between a series of nickel peaks, but with closer examination it can be seen that the RBFNN's predictions matched very well with most of the cobalt peaks within the scan. The nickel prediction was slightly higher than that observed in the test-stand data; this could probably best be explained by the corruption of the nickel peaks by the neighboring cobalt lines.

A statement that can be made with certainty here is that the RBFNNs accomplished the anomaly *detection* task. As for the *quantification* task, initial uncertainty analysis along with many spectral fits, similar to the foregoing fits, have demonstrated an initial capacity of the RBFNNs to correctly predict metal properties in the SSME exhaust plume.

Conclusions

SSME health monitoring via plume spectral assessment is predicated upon the following tripartite evaluation scheme: anomaly identification, quantification, and isolation. Recognizing anomalies in the plume spectral scans and determining their severity in real-time situations can be computationally exhaustive. For this reason, neural network techniques have been investigated that accomplish these tasks in an expeditious manner.

The SPECTRA plume emulation code determines the specific plume spectral signature for a given metal composition and flow condition. The ultimate goal of the RBFNN architecture was to provide the "inverse" of this SPECTRA operation. More specifically, it was to discover a functional mapping allowing for the inference of the plume metallic state for a given spectrum. Using data sets created by the SPECTRA code, data mining procedures were incorporated that provided means for the selection of RBFNN inputs that compressed the computational space while still adequately representing the spectral information. Trained RBFNN architectures were then developed using this data. Results showed that the networks were capable of quantifying metals contained in the SSME plume.

To validate the RBFNN architecture developed, spectral data taken from the January 1996 SSME failure at NASA's Stennis Space Center were used. The RBFNNs were able to quickly identify anomalous events. Their ability to quantify has also been verified through uncertainty analyses and spectral fits with the test stand data.

Overall, the use of RBFNNs as real-time anomaly identifiers has been substantiated. They provide quick detection of anomalous erosions, identify which metals are eroding, and quantify the degree of erosion. This information could be used to isolate failing internal components, determine engine life, or obviate an imminent engine failure on any propulsive platform that contains some type of EM emission.

References

[1] Powers, W. T., Cooper, A. E., and Wallace, T. L., "OPAD Status Report: Investigation of SSME Component Erosion," SAE Paper 92-1030, 1992.

[2] Cikanek, H. A., and Powers, W. T., "Analysis of UV-Visible Spectral Radiation from SSME Plume," *Advanced Earth to Orbit Propulsion Technology - 1988*, NASA CP-30120, Vol. 2, 1988, pp.595 - 611.

[3] Madzsar, G. C., Bickford, R. L., and Duncan, D. B., "An Overview of In-Flight Plume Diagnostics for Rocket Engines," AIAA Paper 92-3785, 1992.

[4] Powers, W. T., Cooper, A. E., and Wallace, T. L., "Validation of UV-VIS Atomic Spectral Model For Quantitative Prediction of Number Density, Temperature, and Broadening Parameter," *Proceedings of the 1995 JANNAF Propulsion Systems Hazards Subcommittee*, Published by the Chemical Propulsion Information Agency, Columbia, MD, 1995.

[5] Wallace, T. L., Powers, W. T., and Cooper, A. E., "Simulation of UV Atomic Radiation for Application in Exhaust Plume Spectrometry," AIAA Paper 93-2512, 1993.

[6] Buntine, W. L., "OPAD Data Analysis," *Advanced Earth to Orbit Propulsion Technology - 1994*, NASA CP-3282, Vol. 1, 1994, pp. 168-177.

[7] Cooper, A. E., Powers, W. T., Wallace, T. L., Buntine, W., "Recent Results in the Analysis of Large Rocket Engine Anomalies Utilizing State-of-the-Art Spectral Modeling Algorithms," *Proceedings of the 1997 JANNAF Propulsion Systems Hazards Subcommittee*, Published by the Chemical Propulsion Information Agency, MD, 1997.

[8] M. J. L. Orr, "Regularisation in the Selection of Radial Basis Function Centres", *Neural Computation*, Vol. 7, No. 3, 1995, pp. 606-623.

[9] Benzing, D. A., "A Neural Network Approach to Space Shuttle Main Engine Health Monitoring Using Plume Spectra," Master of Science Thesis, Department of Aerospace Engineering and Mechanics, Univ. of Alabama, Tuscaloosa, AL, 1996.

[10] Arfken, G. B., and Weber, H. J., *Mathematical Methods for Physicists*, Academic Press, New York, 1995, pp. 191-280.

[11] Benzing, D. A., Whitaker, K. W., and Moore, D. C., "A Neural Network Approach to Anomaly Detection in Spectra," AIAA Paper 97-0223, 1997.

[12] Hopkins, R. C., and Benzing, D. A., "Uncertainty in Calibration, Detection, and Estimation of Metal Concentrations in Engine Plumes Using OPAD," Final Report - NASA/ASEE Summer Faculty Fellowship Program, Marshall Space Flight Center, August 1997, pp. 126-132.

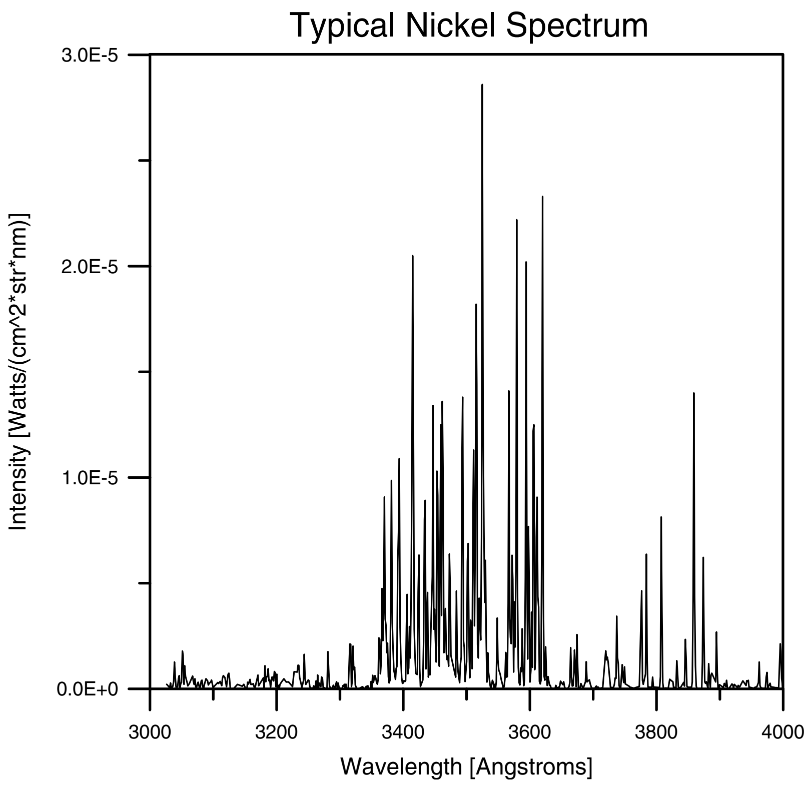


Fig. 1 Typical nickel emission spectrum

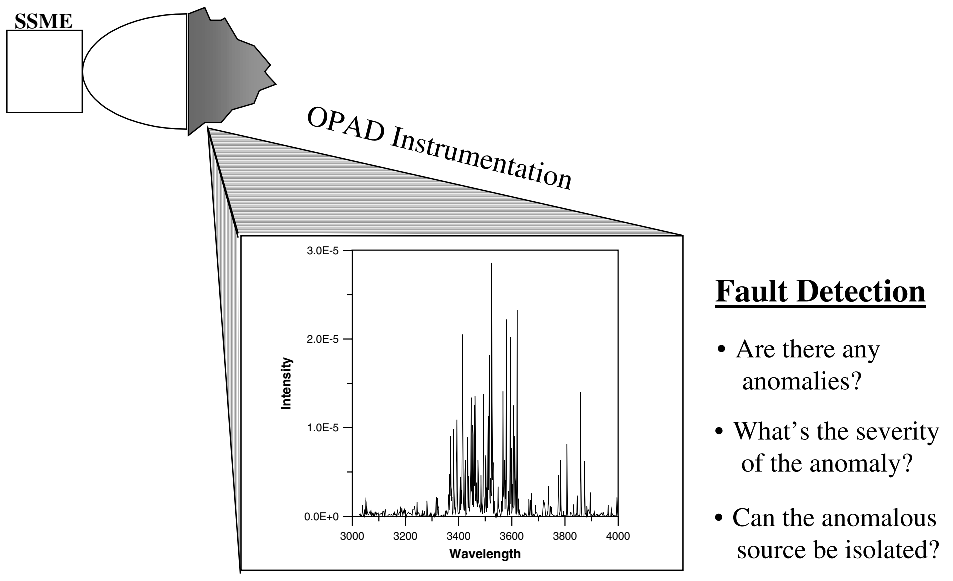


Fig. 2 The OPAD focus.

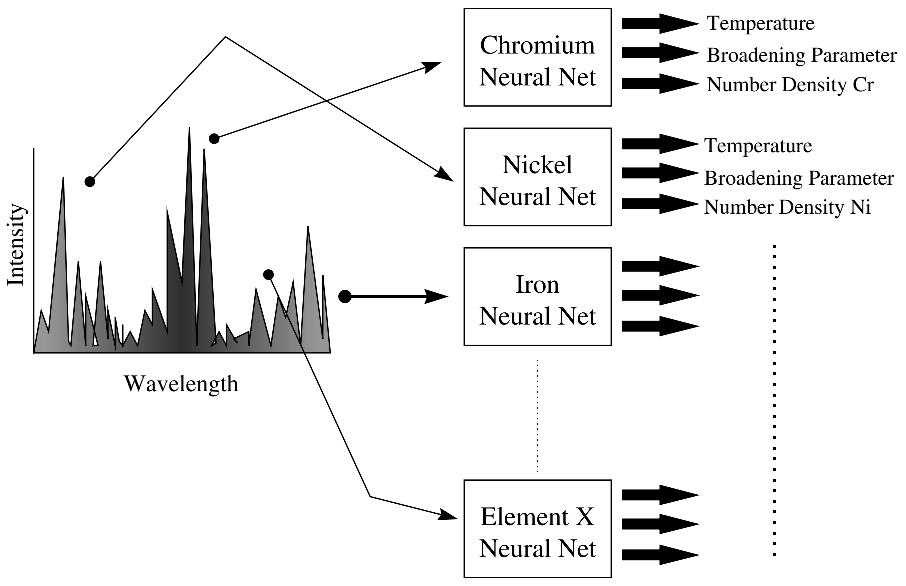


Fig. 3 Multiple neural network architecture.

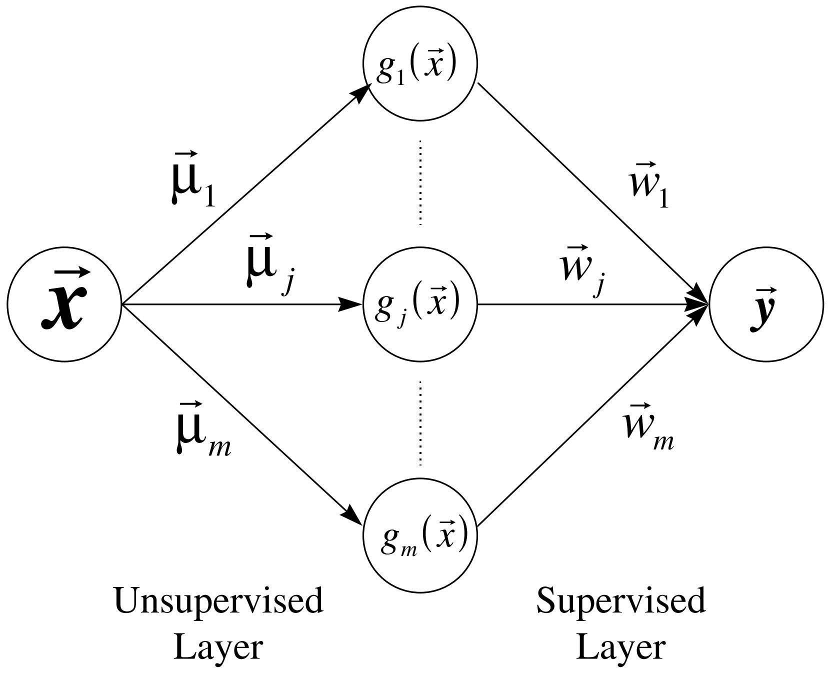


Fig. 4 RBF neural network architecture.

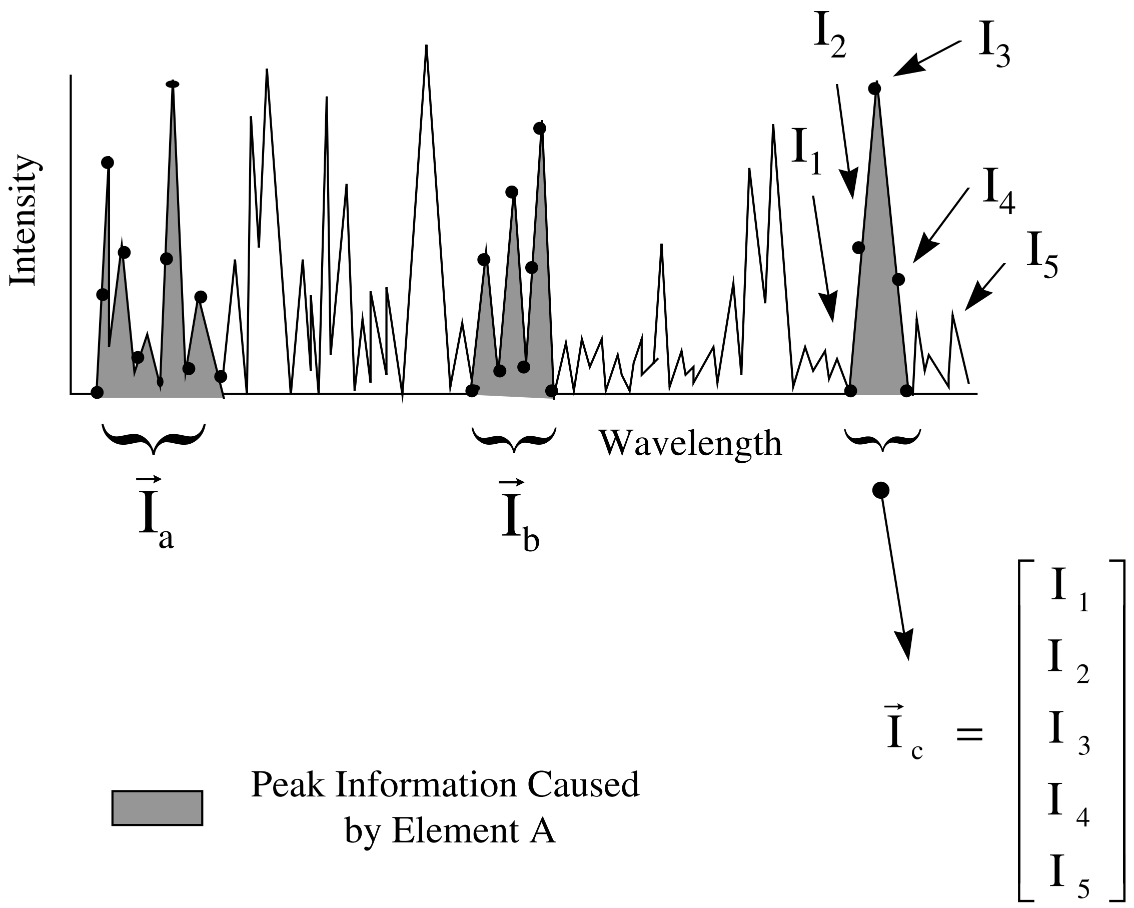

Intensity
Wavelength
I_1
I_2
I_3
I_4
I_5
$\vec{I}_a$
$\vec{I}_b$
$\vec{I}_c = \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \end{bmatrix}$
Peak Information Caused by Element A

Fig. 5 Network input feature extraction.

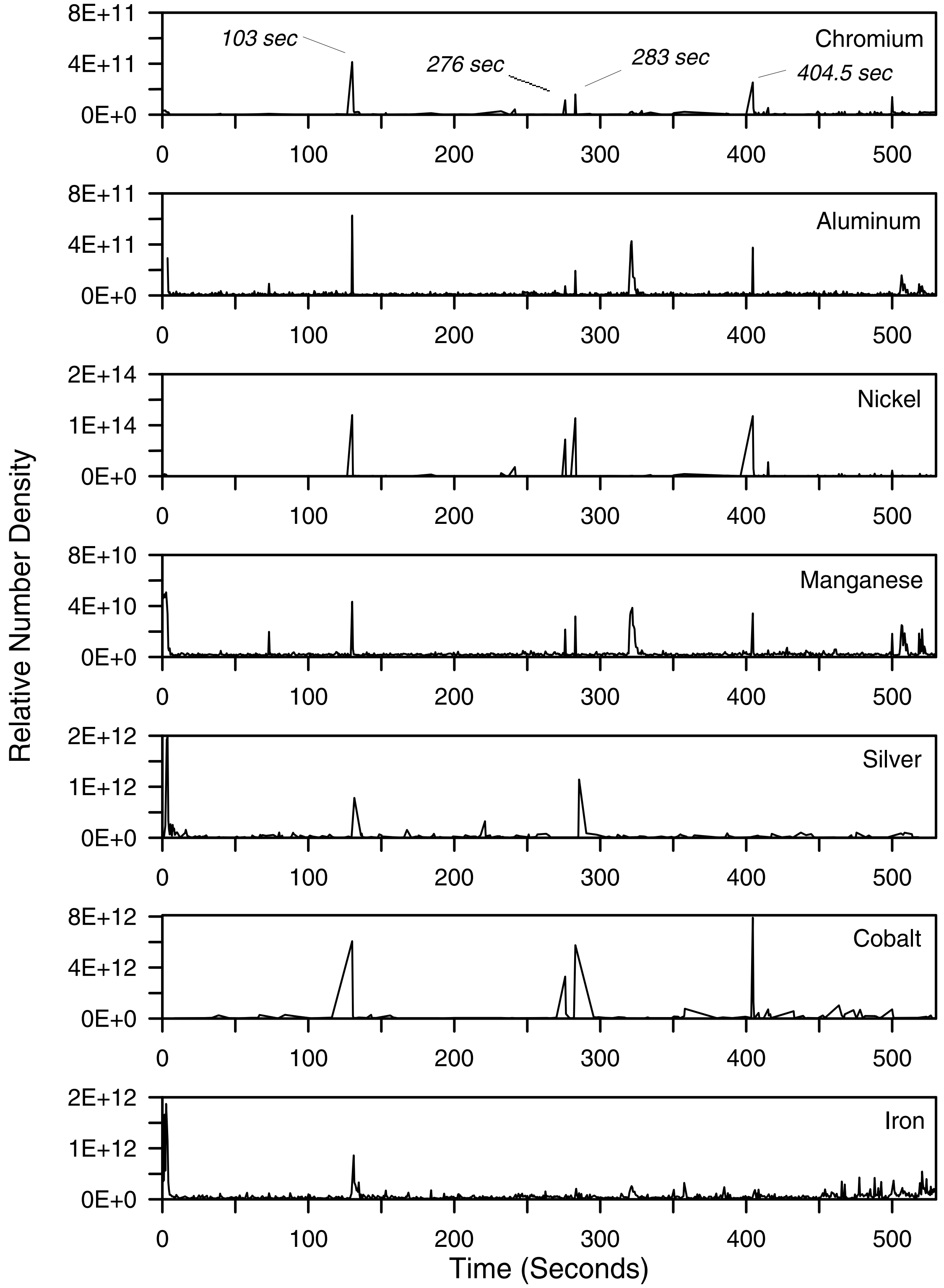


Fig. 6 Number density predictions versus test time (actual SSME test-stand data shown).

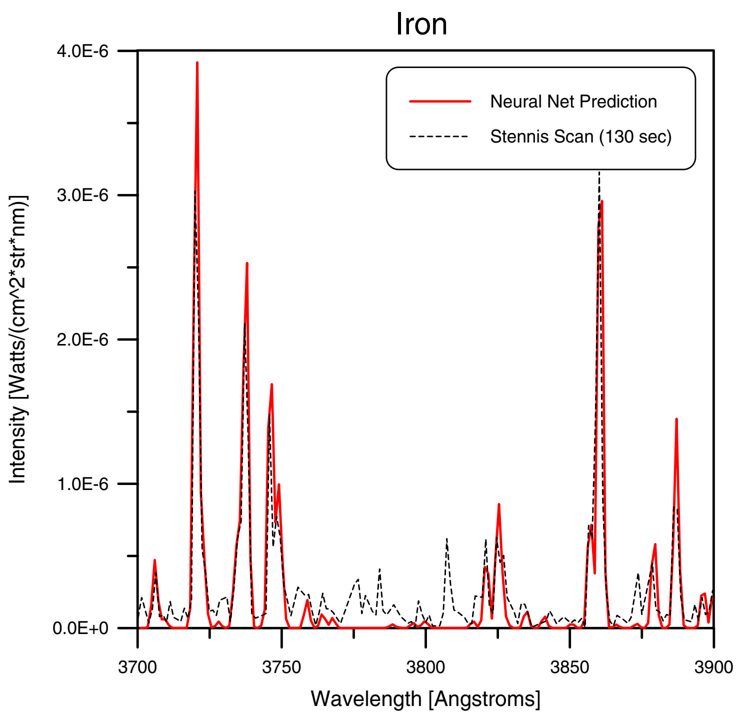


Fig. 7 Iron prediction for the 130 sec Stennis Space Center scan.

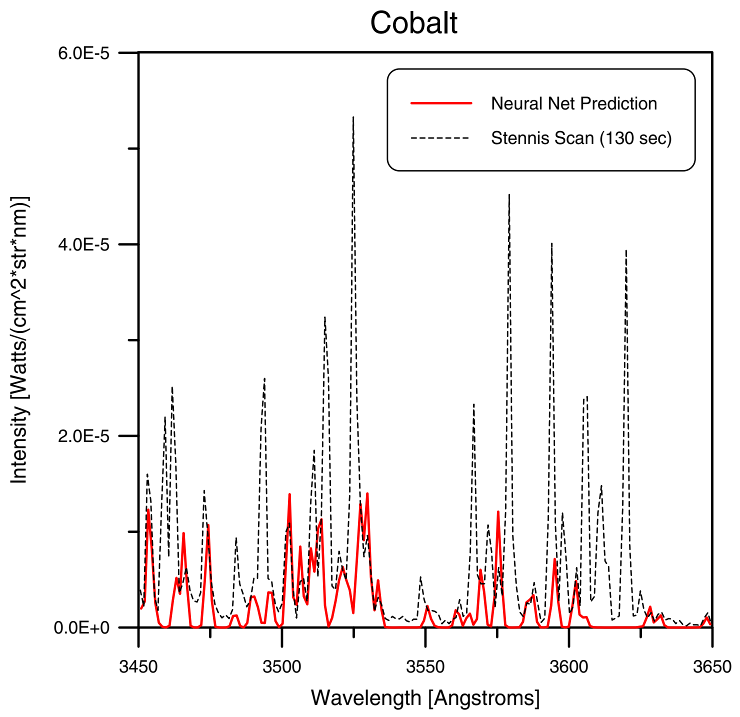


Fig. 8 Cobalt Prediction for the 130 sec Stennis Space Center scan.

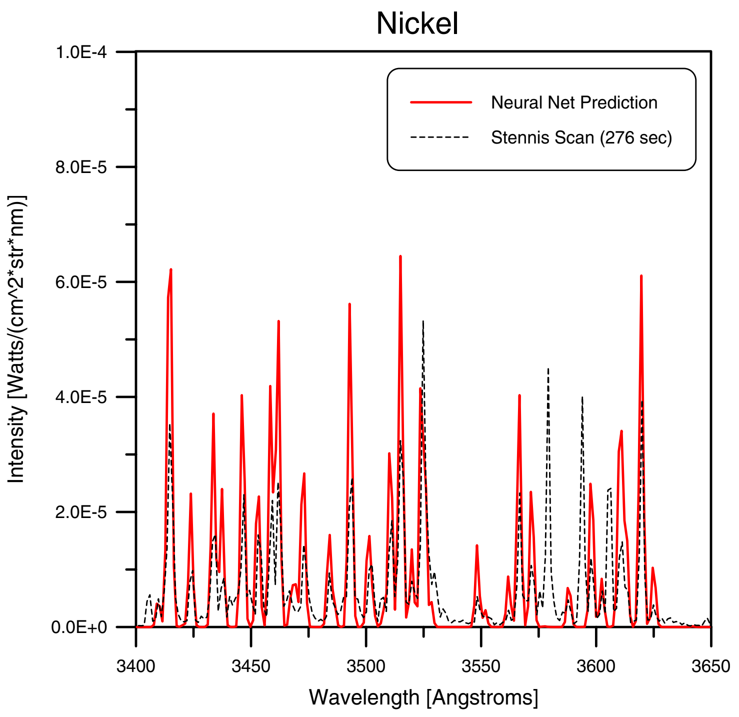


Fig. 9 Nickel Prediction for the 276 sec Stennis Space Center scan.

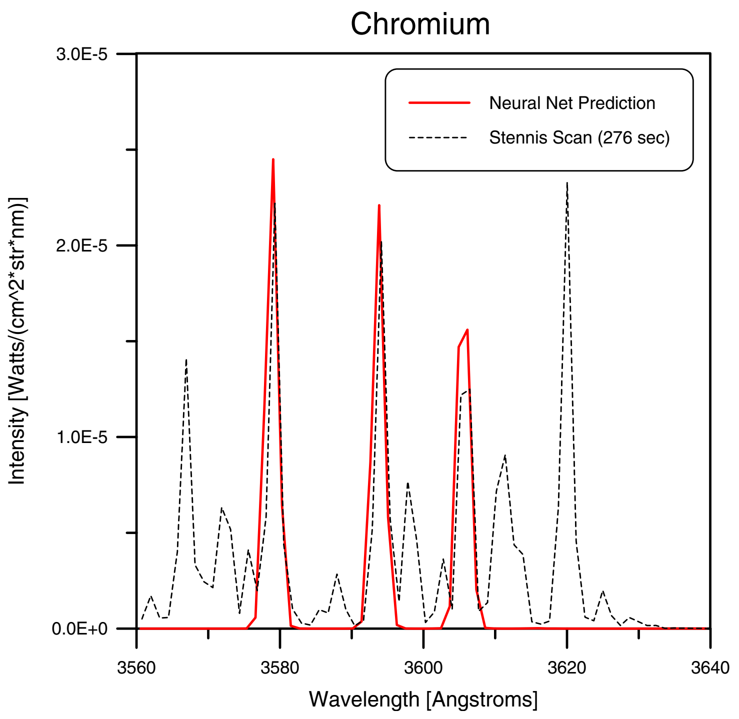


Fig. 10 Chromium Prediction for the 276 sec Stennis Space Center scan.

Challenging Aerospace Problems for Intelligent Systems

K. KrishnaKumar, J. Kanashige
NeuroEngineering Laboratory
NASA Ames Research Center
MS269-1, Moffett Field
CA 94035-1000, USA

kkumar, jkaneshige@mail.arc.nasa.gov

A. Satyadas (Full mailing address not provided)
IBM Corporation, USA.

antony_satyadas@us.ibm.com

1 Abstract

In this paper we highlight four problem domains that are well suited and challenging for intelligent system technologies. The problems are defined and an outline of a probable approach is presented. No attempt is made to define the problems as test cases. In other words, no data or set of equations that a user can code and get results are provided. The main idea behind this paper is to motivate intelligent system researchers to examine problems that will elevate intelligent system technologies and applications to a higher level.

2 Introduction

Intelligent System (IS) applications have gained popularity among aerospace professionals in the last decade due to the ease with which several of the IS tools can be implemented. The applications have gained popularity among both the technical and user communities for both intellectual curiosity and for practical reasons. Some of the novel ideas using IS include spacecraft autonomy, aircraft control, modeling, airfoil design, satellite operations, missile design, vehicle health management, and so on. In the next several sections, we present some problem domains that are challenging to achieve using intelligent system technologies. If successful, the ensuing IS approaches can revolutionize many aspects of aerospace applications. In each of the problem domains discussed, we first present the potential problem area and outline one IS configuration that could help achieve success. The problem domains covered include:

- Automated design
- Intelligent maneuvering
- Smart agent society
- Real-time optimization

3 Automated Design

3.1 The Problem

Aerospace systems of tomorrow will be complex and their design interdisciplinary. For example, design optimization conducted individually on subsystems such as wing, propulsion, and automatic control will not integrate without extensive and costly redesign. A unified design that integrates all facets of a system is difficult if not impossible to find using current optimization techniques. Traditional design approaches rely on cut-and-try approaches adopted by designers that can be conducted very well using high performance computers. The bottleneck for computer implementation is the lack of (1) a universal representation of the design features and (2) a procedure for the design features to be cut-and-tried *by a computer* in an optimal way. There are other potential problems associated with computer-based automated design. These are:

- Time required for a system simulation is large prohibiting use of full fidelity simulation of the problem.
- Analytical derivatives of the objectives of design are frequently unavailable and numerical gradients are expensive to compute.
- Design space consists of both continuous and/or discrete parameters
- Design response surface is non-linear, discontinuous, or undefined in some regions. Several local extrema is common in many applications.
- Initial guesses for the design are costly and might lead to inaccessible solution spaces.

The desire here is to use intelligent system technologies in an unified way to arrive a design framework by which automated design can be achieved using computers.

3.2 A Solution

The idea of letting the computer do the cut-and-try is not new to the world of optimization. A genetic optimization search basically involves a cut-and-try approach that is driven by the survival-of-the-fittest concept. If universal representations (standardized representation) can be developed, then a technique like genetic algorithms could be used. The other challenge is the speed at which designs could be evaluated. This implies that modeling techniques such as neural networks, fuzzy systems and so on can play an important role here.

To obtain a universal representation, we introduce the concept of a design building block. A design building block is a way to represent a feature as an input-output function with tunable parameters. These functions can be polynomials, pieces of a neural network, look-up tables, etc. These building blocks are identified first and a library of these building blocks is constructed. This library is continuously updated as more and more designs are created. This library is then used to construct design solutions for existing problems using variants of a genetic algorithm or other combinatorial optimization problem.

We see several critical benefits of this approach to the design community. These are:

1. Library of Design building blocks: Library techniques are commonly used in modern integrated circuit (IC) design packages. This greatly reduces the number of combinations with which the designer must deal, thereby speeding up the

search process. The design building blocks concept tailors this idea to the specific design.

2. Innovation is achieved by the use of evolutionary search with stochastic operators. This greatly reduces the chance of reaching a local solution and solves many of the problems associated with gradient search techniques.
3. Long-term memory can be provided to the design package to remember good design solutions for later use via the use of the micro-features of the immune system, a variant of a genetic search process [1-3].

In summary, one needs the following to achieve automated design:

- an universal representation scheme for representing the building blocks of the design
- an appropriate genetic coding to accommodate the building block concept
- a class of performance criteria and utility functions

Figure 1 presents the overall architecture of the system proposed.

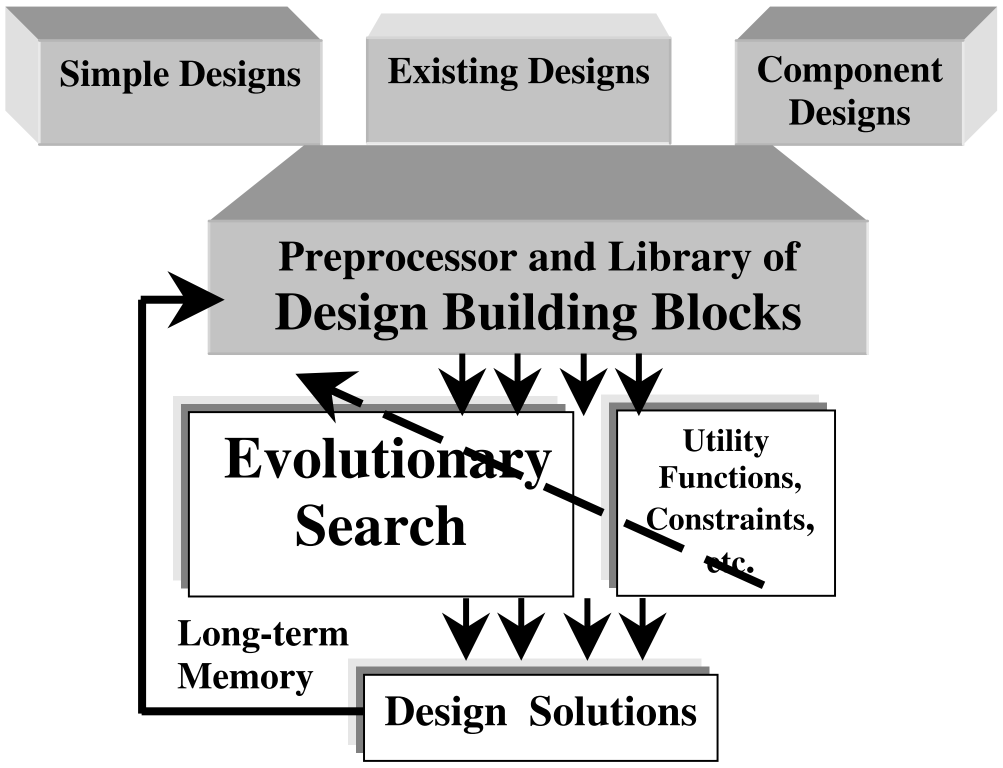


Figure 1. A Concept Diagram for Immunized Design Optimization.

Simple Design Solutions: Many engineering problems have simplified models and solutions that are routinely used as a first guess approximation of the complete solution. These solutions could be a rule-of-thumb solution or mathematically optimized using simpler models. These solutions are then converted to design building blocks to be used in the evolutionary search. Examples include:

- Linearized models and linear solutions for non-linear control problems.
- Knowledge base of operator rules, etc.

<u>Existing Design Solutions:</u> Many designs are in a way an improvement over existing designs. Although this is not true across the whole spectrum, at least the new designs use subsets of the old system components. These existing designs can then be converted to design building blocks and used in the search. Examples include:

- Available airfoil shapes for aerodynamic shape optimization
- Available nozzle shapes for Nozzle design.

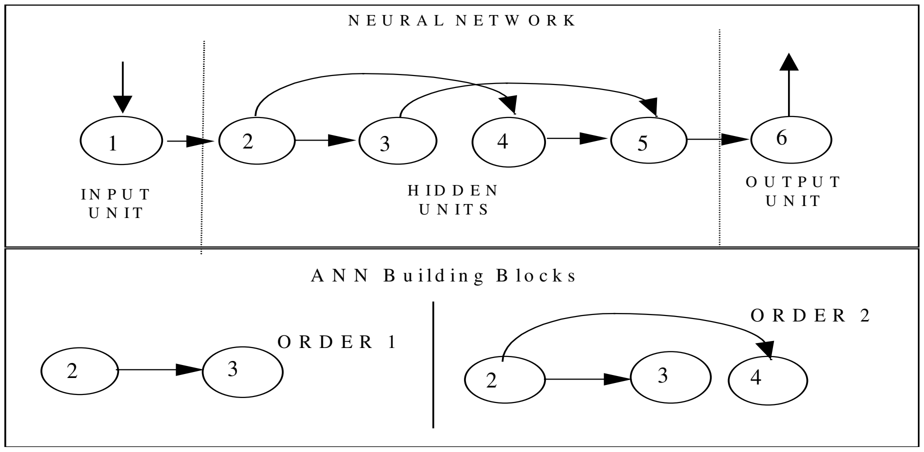
<u>Component Design Solutions:</u> When sub-components interact, the design solutions are easier if they are considered as non-interacting. These design solutions will not be the optimal solutions when interactions are considered but provide a means of arriving at certain fundamental building blocks. Examples include:

- In simultaneous actuator-sensor placement and control system optimization problem, one can design the placement separate from control system design.
- Another example is in aerodynamic-propulsion interaction in all-attitude control problems with thrust vectoring. Once again, the aerodynamic control system can be designed independent of the propulsion control system and converted to design building blocks.

<u>Library of Design Building Blocks:</u> Design building blocks are defined as segments of a design solution that contribute in establishing a good fit to the requirements of the design. Design Building blocks can be of different order. The order of a building block specifies the number of specific modules (or parts or some pre-defined configurations) in a building block. Determining important building blocks is problem dependent and both *a priori* knowledge and the ability to identify through genetic search will be useful here. Once the building blocks are identified, certain preprocessing and statistical analysis can be carried out to enable a faster search.

Next, we present two examples of design building blocks we have defined in our earlier studies [1-3].

- **Neural Network Building Blocks:** Examples of building blocks using neural connections are shown below. The building blocks are specified using a representation scheme that uses a neuron as the basic processing element. Thus the *Order 1* building block consists of two neurons and the relationship between them. In the case of Neural Networks, the relationship is the connection strength and the neurons are characterized by their type (input, hidden, output), the type of aggregation, and activation functions.



- **Aerodynamic shape optimization Building Blocks:** For airfoil optimization, satisfactory results have been obtained using an array of Bezier curves for the definition of the parameterized shapes. A Bezier polynomial of order n is defined by
  $$b(t) = \sum_{i=0}^{n} P_i \Phi_i(t)$$
  where
  $$\Phi_i(t) = \left(\frac{n!}{i!(n-1)!} \right) t^i (1-t)^{n-1}$$
  P_i's ar the vertices of the Bezier control polygon. Given the above parameterization, the design building blocks can be represented by P_i or a family of P_i's. Similar to neural network building blocks, *Order 1* building block will be just one P_i, *Order 2* will be two P_i's and so on.

Utility Functions, etc. To evaluate the designs produced by the genetic search, a series of performance measures and constraints have to be defined. Due to the ability of evolutionary algorithms to search through non-continuous spaces, both traditional and nontraditional measures could be easily combined.

4 Intelligent Maneuvering

Over the past several years, various adaptive control techniques have been developed which are capable of accommodating a wide range of damage or failure conditions [4-6]. These approaches apply techniques, such as neural networks, fuzzy logic, and parameter identification, to improve aircraft stability and control under varying conditions. While these approaches address the continuous-time aspects of "how to control" an aircraft, they do not address the discrete-time strategic and tactical decision-making aspects of "how to fly" an aircraft. The outer-loop control and flight planning portions of flight are normally left to conventional autopilots and waypoint-following flight management systems.

4.1 The Problem

Current levels of automation allow pilots to assign direct tasks to automatic systems, such as autopilots and flight management systems. These automated systems have been used in commercial aircraft for a number of years. While their design can incorporate many aspects of a pilot's experience, they do not possess the reasoning or learning abilities of a pilot. As a result, pilots are still responsible for supervising the performance of these systems as well as providing direction in the event of required changes. By applying intelligent methods of automation, pilots, ground-based operators, or autonomous executives can defer the responsibilities from performing and supervising tasks, to focus on managing goals and objectives.

4.2 A Solution

In order to make reliable decisions when flying aircraft under varying conditions, intelligent technologies must be applied which are capable of responding to changing goals and objectives, while taking correction actions in the presence of internal and external events. Multiple methods, such as heuristics, artificial intelligence concepts, and other soft computing techniques, can be applied in order to replace the experience, reasoning and learning abilities of pilots. Just as pilots use different mental approaches when performing various tasks, different intelligent automation techniques can be applied which correspond to the computational nature and time restrictions associated with those tasks. One method of organizing various intelligent techniques is to establish a tiered architecture, with separate deliberative and reactive decision-making layers.

System Architecture: In terms of achieving a flight-path goal, a pilot's behavior can be captured through a layered model consisting of discrete-time strategic planning and tactical maneuvering, and continuous-time manual control (Figure 2) [7]. The discrete nature of strategic and tactical behaviors allows for automated decision-making techniques to be applied. Furthermore since strategic planning decisions are less time-critical, more computationally intensive approaches can be utilized. All of the real-time processing elements can be isolated in the automation of manual control.

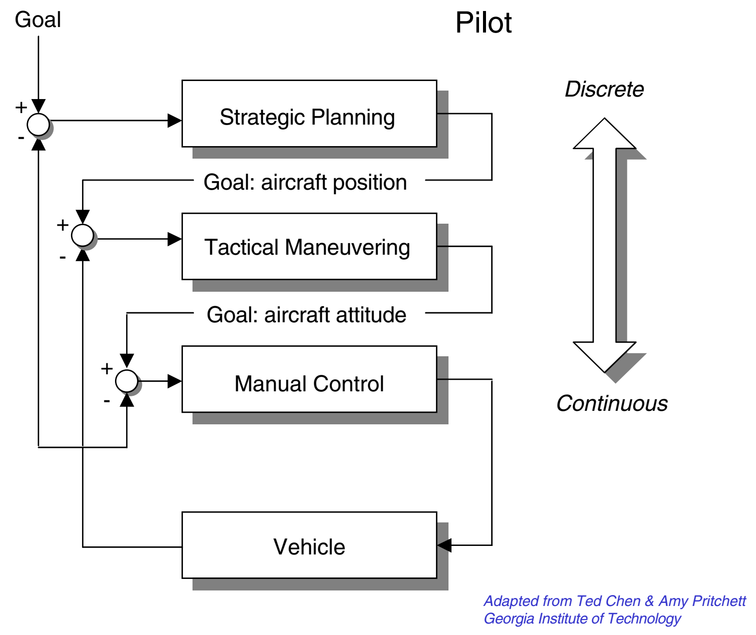


Figure 2. Pilot Behavior Hierarchy

Figure 3 shows the resulting conceptual integrated architecture. Strategic technologies would perform longer-term flight planning, in order to meet dynamic mission goals and objectives, while avoiding obstacles and staying within performance boundaries. Tactical technologies would perform time-critical flight path operations, including aggressive maneuvers in the presence of unexpected obstacles. Conventional and adaptive control techniques would be used to automate the manual control task of the pilot, through the automated selection of flight modes and targets.

Strategic Planning: From a pilot's point of view, any flight can be thought of as a plan of turns, descents, and other discrete actions. These actions alter the continuous flight-path or aircraft trajectory, until desired goals are reached. Actions are not limited to merely changes in the aircraft speed and orientation. Some actions also change the aircraft configuration itself, such as extension of flaps and gears or the dumping of excess fuel. Various trajectory specialists could be used to produce candidate flight path segments, representing the "experience" of a pilot. The selection criteria would be based on mission goals and constraints provided by a pilot, ground-based operator, or autonomous executive.

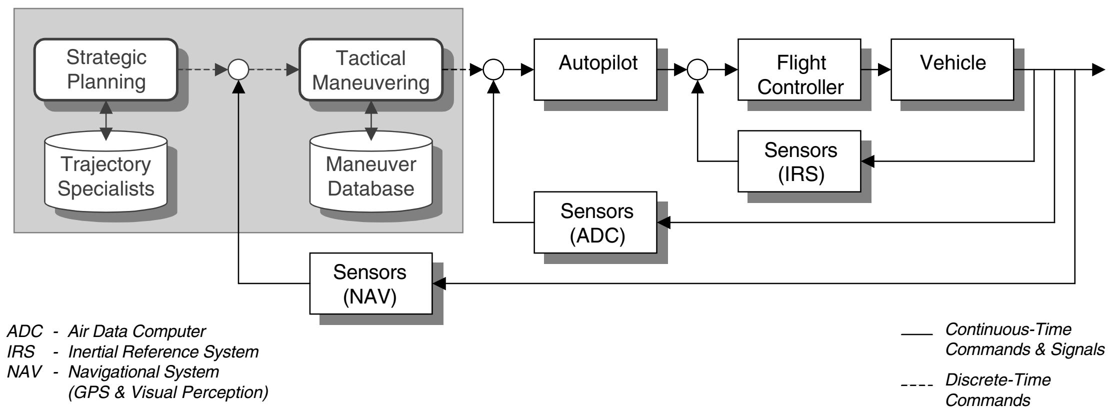


Figure 3. A Concept Diagram for an Integrated Architecture

<u>Tactical Maneuvering:</u> At the highest level, the pilot compares the commanded flight-path with that of the current aircraft and selects the maneuvers capable of achieving that command. Pilots use their knowledge of aircraft capabilities and of near-optimal maneuvering strategies in order to select the necessary actions. Various methods could be used to select the necessary maneuvers. A maneuver database, also representing the "experience" of a pilot, could be used to provide pre-canned, or automatically generated, maneuvering elements and established sequences. Vehicle models can be used to provide the necessary predictive information for decision-making, representing the equivalent of a pilot's "understanding" of the internal performance of the aircraft. Appropriate flight modes and targets would be sent to the autopilot system, when necessary to initiate the desired actions.

<u>Automatic Controls:</u> Conventional control techniques can be used to automate the continuous-time control task of the pilot. However, various adaptive control techniques can also be used to provide the "learning" ability of a pilot. These techniques have the potential of improving handing qualities, and thereby increasing the accuracy of simplified closed-loop models used for decision-making.

5 Society of Prediction Agents

Prediction or forecasting is concerned with using the knowledge of present and past events to make calculated estimates of future events. Prediction is an universal phenomenon used in low level cognitive tasks such as vision, and perception, and high level cognitive tasks such as planning and making inferences. The inherent subjectivity, randomness, domain dependencies, and uncertainties makes the prediction problem extremely difficult and challenging to the scientist and the engineer. Moreover, often there are multiple sources of prediction with varying degrees of reliability and confidence [8,10,11,12,14]. There are two distinct prediction techniques, namely qualitative and quantitative. Qualitative techniques are referred to as judgmental, technological, or non-statistical and usually depend on expert opinion. Examples of such techniques include decision matrices, S-curves, game theory, systems analysis, Delphi method (jury of executive opinion method), and more recent fuzzy decision support expert systems [8]. Quantitative prediction techniques are based on the assumption of historical continuity. Options include time series, regression, and combinations of the two.

5.1 The Problem

In aeronautics applications prediction techniques vary from a simple need to predict an aerodynamic derivative to more complex situation such as fault predictions. In space applications, prediction applications are more complex. Since the information sources can be very varied, it is not sufficient to provide just human-like characteristics atop the prediction technologies, we need to a go a step further and provide the capability of a group of experts to arrive at a prediction. This task is quite challenging.

5.2 A Solution

More recent work related to a group prediction includes the Fuzzy Multiple Criteria Group Decision Making (FMCGDM) based prediction proposed by Satyadas [7] that introduces the notion of a group of expert agents selecting the appropriate prediction from a pool of predictions. This is a hybrid model with a collaboration framework at the highest level and various multi-sources of prediction at lower levels.

The FMCGDM equations can be defined as follows: Given **n** fuzzy rules

$R= [r_1, r_2,......., r_n]$, where

r_n= IF x_{n1} AND/OR x_{n2} AND/OR x_{nj} THEN y_{n1} AND $y_{n2}...y_{nk}$., each consisting of

j inputs/states/antecedents/premises $X=[x_1, x_2,\dots x_j]$,
k outputs/actions/consequents/conclusions $Y=[y_1,y_2, ... y_k]$,

an universe of discourse resolution of *p* steps; the state vector

$S = [s_1, s_2, ...s_n]$, where

$s_n = x_{n1}$ AND/OR x_{n2} AND/OR...x_{nj} and the fuzzy action curve set

$A=[a_1,a_2,....a_n]$, where

$a_n = [z_{n1}, z_{n2}, ...z_{np}]$

is obtained by applying fuzzy implication on S over the universe of discourse. Appropriate fuzzy aggregation (example: Max) and defuzzification (example: Center of Area) algorithms can be applied on A to obtain a crisp output. The fuzzy membership function parameters and the rule structure can be learned using evolutionary algorithms. A weight matrix *W[j+k,i]* may be used to capture the weightage provided by *i* members of the group on the *j+k* criteria, a weight matrix *X[n,i]* may be used to identify the importance of each rule. Techniques such as logarithmic regression or Saaty's AHP may be employed to apply the weights on to the fuzzy rules.

Given *u* sources of prediction, the above-described fuzzy system will provide a measure of importance for each of the prediction as its output. Appropriate ranking and selection techniques have to be used to compute the final prediction. The following technology issues need to be addressed:

1. Given that there are achievable multi dimensional levels of Smart Prediction Agents, what are the best set of building blocks that will enable easy implementation of these Predictors.
2. Once these building blocks are defined and developed, develop a set of intelligent prediction algorithms that integrate these building blocks. These shall be uniquely indexed using the Smart Prediction Agent (SPA) rating given in Table 1. The relevance and value of the algorithms are context dependent.
3. The implementation challenge will be to ensure component-based architecture and standards that will allow interoperability and meets non-functional requirements such as scalability, security, reliability, and the like.

The multi dimensional levels of smart prediction presented in [8] (See Table 1 below) provides a means for qualifying and quantifying smart prediction techniques. We believe that a practical way to define the predictive capabilities of a system is to approach it as having multi dimensional levels of capabilities for self-improvement, problem solving, knowledge and domain bounds, and trade-offs. The capabilities are additive towards higher levels.

Any smart prediction agent may be identified as SPA[*p,i,k,s,r*] where *p*=problem solving, *i*=self-improvement, *k*=knowledge building, *s*=severity/domain bounds, and *r*=risk management/trade-off. This enables a SPA rating for various prediction systems based on the values of *p, i, k, s*, and *r*.

Table 1. Levels of Dimensions for Smart Prediction

Level	Problem Solving	Self-Improvement	Knowledge Building	Severity	Risk Management/ Trade Offs
0	Uni-variate	Statistical	A priori	Crisp compute	Accuracy/Trend
1	Multi-Variate	Generalized	Derived	Imprecise	Local/distributed
2	Patterns/Cases	Adaptive	Inferred	Incomplete	Temporal
3	Stationarity	Optimized	Discovered	Subjective	Spatial
4	Multi-source	Plan		Complex	Static/Dynamic
5	Causality			Chaotic	Maximum/Avg

<u>Problem Solving Dimension for Smart Prediction</u>

Six levels have been identified in this dimension (Table 1). The uni-variate and multi-variate refers to the dependent variables. Patterns/cases with the associated behavior present a higher difficulty level for problem solving. Stationarity and ergodicity issues present the next level of challenge. A stationary process will have time-invariant mean and variance. The covariance between values of the process at two time points will depend only on the distance between these time points and not on time itself. The ergodicity assumption requires that values of the process sufficiently far apart in time are almost un-correlated so that averaging a time series through time continually adds new and useful information to the average. This is followed by multiple sources of prediction that includes the system approach.

Self-Improvement Dimension for Smart Prediction

The levels identified have been motivated by the levels of intelligent control [9]. Level 0 relies on statistical information to improve the quality of prediction. The next level requires generalization capabilities that can be achieved using computational intelligence techniques such as artificial neural networks. This is followed by adaptive learning capabilities. Optimality of prediction, with the goal of minimization or maximization of a utility function over time, is addressed by Level 3. Coupling to a control module, that may require modified predictions, is introduced in the next level. Level 4 presents the planning aspects of self-improvement that provides the ability to perform predictions based on a plan with the associated goals and risk mitigation strategies.

Knowledge Building Dimension for Smart Prediction

Level 0 relies on a priori knowledge. Various techniques, from ad-hoc to well defined and complex, from artificial intelligence have been proposed by researchers. Techniques include memory modules (neural and otherwise) with fixed weights, fixed fuzzy rules and parameters, and suitable parameters for GA. The generic coding structure of GA is an attractive feature. Simple derivations based on observed facts are provided in Level 1. The next level has inference capabilities. Level 3 provide knowledge discovery capabilities that demand advanced data and text mining features. Learning plays a critical role here.

Severity Dimensions for Smart Prediction

The severity dimension provides a measure of the domain bounds. The levels start from crisp computation, imprecise and/or incomplete domains, and subjectivity. Level 4 represents a complex domain that is in the edge of chaos, and level 5 - a chaotic domain.

Risk Management Dimension for Smart Prediction

This dimension address trade-off that promotes risk management. Level 0 trade-off is between accurate prediction and just being able to predict the trend (whether it is upward or downward). The local/distributed aspect of the problem/solution space is introduced in Level 1. This is followed by temporal and spatial trade-off. Level 4 mitigates the risks between static and dynamic prediction. Choices between optimizing predicted values and just obtaining an average prediction is provided in Level 5.

We believe some of the challenges include:

- Challenges in introducing the upper layer that brings together prediction, control, and other capabilities.
- Challenges in cognition that will be required to be solved.
- Other aspects of the aircraft and linkages with the base - where adding smartness is valuable?

6 Real-time optimization under uncertainty

6.1 The Problem

Building large-scale intelligent solutions for optimal decision-making and control is of great importance to the advancement of operational autonomous systems. Werbos[15] has addressed an important formulation for achieving higher-order intelligence in these systems. The outlined approach extends the capabilities of adaptive critics[1] [16-18] to include temporal chunking. Although many researchers believe that real-time optimization over time and under uncertainty is a good approach to replicating higher order intelligence, the current applications of intelligent systems do not reflect the type of complexity processed by the mammalian brain.

6.2 The Solution

The central theme of the proposed solution is to arrive at a modular architecture for achieving higher order intelligent decision-making and control. To achieve this, we first define a MAC (Model-Action-Critic) neuron and extend it to represent a network of these with associated training signals. For effective autonomous decision making, we need adaptive control driven by an adaptive critic.

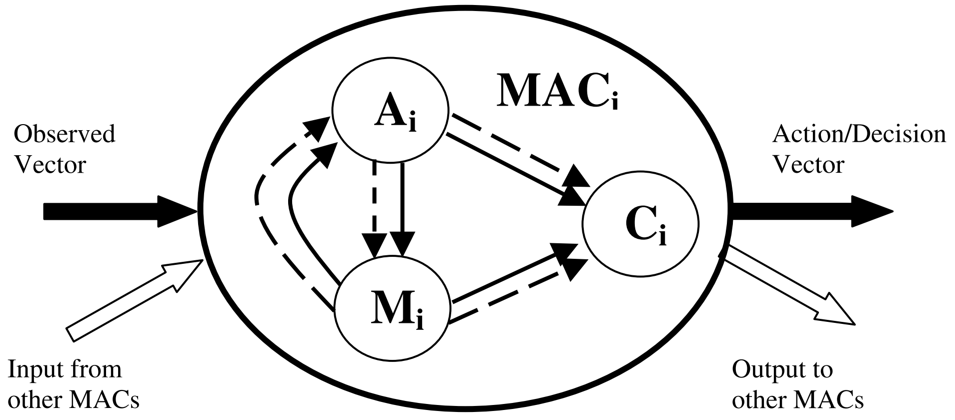


Figure 4. Model-Action-Critic module represented as one computational entity

In Figure 4, we present a single MAC (Model-Action-Critic) neuron in which the interactions within three sub modules and with the external world (fat solid arrows) and other MACs (fat hollow arrows) are shown. These inputs are available to all the three modules (M, A, and C). The dashed-lines and the solid lines between the three sub modules denote information being passed for different time periods to keep in mind the temporal nature of the training.

[1] An adaptive critic adapts itself and at the same time it outputs a performance measure that can be used to update a controller network and/or a decision network. Howard's formulation of dynamic programming [19] is the inspiration behind the simplest version of adaptive critics, namely, Heuristic Dynamic Programming (HDP) critics.

$$J(x_t) = \underset{u_t}{Max}\left\{ U_{PM}(x_t) + \gamma J(x_{t+1}) \middle| \; x_{t+1} = f(x_t, u_t, noise) \right\}$$

where x_t is the state vector, u_t is the control vector, $U_{PM}(.)$ is the one stage Performance Measure function, *f(.,.,.)* is the model of the system, and $\gamma(0 < \gamma \le 1)$ is the discount factor.

Once we identify MAC as one computational entity, network of these critics can be built by specifying either the partitioning before hand (an example is given later) or defining the desired hierarchy (number of MAC neurons, number of layers, etc.). Once these decisions have been taken, the MAC neurons can be assembled as a network of critics.

<u>Temporal Chunking for Brain-like Intelligence</u>
Given a policy π (control or decision strategy) and a probability transition matrix $\underline{\underline{P}}$, one can write the Howard's formulation of the recursive equation as:

(2) $\underline{J}^{\pi} = \underline{U}^{\pi} + ((\underline{\underline{P}}^{\pi})^{T}/(1+r))(\underline{J}^{\pi})$

Now if substitute $\underline{\underline{M}}^{\pi} = (\underline{\underline{P}}^{\pi})^{T}/(1+r)$, we have

(3) $\underline{J}^{\pi} = \underline{U}^{\pi} + (\underline{\underline{M}}^{\pi}\underline{J}^{\pi})$

The above equation represents a fixed policy with just value updates. In a policy update, the control *u* (or decision *d*) is chosen to optimize (maximize or minimize) the right hand side of the equation.

In equation 3, if the matrix $\underline{\underline{M}}^{\pi}$ is sparse (which is usually the case), one can partition the state space into smaller blocks within which the transition probabilities are non-zero and transition from one block to the other happens only for exit and entry states. This partition could be defined *a priori* or learnt on-line (which is a much more difficult task). In some problems, as shown later, such partitions are well understood.

Now let us say we have two partitioned blocks *A* and *B*, with *A* representing the current block in which the system is operating and *B* representing a block to which the system could transition to. Starting from Block *A*, we have two transition probability matrices, P^A representing transition probability within block *A* and P^{AB} the probability of transition into block *B*. Similar to *M* defined earlier, one can define M^A and M^{AB} matrices and derive an equation similar to equation 3,

(5) $\underline{J}^{\pi}\big|_A = \underline{U}^{\pi}\big|_A + \underline{\underline{M}}^{A}\underline{J}^{\pi}\big|_A + \underline{\underline{M}}^{AB}(\underline{J}^{\pi}\big|_B)$

In the above equation, the notation $\underline{V}\big|_A$ represents a portion of the variable vector $\underline{V}$ that applies to states within block *A*.

If we now extend the partition B to include more blocks, we have

(6) $\underline{J}^{\pi}\big|_A = \underline{U}^{\pi}\big|_A + \underline{\underline{M}}^{A}\underline{J}^{\pi}\big|_A + \sum_{B\in n(A)} \underline{\underline{M}}^{AB}(\underline{J}^{\pi}\big|_B)$

where $B \in n(A)$ represents the blocks in *B* that can be transitioned into by states in block *A*.

As shown by Werbos[15], the above equation can be written as,

(7) $$\underline{J}^{\pi}\Big|_A = \underline{J}^A + \sum_{B\in n(A)} \underline{\underline{J}}^{AB}(\underline{J}^{\pi}\Big|_B)$$

with $\underline{J}^A = (\underline{\underline{I}} - \underline{\underline{M}}^A)^{-1}(\underline{U}^{\pi}\Big|_A)$ and $\underline{J}^{AB} = (\underline{\underline{I}} - \underline{\underline{M}}^A)^{-1}\underline{\underline{M}}^{AB}$

Now one can write the following recursive relations similar to Bellman's

(8) $\underline{J}^A = \underline{U}^{\pi}\Big|_A + \underline{\underline{M}}^A \underline{J}^A$ and $\underline{J}^{AB} = \underline{\underline{M}}^{AB} + \underline{\underline{M}}^A \underline{J}^{AB}$

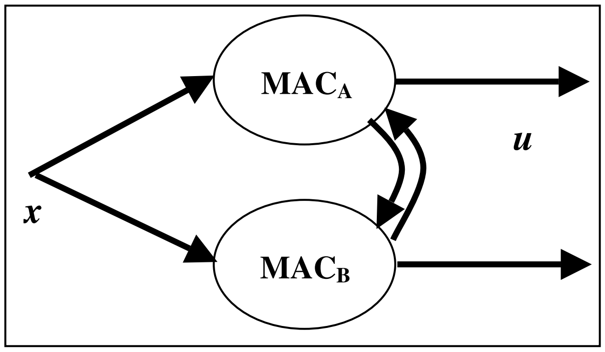
Using equations 8a and 8b in conjunction with a learning algorithm, $\underline{J}^A$ *and* $\underline{J}^{AB}$ can be updated and thus $\underline{J}$ can be calculated using equation 7. Finally, control (or decision) can be updated based on $\underline{J}$.

<u>A simple example:</u> We will define a simple example to illustrate the use of the idea presented. Let a system consist of two states and one control as shown below.

$$\dot{x} = f_A(x_1, x_2, u) \quad for \quad x_{21} \le x_2 \le x_{22}$$
$$\dot{x} = f_B(x_1, x_2, u) \quad for \quad x_{22} < x_2 \le x_{23}$$

where $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, $f_A = \begin{bmatrix} f_{A1} \\ f_{A2} \end{bmatrix}$, $f_B = \begin{bmatrix} f_{B1} \\ f_{B2} \end{bmatrix}$,

and u is the control.



Note that the system has a clear partition based on the state x_2. Based on this pre-defined partition, one can design MAC$_A$ partition $(x_{21} \le x_2 \le x_{22})$ and MAC$_B$ partition $(x_{22} \le x_2 \le x_{23})$ and can be implemented as shown to the right.

<u>Learning the underlying hierarchy:</u> In many problems, the underlying partitions are known or can be conceptualized. For true brain-like intelligence, it will be desirable to learn such partitions. This could be achieved either off-line or on-line. Definitely on-line implementation will be very difficult to achieve. Off-line synthesis could be achieved in several ways. One simple way is to use parsimonious networks with pruning capabilities to learn the internal partitions. Another approach could be clustering via unsupervised learning.

<u>Decisions and failure accommodation</u>

As suggested by Werbos and others, the partitions can be navigated more efficiently by including a decision layer in the network of critics. This concept is illustrated below. Also, failure accommodation could be handled inside of the decision layer. In the figure shown, we assume that the network behaves like a winner-all type with lateral inhibition implying that only one of the neurons output the control vector.

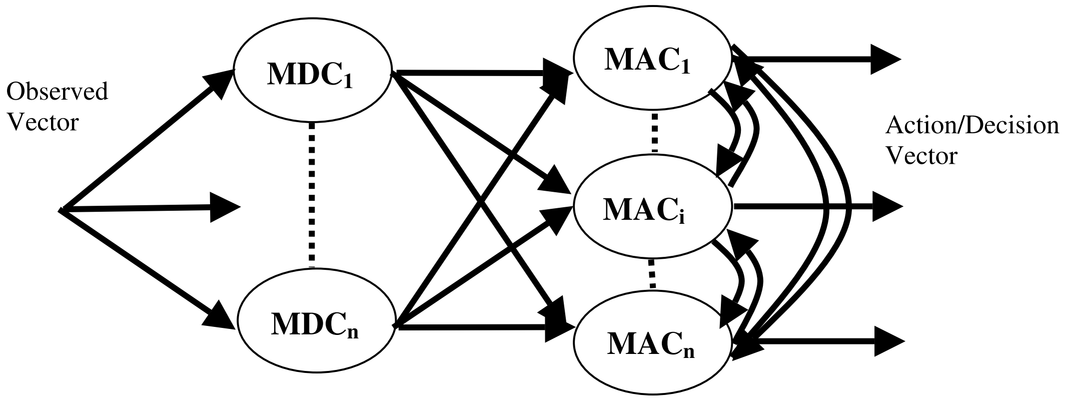


Figure 5. Network of Critics with a Decision Maker Layer
(MDC – Model-Decision-Critic)

7 Conclusions

In this paper, we presented four problem domains that are well suited for further advances in intelligent systems. These domains include, design, autonomous maneuvering, prediction, and decision-making. All of these require advances in intelligent system technologies to enable higher levels of automation, intelligence, and application success. All four problem domains defined are accompanied by one possible approach based on authors' knowledge and perspective. It is hoped that this will further advance intelligent system research and development for aerospace applications.

There are definitely several other problem domains where intelligent system technologies could help but need major advances to achieve high levels of success. Some of these areas include data-driven inverse design and modeling, autonomous planning and scheduling, and bio-inspired aerial vehicles.

8 References

1. K. KrishnaKumar, Immunized Neurocontrol: Concepts and Initial Results, Presented at the workshop on combinations of genetic algorithms and neural networks, COGANN'92, Baltimore, MD, June, 1992.
2. K. KrishnaKumar and J. C. Neidhoefer, Immunized Artificial Systems--Concepts and Applications, in Genetic Algorithms in Computers and Engineering, John Wiley & Sons, 1997.
3. K. KrishnaKumar and J. C. Neidhoefer, Immunized Neuro-control, Expert Systems with Applications, 1997.
4. J. Kaneshige and K. Gundy-Burlet, Integrated Neural Flight and Propulsion Control System, AIAA-2001-4386, August 2001.
5. R. T. Rysdyk and Anthony J. Calise, Fault Tolerant Flight Control via Adaptive Neural Network Augmentation, AIAA 98-4483, August 1998.
6. K. Krishnakumar, N. Kulkarni, "Inverse Adaptive Neuro-Control for the control of a turbofan engine", Proceedings of AIAA conference on Guidance, Navigation and Control, Portland, OR, 1999.
7. T. L. Chen, A. R. Pritchett, On-The-Fly Procedure Development for Flight Re-Planning Following System Failures, AIAA 2000-0300, January 2000.

8. A. Satyadas, Cognitive Prediction and Control using Soft Computing, Technical Report, University of Alabama, USA, 1998.
9. K. KrishnaKumar, Levels of Intelligent Control, AIAA Tutorial at New Orleans, LA, August 1997.
10. G. C. Reinsel, Element of multivariate time series analysis, Springer-Verlag, New York, 1986.
11. A. S. Weigend, N. A. Gershenfeld (editors), Time Series Prediction, Addison-Wesley Publishing Co., New York, 1994.
12. D. G. Bails, L. C. Peppers, Business fluctuations: forecasting techniques and applications, Prentice-hall, Inc., New Jersey, 1982.
13. U. Harigopal, A. Satyadas, Cognizant Enterprise Maturity Model, co-guest editors: A. Satyadas, U. Harigopal, N. Cassaigne, IEEE Transactions on Systems Man and Cybernetics - Part C: Applications and Reviews: Special Issue on Knowledge Management, Vol. 31, No. 4, pp 449-459, November 2001.
14. J. Hamilton, Time Series Analysis, Princeton University Press, Princeton, NJ, USA, 1994.
15. P. Werbos, A Brain-like Design to Learn Optimal Strategies in Complex Environments, in Brain-Like Computing and Intelligent Information Systems, Springer-Verlag Singapore Pte. Ltd. 1998
16. A. Barto, Reinforcement Learning and Adaptive Critic Methods. Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches. Van Nosttrand Reinhold, Kentucky, 1992.
17. I. Bertsekas, J. Tsitsiklis, Neuro-Dynamic Programming, Athena Scientific, Belmont, Massachusetts, 1996.
18. P. Werbos, Approximate Dynamic Programming For Real-Time Control and Neural Modeling, Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches. Van Nosttrand Reinhold, Kentucky, USA, 1993.
19. R. Howard. Dynamic Programming and Markov Process, Cambridge, MA, MIT Press, 1960.
20. K. Krishnakumar, Optimization of the Neural Net Connectivity Pattern Using a Back-Propagation Algorithm, Journal of Neurocomputing, (5) pp. 273-286, 1993.

REPORT DOCUMENTATION PAGE

1. Recipient's Reference	**2. Originator's References** RTO-EN-022 AC/323(AVT-095)TP/69	**3. Further Reference** ISBN 92-837-1101-7	**4. Security Classification of Document** UNCLASSIFIED/ UNLIMITED

5. Originator Research and Technology Organisation
North Atlantic Treaty Organisation
BP 25, F-92201 Neuilly-sur-Seine Cedex, France

6. Title
Intelligent Systems for Aeronautics

7. Presented at/sponsored by
the Applied Vehicle Technology Panel (AVT) and the von Kármán Institute for Fluid Dynamics (VKI) in Rhode-Saint-Genèse, Belgium, 13-17 May 2002.

8. Author(s)/Editor(s) Multiple	**9. Date** June 2003
10. Author's/Editor's Address Multiple	**11. Pages** 272

12. Distribution Statement There are no restrictions on the distribution of this document. Information about the availability of this and other RTO unclassified publications is given on the back cover.

13. Keywords/Descriptors

Adaptive systems
Airfoils
Control
UAV (Unmanned Air Vehicles)
Decision aids
Decision making
Design
Game theory
Fault identification
Genetic algorithms
Intelligent Systems (IS)
Military applications
Missile design
Multi-agent theory
Neural networks
Optimization
Pattern recognition
Planning
Problem solving
Rocket plume data for condition monitoring
Search theory
Space exploration
Sub-symbolic systems
Symbolic
Uncertainty management

14. Abstract

Intelligent systems are suited as search and optimization, pattern recognition and matching, planning, uncertainty management control and adaptation.

These lecture notes cover techniques and application with emphasis on aeronautical and space applications.

Techniques reviewed: decision strategy tools based on game theory, neural network techniques for fault identification, genetic algorithms and multi-agent theory.

Application reviewed: air combat tactics, control of unmanned air vehicles, air combat simulation, space exploration, missile design, airfoil design, analysis of rocket plume data for condition monitoring. These applications highlight the relevance and importance of Intelligent Systems for military issues.

DIFFUSION DES PUBLICATIONS

RTO NON CLASSIFIEES

L'Organisation pour la recherche et la technologie de l'OTAN (RTO), détient un stock limité de certaines de ses publications récentes, ainsi que de celles de l'ancien AGARD (Groupe consultatif pour la recherche et les réalisations aérospatiales de l'OTAN). Celles-ci pourront éventuellement être obtenues sous forme de copie papier. Pour de plus amples renseignements concernant l'achat de ces ouvrages, adressez-vous par lettre ou par télécopie à l'adresse indiquée ci-dessus. Veuillez ne pas téléphoner.

Des exemplaires supplémentaires peuvent parfois être obtenus auprès des centres nationaux de distribution indiqués ci-dessous. Si vous souhaitez recevoir toutes les publications de la RTO, ou simplement celles qui concernent certains Panels, vous pouvez demander d'être inclus sur la liste d'envoi de l'un de ces centres.

Les publications de la RTO et de l'AGARD sont en vente auprès des agences de vente indiquées ci-dessous, sous forme de photocopie ou de microfiche. Certains originaux peuvent également être obtenus auprès de CASI.

CENTRES DE DIFFUSION NATIONAUX

ALLEMAGNE
Streitkräfteamt / Abteilung III
Fachinformationszentrum der Bundeswehr, (FIZBw)
Friedrich-Ebert-Allee 34
D-53113 Bonn

BELGIQUE
Etat-Major de la Défense
Département d'Etat-Major Stratégie
ACOS-STRAT-STE – Coord. RTO
Quartier Reine Elisabeth
Rue d'Evère, B-1140 Bruxelles

CANADA
DSIGRD2
Bibliothécaire des ressources du savoir
R et D pour la défense Canada
Ministère de la Défense nationale
305, rue Rideau, 9e étage
Ottawa, Ontario K1A 0K2

DANEMARK
Danish Defence Research Establishment
Ryvangs Allé 1, P.O. Box 2715
DK-2100 Copenhagen Ø

ESPAGNE
INTA (RTO/AGARD Publications)
Carretera de Torrejón a Ajalvir, Pk.4
28850 Torrejón de Ardoz - Madrid

ETATS-UNIS
NASA Center for AeroSpace Information (CASI)
Parkway Center
7121 Standard Drive
Hanover, MD 21076-1320

FRANCE
O.N.E.R.A. (ISP)
29, Avenue de la Division Leclerc
BP 72, 92322 Châtillon Cedex

GRECE (Correspondant)
Defence Industry & Research General Directorate
Research Directorate
Fakinos Base Camp
S.T.G. 1020
Holargos, Athens

HONGRIE
Department for Scientific Analysis
Institute of Military Technology
Ministry of Defence
H-1525 Budapest P O Box 26

ISLANDE
Director of Aviation
c/o Flugrad
Reykjavik

ITALIE
Centro di Documentazione Tecnico-Scientifica della Difesa
Via XX Settembre 123a
00187 Roma

LUXEMBOURG
Voir Belgique

NORVEGE
Norwegian Defence Research Establishment
Attn: Biblioteket
P.O. Box 25, NO-2007 Kjeller

PAYS-BAS
Royal Netherlands Military Academy Library
P.O. Box 90.002
4800 PA Breda

POLOGNE
Armament Policy Department
218 Niepodleglosci Av.
00-911 Warsaw

PORTUGAL
Estado Maior da Força Aérea
SDFA - Centro de Documentação
Alfragide
P-2720 Amadora

REPUBLIQUE TCHEQUE
DIC Czech Republic-NATO RTO
VTÚL a PVO Praha
Mladoboleslavská ul.
Praha 9, 197 06, Česká republika

ROYAUME-UNI
Dstl Knowledge Services
Kentigern House, Room 2246
65 Brown Street
Glasgow G2 8EX

TURQUIE
Millî Savunma Başkanliği (MSB)
ARGE Dairesi Başkanliği (MSB)
06650 Bakanliklar - Ankara

AGENCES DE VENTE

NASA Center for AeroSpace Information (CASI)
Parkway Center
7121 Standard Drive
Hanover, MD 21076-1320
Etats-Unis

The British Library Document Supply Centre
Boston Spa, Wetherby
West Yorkshire LS23 7BQ
Royaume-Uni

Canada Institute for Scientific and Technical Information (CISTI)
National Research Council
Acquisitions
Montreal Road, Building M-55
Ottawa K1A 0S2, Canada

Les demandes de documents RTO ou AGARD doivent comporter la dénomination "RTO" ou "AGARD" selon le cas, suivie du numéro de série (par exemple AGARD-AG-315). Des informations analogues, telles que le titre et la date de publication sont souhaitables. Des références bibliographiques complètes ainsi que des résumés des publications RTO et AGARD figurent dans les journaux suivants:

Scientific and Technical Aerospace Reports (STAR)
STAR peut être consulté en ligne au localisateur de ressources uniformes (URL) suivant:
http://www.sti.nasa.gov/Pubs/star/Star.html
STAR est édité par CASI dans le cadre du programme NASA d'information scientifique et technique (STI)
STI Program Office, MS 157A
NASA Langley Research Center
Hampton, Virginia 23681-0001
Etats-Unis

Government Reports Announcements & Index (GRA&I)
publié par le National Technical Information Service
Springfield
Virginia 2216
Etats-Unis
(accessible également en mode interactif dans la base de données bibliographiques en ligne du NTIS, et sur CD-ROM)

Imprimé par Groupe d'imprimerie St-Joseph inc.
(Membre de la Corporation St-Joseph)
1165, rue Kenaston, Ottawa (Ontario), Canada K1G 6S1

DISTRIBUTION OF UNCLASSIFIED RTO PUBLICATIONS

NATO's Research and Technology Organisation (RTO) holds limited quantities of some of its recent publications and those of the former AGARD (Advisory Group for Aerospace Research & Development of NATO), and these may be available for purchase in hard copy form. For more information, write or send a telefax to the address given above. **Please do not telephone**.

Further copies are sometimes available from the National Distribution Centres listed below. If you wish to receive all RTO publications, or just those relating to one or more specific RTO Panels, they may be willing to include you (or your organisation) in their distribution.

RTO and AGARD publications may be purchased from the Sales Agencies listed below, in photocopy or microfiche form. Original copies of some publications may be available from CASI.

NATIONAL DISTRIBUTION CENTRES

BELGIUM
Etat-Major de la Défense
Département d'Etat-Major Stratégie
ACOS-STRAT-STE – Coord. RTO
Quartier Reine Elisabeth
Rue d'Evère, B-1140 Bruxelles

CANADA
DRDKIM2
Knowledge Resources Librarian
Defence R&D Canada
Department of National Defence
305 Rideau Street, 9th Floor
Ottawa, Ontario K1A 0K2

CZECH REPUBLIC
DIC Czech Republic-NATO RTO
VTÚL a PVO Praha
Mladoboleslavská ul.
Praha 9, 197 06, Česká republika

DENMARK
Danish Defence Research Establishment
Ryvangs Allé 1, P.O. Box 2715
DK-2100 Copenhagen Ø

FRANCE
O.N.E.R.A. (ISP)
29 Avenue de la Division Leclerc
BP 72, 92322 Châtillon Cedex

GERMANY
Streitkräfteamt / Abteilung III
Fachinformationszentrum der Bundeswehr, (FIZBw)
Friedrich-Ebert-Allee 34
D-53113 Bonn

GREECE (Point of Contact)
Defence Industry & Research General Directorate
Research Directorate
Fakinos Base Camp
S.T.G. 1020
Holargos, Athens

HUNGARY
Department for Scientific Analysis
Institute of Military Technology
Ministry of Defence
H-1525 Budapest P O Box 26

ICELAND
Director of Aviation
c/o Flugrad
Reykjavik

ITALY
Centro di Documentazione Tecnico-Scientifica della Difesa
Via XX Settembre 123a
00187 Roma

LUXEMBOURG
See Belgium

NETHERLANDS
Royal Netherlands Military Academy Library
P.O. Box 90.002
4800 PA Breda

NORWAY
Norwegian Defence Research Establishment
Attn: Biblioteket
P.O. Box 25, NO-2007 Kjeller

POLAND
Armament Policy Department
218 Niepodleglosci Av.
00-911 Warsaw

PORTUGAL
Estado Maior da Força Aérea
SDFA - Centro de Documentação
Alfragide
P-2720 Amadora

SPAIN
INTA (RTO/AGARD Publications)
Carretera de Torrejón a Ajalvir, Pk.4
28850 Torrejón de Ardoz - Madrid

TURKEY
Millî Savunma Başkanliği (MSB)
ARGE Dairesi Başkanliği (MSB)
06650 Bakanliklar - Ankara

UNITED KINGDOM
Dstl Knowledge Services
Kentigern House, Room 2246
65 Brown Street
Glasgow G2 8EX

UNITED STATES
NASA Center for AeroSpace Information (CASI)
Parkway Center
7121 Standard Drive
Hanover, MD 21076-1320

SALES AGENCIES

NASA Center for AeroSpace Information (CASI)
Parkway Center
7121 Standard Drive
Hanover, MD 21076-1320
United States

The British Library Document Supply Centre
Boston Spa, Wetherby
West Yorkshire LS23 7BQ
United Kingdom

Canada Institute for Scientific and Technical Information (CISTI)
National Research Council
Acquisitions
Montreal Road, Building M-55
Ottawa K1A 0S2, Canada

Requests for RTO or AGARD documents should include the word 'RTO' or 'AGARD', as appropriate, followed by the serial number (for example AGARD-AG-315). Collateral information such as title and publication date is desirable. Full bibliographical references and abstracts of RTO and AGARD publications are given in the following journals:

Scientific and Technical Aerospace Reports (STAR)
STAR is available on-line at the following uniform resource locator:
http://www.sti.nasa.gov/Pubs/star/Star.html
STAR is published by CASI for the NASA Scientific and Technical Information (STI) Program
STI Program Office, MS 157A
NASA Langley Research Center
Hampton, Virginia 23681-0001
United States

Government Reports Announcements & Index (GRA&I)
published by the National Technical Information Service
Springfield
Virginia 22161
United States
(also available online in the NTIS Bibliographic Database or on CD-ROM)

Printed by St. Joseph Print Group Inc.
(A St. Joseph Corporation Company)
1165 Kenaston Street, Ottawa, Ontario, Canada K1G 6S1

ISBN 92-837-1101-7