Advances in Discrete Sensitivity Methods Applied to Uncertainty Analysis

Chad E. Burdyshaw[*], **W. Kyle Anderson**[†]
UT SimCenter at Chattanooga
701 East M.L. King Blvd
Chattanooga TN 37403

Chad-Burdyshaw@utc.edu

ABSTRACT

Small uncertainties in the modelling of physical phenomena can result in large inaccuracies in computationally predicted outcomes. When these predictions are used to inform design in various real world applications (civil structures, vehicles, etc.) results can be costly and disastrous.[1] Therefore, methods to quantify the confidence of computed solutions are an imperative.

Currently existing technologies in computational sensitivity analysis may be used to address particular sources of parameter uncertainty (epistemic). Computational sensitivity analysis techniques have a long history in their application to uncertainty analysis.[2,3,4,5] However, difficulties associated with the efficiency, accuracy, and implementation of these methods have limited their use.

This paper discusses recent developments in computational sensitivity analysis which address the problems restricting its use. In particular, a suite of discrete sensitivity methods (direct and adjoint) have been implemented which address the issues of accuracy, efficiency, ease of implementation, and extensibility.

1. INTRODUCTION

Uncertainty and error in computational predictions are the result of several factors. These factors include: intrinsic variability in the physical system being modelled, such as oscillatory and/or turbulent behaviour and assumptions of unknown values, such as boundary conditions, initial conditions, and mathematical modelling parameters (turbulence, chemical kinetics, equation of state, etc). In addition to uncertainty, error is knowingly introduced as a result of approximations of known quantities resulting from discretization of the field, reduced order of the governing equations, poor convergence of iterative PDE solvers, and finite representation of floating point variables.

In some cases, such as discontinuous or noisy functions, wide parameter variability, small parameter sets, or cheap function evaluations, the use of stochastic or non-deterministic methods (Monte Carlo[6], Interval analysis[7], Polynomial chaos[8,9], etc.) are appropriate. For other cases, most notably those with computationally expensive function evaluations and large parameter sets, deterministic or derivative methods are the only feasible methods for determining uncertainty.

The principal approach addressed in this study is the adjoint method of sensitivity analysis. This deterministic method is ideally suited for computing first derivatives of a function with respect to very large numbers of design variables. An adjoint solver can be constructed using either a continuous formulation,[10] in which the

[*] Assistant Research Professor, SimCenter, University of Tennessee at Chattanooga
[†] Professor, SimCenter, University of Tennessee at Chattanooga

Burdyshaw, C.E.; Anderson, W.K. (2007) Advances in Discrete Sensitivity Methods Applied to Uncertainty Analysis. In *Computational Uncertainty in Military Vehicle Design* (pp. 52-1 – 52-20). Meeting Proceedings RTO-MP-AVT-147, Paper 52. Neuilly-sur-Seine, France: RTO. Available from: http://www.rto.nato.int.

continuous form of the governing equations are differentiated and then discretized, or alternatively, a discrete formulation[11,12,13] in which the discretized form of the governing equations are differentiated. As the field discretization approaches infinity, these two methods converge to give the same solution. However, in order to measure the uncertainty of a solution computed using the discretized continuity equations of a flow solver, a discrete adjoint formulation is more appropriate. Thus the discrete adjoint formulation has been adopted for this study.

Adjoint methods have been used in computational science applications for over three decades,[14,15,16] but their proliferation in the analysis community has been hindered by a number of factors. Specifically, the prohibitive cost of implementing and maintaining a large-scale solver that is both accurate and efficient. This paper presents an adjoint implementation method which addresses the barriers to proliferation in the wider solver community. The method is applied to the Tenasi[17] parallel (distributed), 3-D, unstructured, finite-volume, Navier-Stokes solver.

This report will highlight the utility and ease of implementation of the sensitivity method. As an example of the application of the method, this study presents an analysis of the effects of the parameters of various turbulence models (Spalart-Allmaras, k-ε, q-ω) on the prediction of the point of flow reattachment. In addition, the effects of Arrhenius reaction rate constants on the specific impulse calculation, is presented for combustion in a supersonic nozzle.

2. THE DISCRETE ADJOINT SENSITIVITY METHOD

One approach to the formulation of the discrete adjoint equations can be explained in terms of a constrained optimization problem (for an alternate linear algebra approach see Giles and Pierce[18]). An objective function is chosen which may be a function of state variables (Q), mesh variables (χ), and design variables (β), though for exposition, our function eq.(1), will be dependent only on the state and design variables.

$$f(Q,\beta) \qquad (1)$$

The chain rule is applied to obtain the total derivative of f with respect to β.

$$\frac{df}{d\beta} = \frac{\partial f}{\partial \beta} + \frac{\partial f}{\partial Q}\frac{\partial Q}{\partial \beta} \qquad (2)$$

A Lagrangian form of the objective function is constructed by adding the constraint $\lambda^{\mathrm{T}} R(Q,\beta)$, where it is assumed that the flux residual function $R(Q,\beta) = 0$ (i.e. we have a steady state problem) . The result is that the value of the adjoint or "costate" variable (λ) is arbitrary.

$$L(Q,\beta,\lambda) = f(Q,\beta) + \lambda^T R(Q,\beta) \qquad (3)$$

Applying the chain rule to equation (3) and rearranging terms to obtain equation (4), allows the elimination of the $\frac{\partial Q}{\partial \beta}$ term with the solution of the adjoint equation (5). By eliminating this term, the process is freed from its dependency on the size of the β vector. The result is that the total derivative no longer requires a state

variable solution to be computed for each design variable as in the direct sensitivity method from eq.(2).

$$\frac{dL}{d\beta} = \frac{\partial f}{\partial \beta} + \lambda^T \frac{\partial R}{\partial \beta} + \left[\frac{\partial f}{\partial Q} + \lambda^T \frac{\partial R}{\partial Q} \right] \frac{\partial Q}{\partial \beta} \quad (4)$$

$$\left[\frac{\partial R}{\partial Q} \right]^{\mathrm{T}} \lambda = -\left[\frac{\partial f}{\partial Q} \right]^{\mathrm{T}}, \quad \text{solve for } \lambda \quad (5)$$

With the solution of the λ vector, the expression for the total derivative is reduced to equation (6).

$$\frac{dL}{d\beta} = \frac{\partial f}{\partial \beta} + \lambda^T \frac{\partial R}{\partial \beta} \quad (6)$$

The advantage of the adjoint method is balanced by the considerable assumption that the flux residual of the state variables is approximately zero. However, this assumption is valid for a large class of problems (i.e. steady state).

The primary barrier to proliferation of the adjoint method is the difficulty in the implementation of a solver that is both accurate and efficient. The key to simplifying the implementation process is in choosing the right technology to compute the partial derivative terms in equations (5) and (6).

The simplest way to numerically compute a partial derivative is to use a Taylors series based method, known as a finite difference. For example, the central-difference method requires simply taking the difference between two function evaluations operating on identical domains, with the exception of a small perturbation in equal and opposite directions on the variable of interest. The advantages are that finite differences may be computed without requiring access to the solvers source code. Furthermore, derivatives produced using the central difference have second-order truncation error with respect to perturbation size. However the perturbation size is restricted to a lower limit (usually around 10^{-5}) due to a phenomenon known as subtractive cancellation error. As a result, the accuracy of the method is limited, and thus not suitable for cases of weak or unknown sensitivity.

Traditionally, the only way to implement a suitably accurate adjoint solver is to apply “hand differentiation” (HD) to the function solver code. This is accomplished by adding, line by line, chain rule differentiations for each mathematical expression contained within the source code. This process is extremely labor intensive and error prone, and as a consequence, is prohibitive for most large scale solvers.[19,20]

Attempts to automate the HD process have resulted in Automatic Differentiation (AD) codes which have a fairly long history of success in generating direct mode sensitivity solvers.[21] These AD codes operate by applying transformational routines to the original solver code, thereby generating a new, differentiated source code through automated chain rule applications.

There are only a small number of AD codes currently available that can implement an adjoint solver.[22,23] Adjoint AD solvers are available for commonly used programming languages (C,C++, FORTRAN). In principle, AD appears to be an ideal tool for differentiation. In practice however, the process is often not straightforward. Furthermore, flexibility in the application is poor due to reliance on external software. Tailoring or tuning the implementation for special situations requires modification of the machine generated

source code which can be unintelligible to a human reader. Some of these situations include performance issues, specific language features, and parallel implementation.

An alternative exists which significantly reduces the implementation costs associated with either HD or AD while retaining the accuracy and application flexibility of HD. This method, termed the CTSE adjoint or "complex" adjoint [24,25,26], uses the complex Taylors series expansion (CTSE) method [27,28,29] to compute partial derivative terms. The CTSE method is akin to finite difference methods, but does not suffer from the associated subtractive cancellation error. As a result, partial derivatives computed in this way have an accuracy equivalent to the chain rule differentiation implemented in HD and AD codes.

Complex Taylors series expansion method (CTSE)

The CTSE method is developed from a Taylors series expansion of a real valued function in which a perturbation is applied along the imaginary axis, eq.(7).

$$f(\beta+i\Delta h)=f(\beta)+ih\frac{\partial f(\beta)}{\partial\beta}-h^2\frac{1}{2}\frac{\partial^2 f(\beta)}{\partial\beta^2}-ih^3\frac{1}{6}\frac{\partial^3 f(\beta)}{\partial\beta^3}+h^4\frac{1}{6}\frac{\partial^4 f(\beta)}{\partial\beta^4}+\cdots \quad (7)$$

The real part of the series returns the unperturbed function value.

$$f(\beta)=\text{Real}\left[f(\beta+i\Delta h)\right]+\text{O}(h^2) \quad (8)$$

The imaginary part of the series, divided by the perturbation size returns the first derivative.

$$\frac{\partial f(\beta)}{\partial\beta}=\text{Imag}\left[\frac{f(\beta+i\Delta h)}{h}\right]+\text{O}(h^2) \quad (9)$$

Accuracy

Both expressions have a second order truncation error, however if the perturbation size is less than or equal to the square root of machine zero (e.g. 1.0^{-8} for 16-bit accuracy), the resulting truncation error diminishes to machine zero. Under this condition, the CTSE derivative technique has an accuracy equivalent to an HD or AD generated code.

3. IMPLEMENTATION

Implementing the CTSE adjoint code involves both additions and modifications to the function solver code. The resulting sensitivity solver is integrated into the function solver and not a separate piece of code. This integrated approach allows for a simplified implementation wherein much of the code is reused. Once implementation is complete, subsequent modifications to the function solver are naturally and automatically differentiated and included in the adjoint solver.

The implementation of a CTSE adjoint solver involves three basic operations: definition of an objective function, computation of selected partial derivatives, and solution of the linear system of equations.

Objective Function Definition

Definition of the objective function simply requires that the choice of objective (e.g. Lift, Drag, Specific Impulse, Reattachment length, etc.) be in some encapsulated form in order to facilitate differentiation. In many cases the desired function is not provided for in the original solver and thus requires an unavoidable bit of additional coding.

Partial Derivative Computation

Partial derivative computation is accomplished by applying the CTSE method to the expressions of interest. If the expressions to be differentiated are in an encapsulated form (e.g. a function call), the CTSE method can be applied by creating a duplicate function in which complex variables are substituted in place of real variables. This is a simple process, though it requires updating whenever modifications are made to the function solver code.

Polymorphism

However, if the function solver code is written in a language that allows functional parametric polymorphism (C++,C#,etc.), the function may be templated such that it can accept parameters with changing data types. The results will then be computed according to both the values and types of the given input parameters (Figure 1). Subroutines called within this function also take on the character of the given input parameters, thereby propagating the CTSE-based differentiation throughout the hierarchy of function calls.

Polymorphic functions are the preferred method of "complexification" for a number of reasons. First of all, the templates are applied to the original function solver code, which allows subsequent modifications of the encapsulated objects/functions in the function solver to be automatically differentiated and incorporated into the sensitivity solver without requiring additional attention from the programmer. Furthermore, efficiency is improved (over a completely complex code) since the necessary routines use complex arithmetic only when directed to do so. Another advantage is that a forward mode sensitivity solver is obtained for free, by simply passing the appropriate complex values into the function solver at the top level.

Figure 1 shows an example of a polymorphic parameter Addition function. This function takes variables of various types and computes the result according to the type given.

Type	V1	V2
Int	3	2
String	"Poly"	"morph"
Complex	1+4i	6+3i
Int Array	{1,2,3}	{4,5,6}

→

```
Add(Type V1, Type V2)
{
    return (V1+V2);
}
```

→

Result
5
"Polymorph"
7+7i
{5,7,9}

Figure 1: Polymorphic Function

Efficient Implementation

In order to compute the derivatives using the CTSE method described in equation (9), a perturbation has to be introduced to the design variable, followed by a function evaluation, and then division by the perturbation

size. This requires implementation of a control structure to coordinate the necessary perturbations, evaluations, and storage of the results. If this process is not done carefully, the computational costs can be prohibitively expensive in terms of both time and memory.

Targeted differentiation (global functions e.g. integrals)

A naïve approach to calculating a partial derivative involves perturbing the independent variable, evaluating the function over the entire domain, and then dividing the imaginary portion of the scalar result. This process is highly inefficient due to the fact that, in a discretized field, only the domain about the point of perturbation has any effect on the result. Thus the majority of the work in such a derivative computation is unnecessary. This inefficiency can be remedied by limiting the function evaluation to the local domain upon which the derivative is dependent.

In order to orchestrate these localized evaluations, for an unstructured mesh topology, it is necessary to compile a list of the neighbours for each node in the mesh according to the spatial dependency of the function (i.e. 1st or 2nd order). Furthermore, for each function to be differentiated, the control structures which loop over the entire mesh must be altered to allow a loop over only the nodes specified in a given localized domain. This process of "localizing" some or all of the function solver code, constitutes the bulk of the work required for the sensitivity code implementation. Still the implementation costs are far less than HD and errors are generally obvious.

Simultaneous evaluation (vector valued functions e.g. Jacobians)

The use of localized evaluations is most effective when calculating derivatives for vector valued functions such as the flux residual. A large number of partial derivative values (e.g. $\partial R / \partial Q$, $\partial R / \partial \chi$) can be computed simultaneously, provided the influence of the perturbations do not overlap.

A simple one dimensional example of the simultaneous evaluation process is explained in Figure 2. The mesh is divided into groups or "colors" of nodes which can be perturbed without affecting other nodes of the same color.

The coloring process begins when the leftmost node is selected and "colored" (color 1 shown in red). The next node in line is checked by comparing its computational stencil (outlined in black) for overlap with the stencils of all nodes of color 1. The second node fails the test, as does the third, and are passed over. The fourth node from the left passes the test and is colored by color 1. This testing process continues with color 1, until all nodes have been checked. When no more nodes can be added to color 1, a new color is started and the process begins again with the first uncolored node, checking for stencil overlap with all nodes of color 2. The coloring process continues until all nodes have been assigned to a color. In this example case, a third color is required to cover the remaining mesh nodes.

With the mesh colored in this manner, the sensitivities of node values can be computed simultaneously for all nodes of the same color. This is accomplished by performing all necessary perturbations for nodes of the same color, followed by a single function evaluation over all localized domains of these nodes, and finally dividing the results by the perturbation size and storing the result. This process is repeated for each subsequent color.

The result of coloring the mesh in Figure 2 is that the linearized residual ($\partial R / \partial Q$) can now be computed with just three residual evaluations, whereas a naïve approach would have required twenty separate residual

evaluations. As the size of the mesh grows, the efficiency of this method increases.

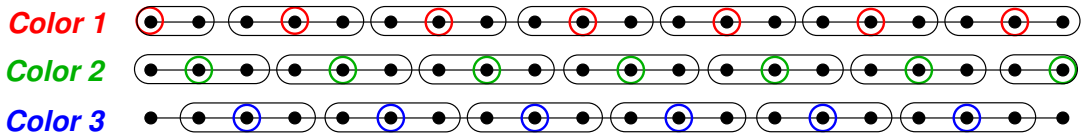

Figure 2: Mesh Coloring for Simultaneous Evaluation

Code Reuse and the Linear Equations Solver

In addition to the function and residual codes being reused in the calculation of partial derivatives, the routines used to solve the linearized conservation equations can also be used to compute the adjoint solution. In the Tenasi code, a couple of alternatives are available for the solution of linear systems of equations.

The adjoint variables can be computed directly using a Krylov subspace method such as the generalized minimum residual method (GMRES).[30] A great advantage to using GMRES is that it does not suffer from stability problems, but it does require a significant amount of additional storage. The memory requirements scale linearly with the number of Krylov search directions, and the use of a pre-conditioner can double the memory needed to store the Jacobian.

$$\text{Solve}: \left[\frac{\partial R}{\partial Q}\right]^{\mathrm{T}} \lambda = -\left[\frac{\partial f}{\partial Q}\right]^{\mathrm{T}}, \quad \text{solve for } \lambda \qquad (10)$$

Another approach is to embed the linear adjoint system of equations into a nonlinear framework which allows the use of an iterative nonlinear Newton solver with Gauss Seidel or GMRES for the inner solve. The adjoint equation is cast in iterative form (11) by solving for $\Delta\lambda$ at each Newton iteration.

$$\left\{\tilde{I}\frac{Vol}{\Delta t}+\left[\frac{\partial R}{\partial Q}\right]^{\mathrm{T}}\right\}\Delta\lambda^{i} = -\left[\frac{\partial f}{\partial Q}\right]^{\mathrm{T}} + \left[\frac{\partial R}{\partial Q}\right]^{\mathrm{T}} \lambda^{i-1}, \quad \text{solve for } \Delta\lambda^{i} = (\lambda^{i} - \lambda^{i-1}) \to 0 \qquad (11)$$

Unlike the Krylov methods, there is no need to store previous search information, and preconditioning takes the form of a pseudo time derivative multiplier $\left(\tilde{I}\frac{Vol}{\Delta t}\right)$ to the diagonal. Thus the Newton method takes no additional memory beyond storage of a second-order Jacobian, and the temporary $\Delta\lambda$ vector. An additional advantage of this method is that a lower order linearized residual ($\partial R/\partial Q$) can be used on the left hand side, which allows the solver to operate on a system with a smaller condition number than the full second-order Jacobian. The disadvantage to the Newton method is that in many cases, the stability is too poor to obtain a solution.

4. ALTERNATIVE DISCRETE SENSITIVITY METHODS

As mentioned above, the adjoint sensitivity methods primary advantage is its ability to simultaneously

compute the sensitivity of a single objective with respect to a large set of design variables. This method's accuracy however, is limited by the degree of convergence of the original state variable solution. For cases in which a well converged flow solution is not possible, direct differentiation, using forward CTSE, may be used to compute sensitivities independently, one design variable at a time.

Direct differentiation of a partially converged solution may not accurately predict the behaviour of the physical system being modelled but will predict the behaviour of the computational solution in its unconverged state, and thus may be used to express uncertainty in the computation of the objective function at various stages of convergence. This method requires a separate flow solution for each design variable, and so is infeasible for analysis of systems with large sets of design variables. However, this method can be applied where appropriate, to obtain limited sets of derivatives which can then be used to either check the accuracy of the adjoint solution or to directly propagate parameter uncertainties for unsteady problems. The proposed adjoint implementation method has the added benefit of providing the structure to compute sensitivities using the direct mode.

Direct Mode Differentiation

In the Direct (or Forward) mode analysis, sensitivities are obtained by computing the total derivative expression of equation (12).

$$\frac{df}{d\beta} = \frac{\partial f}{\partial \beta} + \frac{\partial f}{\partial Q}\frac{\partial Q}{\partial \beta} \qquad (12)$$

Simultaneous analysis

The total derivative may be computed simultaneously along with the flow solution by simply computing the flow solution using complex variable arithmetic. The design variable is perturbed prior to execution, thus the imaginary portion of the solution provides the derivative, which evolves along with the flow solution. In this manner, the resulting derivative gives a history of the behaviour of the flow solution at each time step. The disadvantage of this method is the high cost of computing the flow solution using complex variable arithmetic, which can be up to four times as costly as a real variable solution. Furthermore, a disadvantage of the direct method in general is that a new flow solution is required for each additional design variable.

5. UNCERTAINTY PROPAGATION

Uncertainty of the objective function is propagated through the derivatives of the function, and can be expressed as a linear extrapolation about the unperturbed values of the design variables.

The total uncertainty is computed as the sum of the absolute values of the derivatives multiplied by an absolute variance associated with each design variable as shown in equation (13). The linear assumption of the behaviour of the objective function requires the magnitude of the parameter variance be small for this expression to be accurate. Careful choice of a reasonable variance for selected parameters should of course be dependent upon knowledge of the physical meaning of the parameters and/or experimental variance.

$$|\Delta f| \approx \sum_{i=1}^{n} \left| \frac{df(Q,\beta)}{d\beta_i} \right| |\Delta\beta_i| \quad (\Delta\beta_i = \text{max variance of parameter}) \qquad (13)$$

Suggested variances for turbulence model parameter range from 4% cited by Jones and Launder for the k-e model[31] up to a maximum variance of 300% for the C_{t4} parameter in the Spalart-Allmaras[36] model. Most parameter uncertainties however, have a variance around 4%-8% of the initial value. Other turbulence model uncertainty studies such as Pelletier[2], and Turgeon, Pelletier, Etienne and Borggaard[3], choose much smaller variances of 0.25% to 5% for selected values. In the following example cases, where suggested variances can not be found, a liberal variance is taken to be 10% of the initial value.

6. DISCRETIZATION ERROR

Several studies have shown that field discretization error is a major factor in the accuracy of computational results. In order to avoid over refinement, resulting in large and unnecessary cost increases, an adaptive refinement scheme is chosen which makes further use of the adjoint solver. Traditional gradient based adaptation may help to resolve certain flow features such as discontinuities. However, this can result in an overemphasis on regions which may have little impact on the value of interest.

An alternative adaptation method known as "output based error correction and adaptation," [32,33] directly targets the accuracy of a chosen functional by assigning an error estimate to each node based on a combination of the function and adjoint values on both the original mesh and an interpolation of these values to a dynamically imbedded refined mesh. An adaptation of the original mesh involves refining the elements around nodes which have an error value above a chosen threshold. This insures that the mesh is refined only in regions which have an impact on the accuracy of the function evaluation.

Mesh adaptation driven by either gradient based or output based error methods presents its own set of difficulties. In particular in the presence of structured boundary layer elements, adaptation of mixed elements must be handled carefully in order to avoid dividing high aspect ratio hexahedrons into pyramids, prisms, or tetrahedrons with angles close to 180º. Thus, for the following turbulent flow example, the adaptation tolerance was very aggressive, ensuring in most cases that the boundary layer elements be divided into elements of the same type, i.e. hexahedrons.

7. EXAMPLE CASES

To illustrate the flexibility of the CTSE sensitivity solver, a number of cases are presented which span a range of applications. The first case examines the sensitivity of predicting the reattachment point behind a backward facing step to the modelling parameters of a selection of turbulence models. The second case examines an H2+O2 combustion model to determine the sensitivity of the specific impulse prediction, due to variability in reaction rate model coefficients. Each of these examples contain additions to the flow solver code which involve complicated mathematical models that would be prohibitive or at best, difficult to include in a sensitivity analysis code using traditional means.

Sensitivity to Turbulence Modelling Parameters

Backward Facing Step

The backward facing step case is a mixed element mesh, using an experimentally determined turbulent flow profile as the inlet flow condition.[34] The Reynolds number for this case is Re= 38428.07 (based on step height) resulting in a fully turbulent boundary layer. The geometry is three dimensional with symmetry boundary conditions at the end planes along the z-coordinate resulting in a slice with a width that is 20% of the step height.

Objective Function

Reattachment length is determined by first identifying the mesh point location on the lower surface, downstream of the step, at which the magnitude of the skin friction coefficient is at a minimum. The immediately adjacent mesh points upstream and downstream of the minimum location are found, in addition to the values of skin friction at these points. From these three data points, a quadratic function is constructed to express skin friction as a function of the x-coordinate variable. The reattachment point (Xr) is taken as the minimum of this function. Defining the objective in this way will result in a discontinuity when the minimum point shifts from one mesh point to another, however when the solution converges, the objective function will smooth out and will be continuous about the point of minimum skin friction. The results of these computations will be compared with the experimentally determined uncertainty [35] of Xr = 7.0±1.0.

Figure 3, shows the streamlines for this flow using the Spalart-Allmaras model. The solution exhibits typical vorticity regions and a boundary layer reattachment point which is about seven step heights downstream from the step. A close up view of the refined mesh near the area of the backward facing step is presented in Figure 4. The full profile of the adapted mesh (after three cycles), shaded with the x-coordinate velocity component is shown in Figure 5.

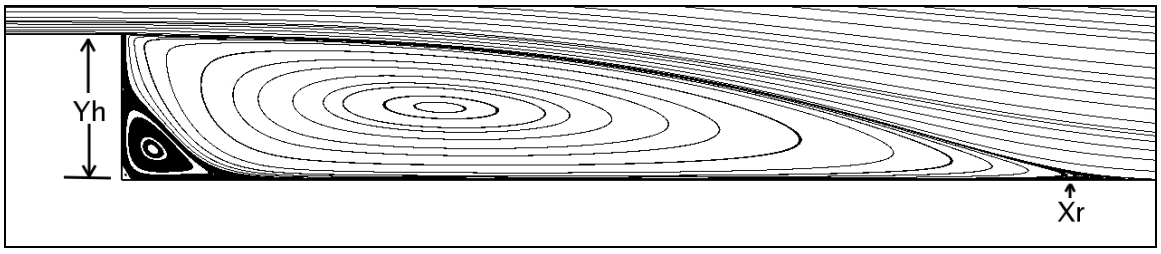

Figure 3: Streamlines behind step with reattachment point at Xr=7.0 Yh

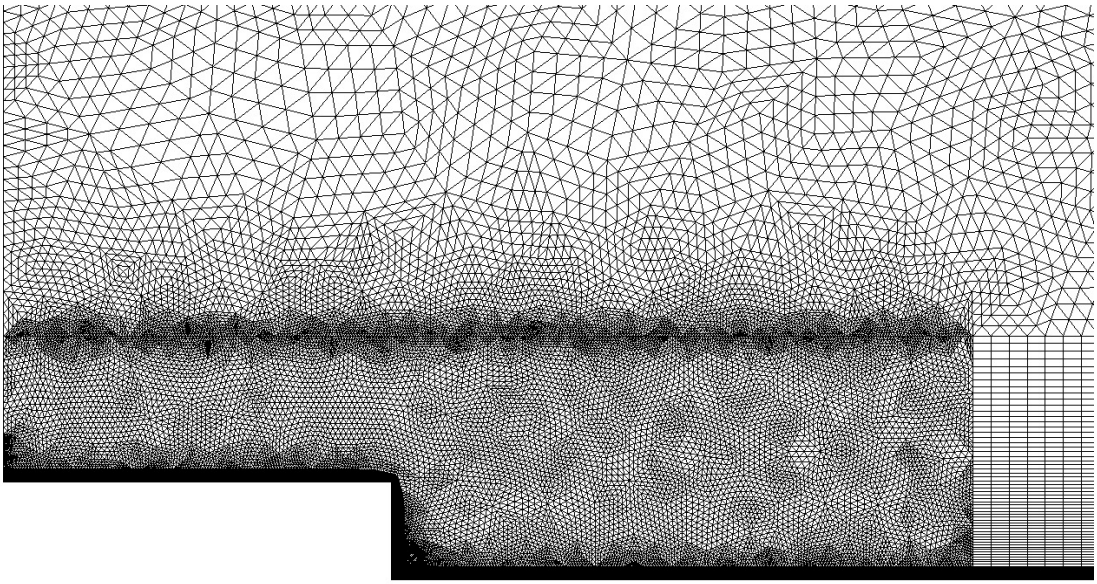

Figure 4: Close-up View of Refined Mesh in Locality of Back Facing Step

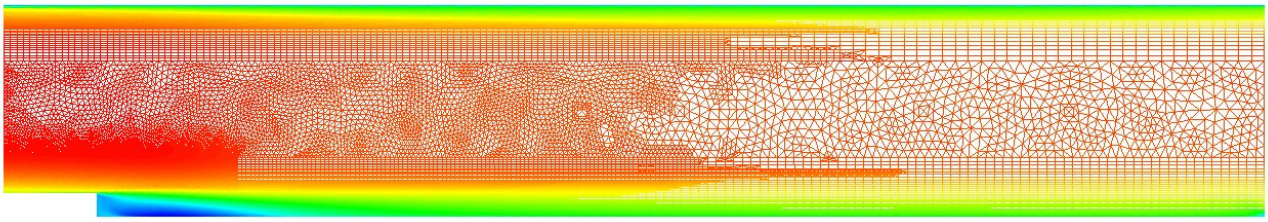

Figure 5: Final Adapted Mesh Shaded With X-Coordinate Velocity Component

Turbulence model parameter uncertainty has been previously studied by others such as Pelletier, Borggaard, et al.[2,3] In their study they use a continuous form of the direct sensitivity method (SEM). This method is similar to the CTSE direct simultaneous method but requires differentiation of the governing flow equations for each design parameter. In addition, also like the CTSE direct method, a separate solution is required for each design parameter. But since the SEM is a continuous method, it will not reflect the exact behaviour of the discrete flow solver until discretization error is minimized. Though all of these direct methods give a wealth of information about the solution, their computational costs must be weighed against the importance of the information. An adjoint solution however can quickly give a measure of the overall sensitivity of an objective, highlighting the relative importance of the various model parameters. With this information in hand, a more detailed analysis via direct differentiation may be performed if necessary.

Several turbulence models are available for use within the Tenasi flow solver. Using the proposed CTSE implementation method, these models have been incorporated into the sensitivity code with minimal effort. The sensitivity solver requires only that the turbulence model codes include templating for complex data types, and follow the coding convention for allowing domain localization. The only coding modification

required by the sensitivity solver is a simple mechanism to control perturbation of the constants in the turbulence model.

The models included in this study are Spalart-Allmaras[36], q-ω[37], and k-ε[38] models. The flow solutions for these cases were converged to a residual on the order of 10^{-14}. Uncertainties have been computed for each of these models on an error adapted mesh with 917,843 nodes. Tables 1, 2, and 3 list the results of the Spalart-Allmaras, q-ω, and k-ε models respectively. The parameter variations listed are either those for which their authors have suggested a range of uncertainty (suggested variance shown in square brackets), or a 10% variance for those without a suggested range.

Param (β)	Value (β)	|Δβ|	dXr/dβ	|ΔXr| for range	Baseline Xr:	7.7060E+00
Re_tilde	38428.07	3.84E+03	-2.91E-05	1.12E-01	Total |ΔXr| :	5.63E-01
Kappa	0.41	4.10E-02	-3.27E+00	3.36E-02		
σ	0.66	[0.60,1.00]=0.34	-6.74E-01	2.292E-01		
Cv1	7.1	[6.90,7.10]=0.2	-1.83E-02	3.660E-03		
Cb1	0.1355	[0.13,0.14]=0.0055	-1.62E+01	8.910E-02		
Cb2	0.622	[0.60,0.70]=0.078	-5.17E-01	4.033E-02		
Cw2	0.3	3.00E-02	-1.92E-01	5.760E-03		
Cw3	2	2.00E-01	2.14E-01	4.280E-02		
Ct3	1.2	[1.00,2.00]=0.8	-4.53E-04	3.620E-04		
Ct4	0.5	[0.50,2.00]=1.5	3.81E-03	5.715E-03		
nu_t_ff	1.341946	1.34E-01	5.73E-03	7.69E-04		

Table 1: Uncertainty in Xr due to Spalart Allmaras Model Parameters

Param (β)	Value (β)	|Δβ|	dXr/dβ	|ΔXr| for range	Baseline Xr:	7.1112E+00
Re_tilde	38428.07	3.84E+03	-5.31E-06	2.04E-02	Total |ΔXr| :	1.66E+00
mut_ff	10	1.00E+00	-5.45E-02	5.45E-02		
cmu	0.09	9.00E-03	-1.39E+01	1.25E-01		
c11	0.5	5.00E-02	1.43E+01	7.14E-01		
c10	0.055	5.50E-03	1.75E+01	9.61E-02		
c2	0.833	8.33E-02	-7.38E+00	6.15E-01		
Prq	2	2.00E-01	-1.02E-01	2.04E-02		
Prw	2	2.00E-01	-3.98E-02	7.96E-03		
alpha_t	0.02	2.00E-03	1.96E+00	3.92E-03		

Table 2: Uncertainty in Xr due to q-ω Model Parameters

Param (β)	Value (β)	|Δβ|	dXr/dβ	|ΔXr| for range	Baseline Xr:	6.5652E+00
Re_tilde	38428.07	3.84E+03	-5.00E-08	1.92E-04	**Total |ΔXr| :**	3.15E-04
mut_ff	10	1.00E-01	8.90E-11	8.90E-12		
k_ff	1.60E-03	1.60E-04	-9.80E-06	1.57E-09		
Cmu	0.09	9.00E-03	-9.90E-09	8.91E-11		
C1	1.44	1.44E-01	-2.17E-07	3.12E-08		
C2	1.92	1.92E-01	1.06E-06	2.04E-07		
a1	1.70E-03	1.70E-04	-5.70E-01	9.69E-05		
a2	1.00E-09	1.00E-09	-8.80E+00	8.80E-09		
a3	5.00E-10	5.00E-11	-2.00E+02	1.00E-08		
Pr_k	1	1.00E-01	2.30E-04	2.30E-05		
Pr_eps	1.3	1.30E-01	-1.80E-05	2.34E-06		

Table 3: Uncertainty in Xr due to k-ε Model Parameters

The results of the uncertainty computations for each model are summarized in Table 4. For the example case presented, uncertainty due to the Spalart-Allmaras model of ±0.56 falls within the experimental range of ±1.0, but the prediction of Xr=7.706 is on the high end of the experimentally determined value of Xr=7.0. The q-ω prediction for Xr=7.11 is close to the experimental value but has a higher uncertainty of ±1.66. Finally the k-ε model has a lower prediction for Xr=6.56, and the uncertainty due to this turbulence model is negligible in relation to experimentally determined values. The respective high and low uncertainty values between q-ω and k-ε models correspond to anecdotal reports regarding data fitting via trial and error model parameter modifications.

Model	Xr	Total |ΔXr|	# Params
Spalart	7.706	5.63E-01	16
q-ω	7.11	1.66E+00	9
k-ε	6.56	3.15E-04	11

Table 4: Model Uncertainty

Sensitivity to Reaction Rate Constants

A well recognized problem exists in the calculation of chemically reacting flows due to the uncertainty in experimentally determined reaction rate constants.[39,40] In particular, the Arrhenius equation, used to compute the rates at which certain multi-step chemical reactions take place, has a large influence on the global equilibrium solution. According to Stephen R. Turns, author of An Introduction to Combustion,[41] "The more reliable rate coefficients often are known no better than within a factor of two, while others may be uncertain by an order of magnitude or more."

A search of the NIST online chemical kinetics database[42] for the equations in the chain of reactions for the following case, found no uncertainties reported for the three parameters used in the Arrhenius equation. Therefore, in this case, uncertainty in the modelling parameters is taken to be 10% of the original value of the parameter. This issue highlights the need for more diligence in reporting not only the uncertainties of analytical and experimental results, but also the uncertainties in the measurements and modeling parameters

used in those obtaining those results. Without the proper parameter variance, computational uncertainty evaluations may be rendered unreliable.

The bimolecular forward reaction rate is computed using the modified (three parameter) form of the Arrhenius equation (14). The empirical parameters in this equation consist of the frequency factor c_f, the activation energy E_a, and an exponent η_f. The values of these constants are approximations and are valid only over a limited range of temperatures.

$$k_f(T) = c_f T^{\eta_f} \exp\left(\frac{-E_a}{R_u T}\right) \quad (14)$$

The example case presented here computes the uncertainty in the specific impulse (ISP) at the exit due to uncertainty in parameter η_f, for each of nine equations in a multi step H2+O2 reaction within a supersonic nozzle geometry. The reference temperature is 3640K, the reference velocity (speed of sound) is 1617.7m/s and the computed exit velocity is Mach 2.98. The surface of the nozzle is considered inviscid for simplicity.

The results of the uncertainty analysis listed in Table 5 illustrate the relative insensitivity of the ISP calculation to each of the exponent parameters for all nine chemical reactions. Note that the ISP listed is un-normalized as is the ΔISP, but the derivatives were computed based on the normalized value of 2.422. The normalization is simply $ISP\frac{c_{ref}}{g}$, where c_{ref} is the reference speed and g is acceleration due to gravity.

Reaction equation	β=ηf	|Δβ|	|dISP/dβ|	|ΔISP|	Baseline ISP:	3.9981E+02
2O+M<=>O2+M	-1	1.00E-01	2.73E-08	4.50E-07	Total |ΔISP| :	1.50E-02
2H+H2<=>2H2	-0.6	6.00E-02	9.31E-06	9.22E-05		
2H+H2O<=>H2+H2O	-1.25	0.125	9.15E-05	1.89E-03		
O+H+M<=>OH+M	-1	0.1	2.39E-06	3.94E-05		
H+OH+M<=>H2O+M	-2	0.2	3.36E-04	1.11E-02		
H+O2<=>O+OH	-6.707	0.6707	1.62E-05	1.80E-03		
O+H2<=>H+OH	2.7	2.70E-01	6.28E-07	2.80E-05		
2OH<=>O+H2O	2.4	2.40E-01	1.17E-07	4.65E-06		
OH+H2<=>H+H2O	1.51	0.151	1.12E-06	2.79E-05		

Table 5: Uncertainty in ISP Due to Reaction Rate Model Parameters

The insensitivity to the exponent parameter is confirmed by computing a direct sensitivity analysis of ISP with respect to the η_f parameter for each of the nine equations. An iterative history of the objective function (solid red line), bracketed by the computed uncertainty bounds (blue dashed line) are plotted in Figure 6. A closer view of this plot is shown in Figure 7.

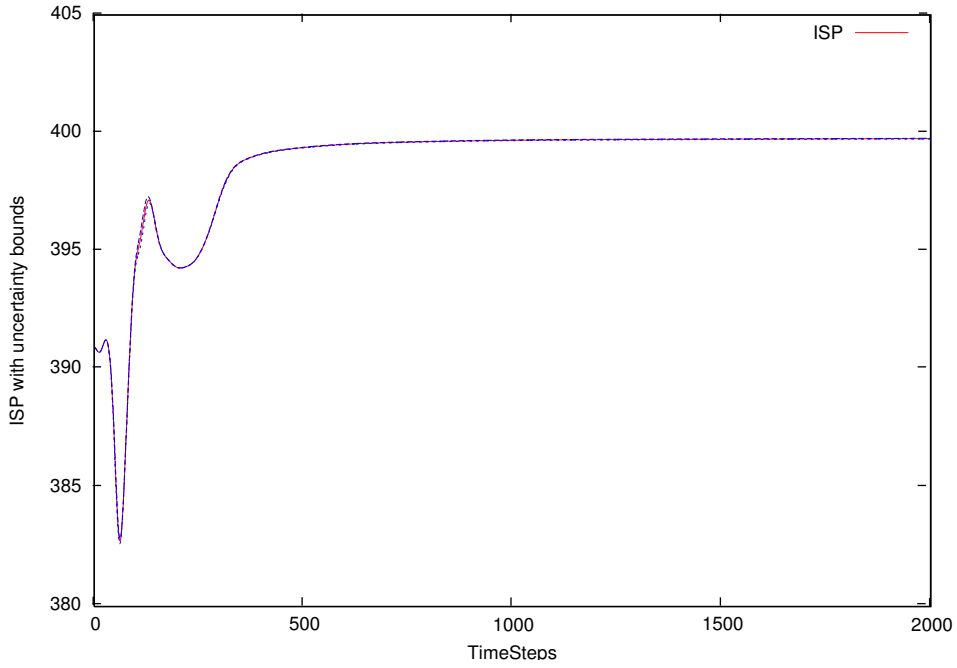

Figure 6: Iterative History of Objective and Uncertainty for Chemically Reactive Flow in Nozzle

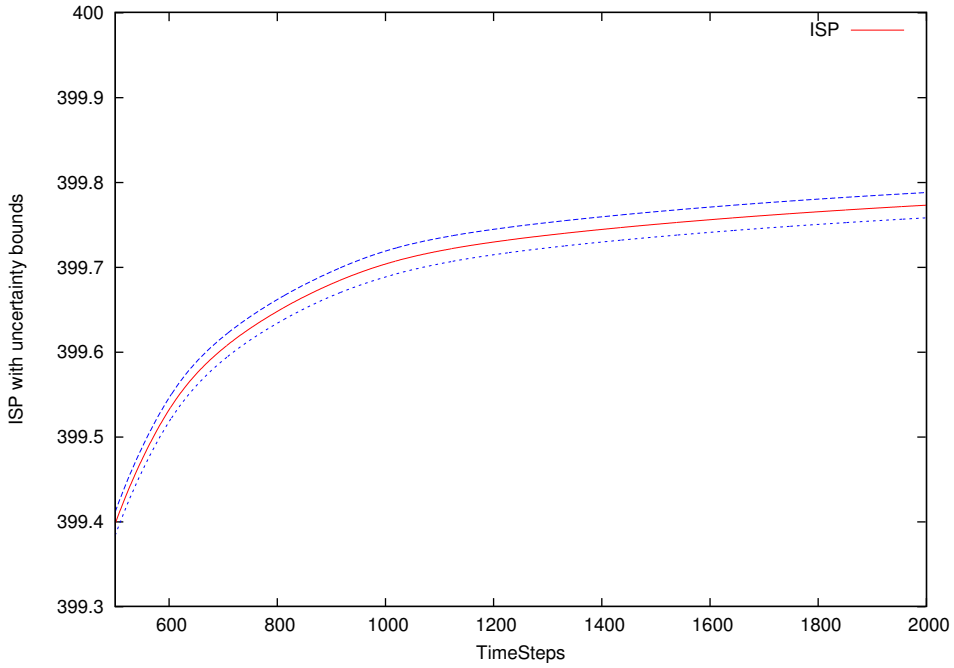

Figure 7: Iterative History of Objective and Uncertainty for Flow in Nozzle (close view)

8. CONCLUSIONS

The adjoint sensitivity method is well suited for computing uncertainties with respect to large numbers of parameters. This capability allows for a wide range of applications, including mesh error estimation, uncertainty analysis, and design optimization. Though the sensitivity method requires the range of uncertainty to be small, influential parameters can be identified for wider variability via Monte Carlo or polynomial chaos methods. Again, a call for more diligence in reporting the uncertainties in measurements and modelling parameters for analytical and experimental data is needed in order for computational uncertainty to be a effective addition to the analysis toolkit. The CTSE adjoint implementation method provides a means for simplifying the heretofore difficult and time consuming task of creating and maintaining an adjoint method for large scale flow solvers.

9. REFERENCES

[1] Wendt, J. F., "External Hypersonic Aerodynamics: State-Of-The-Art and Future Perspectives", Future Aerospace Technology in the Service of the Alliance, Vol. 3 of CP-600. NATO AGARD, pg. C10.1-C10.7, 1997.

[2] Pelletier, D., "Uncertainty Analysis by the Sensitivity Equation Method", 41st Aerospace Sciences Meeting and Exhibit, Reno NV, Jan 6-9, 2003, AIAA Paper, AIAA-2003-412.

[3] Turgeon, E., Pelltier, D., Etienne, S., and Borggaard, J., "Sensitivity and Uncertainty Analysis for Turbulent Flows", 40th AIAA Aerospace Sciences Meeting and Exhibit, Reno NV, Jan 14-17, 2002, AIAA-2002-0985.

[4] Putko, N. M., Newmann, P. A., Taylor, III, A. C., Green, L. L., "Approach for Uncertainty Propagation and Robust Design in CFD Using Sensitivity Derivatives", 15th AIAA Computational Fluid Dynamics Conference, Anaheim CA, Jun 2001, AIAA-2001-2528.

[5] Bischof, C. H., Bucker, H. M., Rasch, A., "Sensitivity Analysis of Turbulence Models Using Automatic Differentiation", *SIAM Journal of Scientific Computing*, 2004, Vol. 26., No. 2, pp 510-522.

[6] Ceperly, D. M., Kalos, M. H., Monte Carlo Methods in Statistical Physics. Ed. K. Binder, Springer-Verlag, New York, 1979.

[7] Alefeld, G., Herzberger, J., Introduction to Interval Computations, Academic Press, New York, 1983.

[8] Ghanem, R. G., Red-Horse, J, "Propagation of Uncertainty in Complex Physical Systems Using a Stochastic Finite Elements Approach," *Physica D*, Vol. 133, pp. 137-144, 1999.

[9] Knio, O. M., Le Maitre, O. P., "Uncertainty Propagation in CFD Using Polynomial Chaos Decomposition", *Fluid Dynamics Research*, Elsevier, Vol. 38, pp 616-640, 2006.

[10] Jameson, A., "Aerodynamic Design via Control Theory." *ICASE* Report, (88-64), 1988.

[11] Elliot, J., Peraire, J., "Practical Three-Dimensional Aerodynamic Design and Optimization Using Unstructured Meshes", *AIAA Journal*, Vol. 35(9), pp. 1479-1487, 1997.

[12] Hou, T., Tadmoor, E., editors. “Discrete Adjoint Approximations With Shocks”, HYP2002, Springer-Verlag, 2002.

[13] Giles, M. B., Duta, M. C., Muller, J., Pierce, N. A., “Algorithm Developments for Discrete Adjoint Methods”, *AIAA Journal*, Vol. 41(2), February 2003.

[14] Haug, E. J., Komkov, V., “Sensitivity Analysis in Distributed Parameter Mechanical Systems Optimization”, *JOTA*, Vol. 23(3), pp. 445-464, 1977.

[15] Arora, J. S., Haug, E. J., “Optimum Structural Design Under Dynamic Loads”, *ASCE Journal of Structural Division*. Vol. 103(10), pp. 2071-2074, 1977.

[16] Jameson, A, “Aerodynamic Design via Control Theory”, ICASE Report, (88-64), 1988.

[17] Sreenivas, K., Hyams, D. G., Nichols, S. D., Mitchell, B., Taylor, L. K., Briley, W. R., Whitfield, D. L., “Development of an Unstructured Parallel Flow Solver for Arbitrary Mach Numbers”, AIAA Paper, (AIAA-2005-0325), 2005.

[18] Giles, M. B., Pierce, N. A., “An Introduction to the Adjoint Approach to Design”, *Flow, Turbulence and Combustion*, Vol. 65, pp 393-415. 2000.

[19] Nielsen, E. J., Anderson, W. K., “Recent Improvements in Aerodynamic Design Optimization on Unstructured Meshes”, AIAA Paper, (AIAA-2001-0596), 2001.

[20] Newman, J. C., Taylor, A. C., “Three Dimensional Aerodynamic Shape Sensitivity Analysis and Design Optimization Using the Euler Equations on Unstructured Grids”, AIAA Paper, (AIAA-1996-2464), 1996.

[21] Sherman, L. L., Taylor, A. C., Green, L. L., Newman, P. A., Hou, G. J., Korivi, V. M., “First and Second Order Aerodynamic Sensitivity Derivatives via Automatic Differentiation with Incremental Iterative Methods”, AIAA Paper, (AIAA-1994-4264), 1994.

[22] Mohammadi, B., “Optimal Shape Design, Reverse Mode of Automatic Differentiation and Turbulence”, AIAA Paper, (AIAA-1997-0099), 1997.

[23] Sundaram, P., Agrawal, S., Hager, J. O., Carle, A., “Viscous Design Optimization Using ADJIFOR – An HPCC Perspective”, Technical report, NASA Ames Research Center, February 2000, HPCC/CAS Workshop.

[24] Burdyshaw, C. E., Anderson, W. K., “A General and Extensible Unstructured Mesh Adjoint Method”, *AIAA Journal of Aerospace, Computing, Information, and Communication*, Vol. 2(10), 2005.

[25] Nielsen, E. J., Kleb, W. L. , “Efficient Construction of Discrete Adjoint Operators on Unstructured Grids Using Complex Variables” , AIAA Paper, AIAA-2005-0324.

[26] Burdyshaw, C. E., “Achieving Automatic Concurrency Between Computational Field Solvers and Adjoint Sensitivity Codes”, Ph.D. Thesis, University Of Tennessee at Chattanooga. May 2006.

[27] Lyness, J. N., Moler, C. B., "Numerical Differentiation of Analytic Functions", *SIAM Journal on Numerical Analysis*, Vol. 4(2), pp. 202-210, June 1967.

[28] Squire, W., Trapp, G., "Using Complex Variables to Estimate Derivatives of Real Functions", *SIAM Review*, Vol. 40(1), pp. 110-112, March 1998.

[29] Newman, J. C., Anderson, W. K., Whitfield, D. L., "Multidisciplinary Sensitivity Derivatives Using Complex Variables", Technical Report, Mississippi State University, (MSSU-COE-ERC98-08), June 1998.

[30] Saad, Y., Schultz, M. H., "GMRES: A Generalized Minimal Residual Algorithm For Solving Nonsymmetric Linear Systems," *SIAM J. Sci. Stat. Comput*, 7(3):856-869, July 1986.

[31] Jones, W. P., Launder, B. E., "The Prediction of Laminarization with a Two-Equation Model of Turbulence," *International Journal of Heat and Mass Transfer*, Vol. 15, 1972, pp. 301-314.

[32] Lee-Rausch, E. M., Park, M. A., Jones, W. T., Hammond, D. P., Nielsen, E. J., "Application of Parallel Adjoint-Based Error Estimate and Anisotropic Grid Adaptation for Three-Dimensional Aerospace Configurations," AIAA Paper, AIAA-2005-4842, June 2005.

[33] Park, M. A., "Adjoint-Based, Three-Dimensional Error Prediction and Grid Adaptation" *AIAA Journal*, Vol. 42, No. 9, September 2004, pp. 1854-1862.

[34] Driver, D. M., Seegmiller, H. L., "Features of a Reattaching Turbulent Shear Layer in Divergent Channel Flow," *AIAA Journal*, Vol. 23, No. 2, February 1985.

[35] Eaton, J.K., Johnston, J.P., "A Review of Research on Subsonic Turbulent Flow Reattachment", *AIAA Journal* Vol. 19, No. 9, September 1981. AIAA-80-1438.

[36] Spalart P.R. Allmaras, S.R., "A One-Equation Turbulence Model for Aerodynamic Flows", 30th Aerospace Sciences Meeting and Exhibit, Jan 6-9 1992, Reno NV. AIAA-92-0439.

[37] Coakley, T. J., "Turbulence Modeling Methods for the Compressible Navier Stokes Equations," AIAA 16th Fluid and Plasma Dynamics Conference, July 12-14, 1983, Danvers, Massachusetts, AIAA-83-1693.

[38] Launder, B. E., and Spalding, J., "The Numerical Computation of Turbulent Flows," *Computers and Methods in Applied Mechanics and Engineering*, 1974, pp. 269-289.

[39] Gupta, Yos, Thompson, Lee, "A Review of Reaction Rates and Thermodynamic Properties for an 11-Species Air Model for Chemical and Thermal Nonequilibrium Calculations to 30,000K.", NASA RP 1232, 1990.

[40] Esch, Siripong, Pike, "Thermodynamic Properties in Polynomial Form for Carbon, Hydrogen, Nitrogen, and Oxygen Systems from 300 to 15000K," NASA TR-70-3, 1970.

[41] Turns, S. R., An Introduction to Combustion: Concepts and Applications. 2nd Edition, McGraw Hill, 2000, ISBN 0-07-230096-5.

[42] NIST chemical kinetics database online: http://kinetics.nist.gov/kinetics/index.jsp

Paper No. 52

Discusser's Name: U. Tabak

Question: Did you experiment in shape sensitivity analysis?

Author's Reply: Yes, the original intent for my adjoint implementation was in determining shape sensitivities to facilitate shape optimization. This is a typical application for the adjoint.

Discusser's Name: N. Kroll

Question: Is there a dependency of a step size h with the CTSE approach?

Author's Reply: There is a truncation error associated with the CTSE approach which is second order with respect to step size h. However, unlike other Taylor series based derivative expressions, there is no lower bound on step size h. So the truncation error vanishes numerically by choosing a step size h which is at least as small as the square root of machine zero for the precision of the floating point representation in your executable code.

Discusser's Name: R. Dwight

Question: How many residual evaluations are needed to obtain adjoint residuals?

Author's Reply: By adjoint residuals, I assume you are referring to linearization of the state variable residual function that's used to compute the adjoint. This is of course dependent on the connectivity of your computational stencil which is defined in part by the element types involved and the spatial order of your solution. An example of a worst case scenario would be a completely tetrahedral mesh arranged in a surface of revolution, or a spherical geometry in which an extremely large number of elements share a vertex. This would result in a very large set of sparsely populated perturbation lists or "colors" which could conceivably lead to a differentiation process in which the number of residual function evaluations needed is equal to the number of nodes in the mesh. This undesirable situation is unlikely to be encountered in practice, as it is indicative of a poor discretization and should be avoided in the solution of the original state variables to begin with. One typical example I can cite is a wing section mesh composed of about 3500 points and multiple element types, hexes around the surface and prisms everywhere else. For this mesh, a first order solution requires approximately 150 residual evaluations to compute the Jacobian, and a second order solution takes close to 300.