DEPARTMENT OF MECHANICAL ENGINEERING
COLLEGE OF ENGINEERING AND TECHNOLOGY
OLD DOMINION UNIVERSITY
NORFOLK, VIRGINIA 23529

# FIRST-AND SECOND-ORDER SENSITIVITY ANALYSIS OF A P-VERSION FINITE ELEMENT EQUATION VIA AUTOMATIC DIFFERENTIATION

**By**
Dr. Gene Hou, Principal Investigator
Department of Mechanical Engineering

**Final Report**

**Prepared for**
NASA Langley Research Center
Attn.: Joseph Murray, Grants Officer
Mail Stop 126
Hampton, VA 2361-0001

**Under**
**NASA Grant Number NAG1-1859**
**ODURF Project Number 162041**

**August 1998**

DEPARTMENT OF MECHANICAL ENGINEERING
COLLEGE OF ENGINEERING AND TECHNOLOGY
OLD DOMINION UNIVERSITY
NORFOLK, VIRGINIA 23529

# FIRST-AND SECOND-ORDER SENSITIVITY ANALYSIS OF A P-VERSION FINITE ELEMENT EQUATION VIA AUTOMATIC DIFFERENTIATION

**By**
Dr. Gene Hou, Principal Investigator
Department of Mechanical Engineering

**Final Report**

**Prepared for**
NASA Langley Research Center
Attn.: Joseph Murray, Grants Officer
Mail Stop 126
Hampton, VA 2361-0001

**Under**
**NASA Grant Number NAG1-1859**
**ODURF Project Number 162041**

**Submitted by**
Old Dominion University Research Foundation
800 West 46th Street
Norfolk, VA 23508

**August 1998**

# FIRST-AND SECOND-ORDER SENSITIVITY ANALYSIS OF A P-VERSION FINITE ELEMENT EQUATION VIA AUTOMATIC DIFFERENTIATION


## Abstract

Sensitivity analysis is a technique for determining derivatives of system responses with respect to design parameters. Among many methods available for sensitivity analysis, automatic differentiation has been proven through many applications in fluid dynamics and structural mechanics to be an accurate and easy method for obtaining derivatives. Nevertheless, the method can be computational expensive and can require a high memory space. This project will apply an automatic differentiation tool, ADIFOR, to a *p*-version finite element code to obtain first- and second-order thermal derivatives, respectively. The focus of the study is on the implementation process and the performance of the ADIFOR-enhanced codes for sensitivity analysis in terms of memory requirement, computational efficiency, and accuracy.


## 1. Introduction

Response derivatives are important to many engineering applications, such as design approximation, design prediction, design optimization, etc. Methods have been developed in the past to calculate response derivatives systematically. Finite differencing, direct differentiation, semi-analytical method, and adjoint variable approach are a few examples that are applicable to the algebraic system equations. Users of those methods are required to generate computer codes according to the sensitivity equations derived by using calculus. Automatic Differentiation (AD), on the other hand, works directly with the computer compiler to generate an enhanced code that includes the derivative information of the responses computed in the input computer code[1,2].

Quite a few publications have documented the experience of using AD tools for sensitivity applications. One AD tool, ADIFOR (Automatic DIfferentiation of FORtran)[3], has been used to differentiate a finite element structural code to obtain first-order derivatives[4]. ADIFOR has also been applied to commercially rated computational fluid dynamics codes to obtain first-order aerodynamic sensitivities[5,6]. Particularly in Ref.7, ADIFOR is not directly applied to the entire computational Fluid Dynamics (CFD) code as a black box; instead, it is applied to the selective subroutines that define the residuals of the aerodynamic algebraic equations. The same procedure is extended to find the second-order aerodynamic derivatives[7]. Another application of ADIFOR can be found in Ref.8 for flutter-speed sensitivity analysis. Valuable observations are made in the paper regarding the use of ADIFOR for sensitivity analysis. These publications[4-8] have illustrated that the ADIFOR-enhanced codes are accurate for sensitivity analysis and can be generated with minimal coding effort. Nevertheless, they observed that the ADIFOR-enhanced codes can be bulky and, compared to the hand-coded sensitivity module, may require more computer resources to compute and store the sensitivity information.

The current work will apply ADIFOR to a *p*-version finite element code for one-dimensional heat transfer. The study will investigate the implementation process, the memory requirement of the ADIFOR-enhanced codes and the computational efficiency and accuracy of the overall sensitivity analysis using the ADIFOR-enhanced codes. The focus is placed particularly upon the second-order sensitivity analysis, which is known for its complexity in derivation and its intensity in computation.

## 2. Problem Formulation

The code of concern is a *p*-version finite element which was written by Bey and Wilson[9]. The code is designed to solve one-dimensional, steady-state heat transfer problems. Besides the conventional constant and linear polynomials, high order polynomials, with zero values and derivatives at the end nodes, are chosen as interpolating functions[10]. In this way, the domain to be analyzed can be discretized in the conventional way, where the nodal temperatures are associated with the constant and linear polynomials. The accuracy of the solution can be improved by increasing the number of "nodeless" degrees of freedom associated with the higher order polynomials.

The finite element equations for the thermal codes can be expressed symbolically as

$$\boldsymbol{K}\boldsymbol{u} = \boldsymbol{f} \tag{1}$$

Where $\boldsymbol{K}$, $\boldsymbol{u}$, and $\boldsymbol{f}$ in represent the heat conduction matrix, the temperature vector, and the vector of external heat generation, respectively. The dimension of the heat conduction matrix $\boldsymbol{K}$ depends upon the order of the polynomials used in the model.

The first-order sensitivity equation can be obtained by differentiating Eq. (1) with respect to an arbitrary design parameter, $b_i$, to obtain

$$\boldsymbol{K}\frac{\partial \boldsymbol{u}}{\partial b_i} = -\frac{\partial \boldsymbol{K}}{\partial b_i}\boldsymbol{u} + \frac{\partial \boldsymbol{f}}{\partial b_i} \tag{2}$$

The process can be continued to obtain the second-order sensitivity equation

$$\boldsymbol{K}\frac{\partial^2 \boldsymbol{u}}{\partial b_i \partial b_j} = \left(-\frac{\partial^2 \boldsymbol{K}}{\partial b_i \partial b_j}\boldsymbol{u}\right) - \left(\frac{\partial \boldsymbol{K}}{\partial b_i}\right)\left(\frac{\partial \boldsymbol{u}}{\partial b_j}\right) - \left(\frac{\partial \boldsymbol{K}}{\partial b_j}\right)\left(\frac{\partial \boldsymbol{u}}{\partial b_i}\right) + \frac{\partial^2 \boldsymbol{f}}{\partial b_i \partial b_j} \tag{3}$$

where $b_j$ is another design parameter.

Equations (2) and (3) provide the values of $\partial \boldsymbol{u}/\partial b_i$ and $\partial^2 \boldsymbol{u}/\partial b_i \partial b_j$ for computing derivatives of an arbitrary performance function, $\psi(\boldsymbol{u})$, required in the gradient-based design optimization process:

$$\frac{\partial \psi}{\partial b_i} = \left(\frac{\partial \psi}{\partial \boldsymbol{u}}\right)^T \left(\frac{\partial \boldsymbol{u}}{\partial b_i}\right) \tag{4}$$

and

$$\frac{\partial^2 \psi}{\partial b_i \partial b_j} = \left(\frac{\partial \boldsymbol{u}}{\partial b_i}\right)^T \left(\frac{\partial^2 \psi}{\partial \boldsymbol{u}^2}\right)\left(\frac{\partial \boldsymbol{u}}{\partial b_j}\right) + \left(\frac{\partial \psi}{\partial \boldsymbol{u}}\right)\frac{\partial^2 \boldsymbol{u}}{\partial b_i \partial b_j} \tag{5}$$

where the superscript $T$ represents the transpose. Those derivatives of $\psi(\boldsymbol{u})$ can also be calculated by the adjoint variable method[14] as

$$\frac{\partial \psi}{\partial b_i} = \lambda^T \frac{\partial \boldsymbol{K}}{\partial b_i}\boldsymbol{u} - \lambda^T \frac{\partial \boldsymbol{f}}{\partial b_i} \tag{6}$$

and

$$\frac{\partial^2 \psi}{\partial b_i \partial b_j} = \left(\frac{\partial \boldsymbol{u}}{\partial b_i}\right)^T \left(\frac{\partial^2 \psi}{\partial \boldsymbol{u}^2}\right)\left(\frac{\partial \boldsymbol{u}}{\partial b_j}\right) + \lambda^T \left(\frac{\partial^2 \boldsymbol{K}}{\partial b_i \partial b_j}\right)\boldsymbol{u} + \lambda^T \left(\frac{\partial \boldsymbol{K}}{\partial b_i}\right)\left(\frac{\partial \boldsymbol{u}}{\partial b_j}\right) + \lambda^T \left(\frac{\partial \boldsymbol{K}}{\partial b_j}\right)\left(\frac{\partial \boldsymbol{u}}{\partial b_i}\right) - \lambda^T \frac{\partial^2 \boldsymbol{f}}{\partial b_i \partial b_j} \tag{7}$$

where the first-order derivative, $\partial \boldsymbol{u}/\partial b_i$ is obtained by Eq. (2) and the adjoint variable vector, $\lambda$, is obtained by solv-

ing the adjoint variable equation

$$\boldsymbol{K}\lambda = -\left(\frac{\partial \psi}{\partial \boldsymbol{u}}\right)^T \tag{8}$$

Note that the sensitivity equations, Eqs. (2) and (3), and the adjoint variable equation, Eq. (8) can be solved with backward substitution, as matrix $\boldsymbol{K}$ has been factorized already when Eq. (1) is solved. Nevertheless, solving sensitivity equations remains a difficult computational task. Many new derivative terms not found in analysis are needed now to construct either the right-hand sides of the sensitivity equations, Eqs. (2) and (3), or the derivatives of Eqs. (6) and (7). These new terms include the derivative $\partial \boldsymbol{K}/\partial b_i$, a three-dimensional matrix, and the derivative $\partial^2 \boldsymbol{K}/\partial b_i \partial b_j$, a four-dimensional matrix. Furthermore, the number of sensitivity equations to be solved can be large. In fact, $m$ and $(m(m+1)/2)$ numbers of linear equations are needed respectively to solve for the first- and second-order derivatives of $\boldsymbol{u}$ or $\psi(\boldsymbol{u})$ in Eqs. (2)-(5), where $m$ is the number of the design parameters. As for the adjoint variable method, $n$ and $(m+n)$ numbers of linear equations are needed in Eqs. (6)-(8), where $n$ is the number of the performance functions. Although using the adjoint variable method may reduce the number of linear equations solved for sensitivity analysis in the case where $n$ is less than $m$, computing the new derivative terms in the sensitivity equations is still required.

Though the design parameters, $b_i$, are directly related to the stiffness matrix, and may also be dependent on each other. In fact, they may be "linked" to the true design variables, $\boldsymbol{x}$, through a pre-determined relation, $\boldsymbol{b}(\boldsymbol{x})$. In this case, for a specific design variable, $x_i$, Eqs. (2) and (3) are rewritten as

$$\boldsymbol{K}\frac{d\boldsymbol{u}}{dx_i} = -\left(\frac{d\boldsymbol{K}}{d\boldsymbol{b}}\boldsymbol{u}\right)s_i + \frac{d\boldsymbol{f}}{d\boldsymbol{b}}s_i \tag{9}$$

and

$$\boldsymbol{K}\frac{\partial^2 \boldsymbol{u}}{\partial x_i \partial x_j} = -\left(\left(\frac{d^2\boldsymbol{K}}{d\boldsymbol{b}^2}\boldsymbol{u}\right)s_i\right)s_j - \left(\left(\frac{d\boldsymbol{K}}{d\boldsymbol{b}}\right)s_i\right)\frac{d\boldsymbol{u}}{dx_j} - \left(\left(\frac{d\boldsymbol{K}}{d\boldsymbol{b}}\right)s_j\right)\frac{d\boldsymbol{u}}{dx_i} + \left(\frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{b}^2}s_i\right)s_j \tag{10}$$

where $s_i$ denotes $\partial \boldsymbol{b}/\partial x_i$, which is often called the seed matrix in the ADIFOR literature and is usually provided by the user. Equations (6) and (7) can be modified in a similar manner to account for the design variable linkage.

## 3. Implementation Procedure

The main task of implementation is to use ADIFOR to construct the new derivative terms in the sensitivity equations. ADIFOR is used here selectively to minimize the coding effort and to minimize the memory requirement while maximizing the computational efficiency of the ADIFOR-enhanced code. The step-by-step procedure is described below for the first-and second-order sensitivity analyses. The procedure is the same for every design variable; hence, explanation of each step is given only for the shape design variables.

The first step involves the collection of the FORTRAN subroutines that compute the elemental stiffness matrix, $\boldsymbol{K}_e$. These subroutines are input to ADIFOR to obtain the derivatives of the element stiffness matrix with respect to the nodal coordinates, $\boldsymbol{b}_e$, of a typical element, which are specified as independent variables. The output of ADIFOR gives a collection of new subroutines that can compute $\boldsymbol{K}_e$ as well as $((d\boldsymbol{K}_e/d\boldsymbol{b}_e)\ \boldsymbol{S})$ where $\boldsymbol{S}$ is a user-specified seed matrix. In this step, $\boldsymbol{S}$ is fixed as an identity matrix with which the ADIFOR-produced code computes $d\boldsymbol{K}_e/d\boldsymbol{b}_e$ as a three- dimensional array.

The second step generates a new subroutine to compute the product $(d\boldsymbol{K}_e/d\boldsymbol{b}_e)\boldsymbol{u}_e$ where $\boldsymbol{u}_e$ is the solution of Eq. (1) pertaining to the element under consideration. The product is a two-dimensional array.

The third step generates another new subroutine to compute the product $((dK_e/db_e)u_e)S_e$ where $S_e$ is a user specified seed matrix, $db_e/dx$, derivatives of nodal coordinates with respect to independent geometric design variables.

The fourth step assembles the element arrays produced in the third step into global arrays which represent the first term on the right-hand side of Eq. (9).

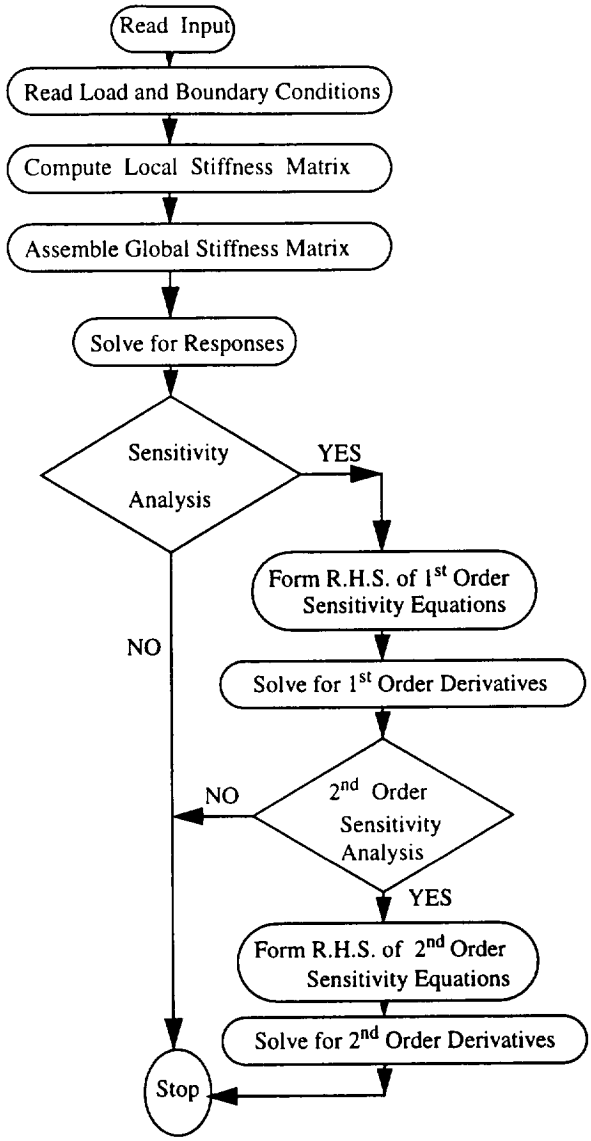



**Fig.1 Computational procedure for analysis and sensitivity analysis.**

The second term on the right-hand side of Eq.(9) and the second and the third terms of Eq. (10) can be constructed in a similar manner. The above procedure, however, needs further work in order to compute the most complicated term: the first term on the right-hand side of Eq. (10). One may proceed to submit the code produced in step one to ADIFOR once more to obtain the second- order derivatives with a seed matrix, $(d^2K_e/db_e^2)S_e$. Again the seed matrix is specified as an identity matrix. The $u_e$ is then multiplied to the second-order derivatives as done in the above

Step two. The resultant three dimensional arrays are then multiplied by $S_e$ twice to obtain $(((d^2K_e/db_e^2)u_e)S_e)S_e$ where $S_e$ is $db_e/dx$. The above fourth step is then followed to complete the first term on the right-hand side of Eq. (10). The code to compute the fourth term in Eq. (10) can be generated in a similar manner.

Once the computer codes that compute the right-hand sides of Eqs. (9) and (10) are assembled, they can then be attached to the original analysis code as a separate module for sensitivity analysis. Such an arrangement is depicted in the flow chart of Fig. 1.

To study the performance of ADIFOR, numerical experiment is done to compare memory requirement and computational efficiency of using the ADIFOR-produced code versus the finite difference method for sensitivity analysis. The results are listed in Table 1 for the first- and second-order shape derivatives of a one-dimensional, *p*-version heat conduction element. In the tables, the memory requirement is obtained by using the "size" command, a UNIX command on an Ultra Sun, which gives the memory required to execute the code; the number of independent design variables is equal to the number of nodes in an element. As the x-coordinates of the nodes are considered as the design variable, only a minimal increment in computer memory and time is observed in Table 1, for heat transfer elements. This is because, as a one-dimensional problem, the number of design variables is limited and this element does not involve coordinate transformation.

In summary, the procedure presented here can effectively use ADIFOR to produce a code to calculate the derivative terms required for the first-and second- order thermal sensitivity analyses.

## Table 1. Comparison of ADIFOR with finite difference for 1-D, *p*-version element

| Element type | Memory before ADIFORing (Bytes) | Memory after ADIFORing for first derivatives (Bytes) | Memory after ADIFORing for second derivatives (Bytes) | CPU time for stiffness matrix evaluation of one element (sec) | CPU time taken by ADIFOR for first derivatives (sec) | CPU time taken by ADIFOR for second derivatives (sec) | Number of independent variables |
|---|---|---|---|---|---|---|---|
| 1-D *p*-version element | 13,706 | 15,496 | 20,864 | 0.02 | 0.03 | 0.04 | 2 |

## 4. Numerical Studies

The heat transfer problem studied here is one-dimensional and is modeled by *p*-version finite elements. It is used mainly to verify the accuracy of the sensitivity analyses, as it is simple and its exact derivatives can be analytically obtained.

The steady-state heat transfer in one dimension is governed by the equation describing the conservation of the energy

$$\frac{d}{dx}\left(-k\frac{dT}{dx}\right) = q(x) \tag{11}$$

where $k$ is the thermal conductivity, $T$ is the temperature, and $q$ is the internal heat generation. In this study, the internal heat generation term is devised as

$$q = \frac{-2k\alpha(-1-\alpha^2 x+\alpha^2 x_o+\alpha^2 x x_o-\alpha^2 x_o^2)}{(1+\alpha^2 x^2-2\alpha^2 x x_o+\alpha^2 x_o^2)^2} \tag{12}$$

where the parameter $\alpha$ controls the steepness of the solution at $x_o$. As a result, the temperature distribution can be analytically solved by

$$T(x) = (l-x)(\tan^{-1}\alpha(x-x_o)+\tan^{-1}(\alpha x_o)) \tag{13}$$

which satisfies the boundary conditions, $T(0) = T(l) = 0$. The value of $l$, $\alpha$ and $x_o$ are specified as 1, 20 and 0.5, respectively.

The analytical expressions of the first and second-order derivatives of T(x), with respect to the length of the domain, $l$, can then be derived as

$$\frac{d}{dl}T(x) = (l-x)\left[\frac{\alpha\left(\frac{x}{l}-\frac{1}{2}\right)}{1+\left(\alpha\left(x-\frac{l}{2}\right)\right)^2}+\frac{\alpha}{2+\alpha^2 l^2}\right]+\left[\tan^{-1}\left(\alpha\left(x-\frac{l}{2}\right)\right)+\tan^{-1}\left(\frac{\alpha l}{2}\right)\right]\left(1-\frac{x}{l}\right) \tag{14}$$

and

$$\frac{d^2}{dl^2}T(x) = -(l-x)\left[\frac{2\alpha\left(\frac{x}{l}-\frac{1}{2}\right)^2\left(x-\frac{l}{2}\right)}{\left(1+\left(\alpha\left(x-\frac{l}{2}\right)\right)^2\right)^2}+\frac{\alpha^3 l}{\left(1+\frac{x^2 l^2}{4}\right)^2}\right]$$
$$+2\left(1-\frac{x}{l}\right)\left[\frac{\alpha\left(\frac{x}{l}-\frac{1}{2}\right)}{1+\left(\alpha\left(x-\frac{l}{2}\right)\right)^2}+\frac{\frac{\alpha}{2}}{1+\left(\frac{\alpha l}{2}\right)^2}\right] \tag{15}$$

where $x_o$ is replaced by $l/2$. The change of domain, $l$, will induce changes in the locations of the mesh nodes and the location of the steepest temperature, $x_o$.

The $p$-version element is defined over a two-node, one-dimensional domain where the temperature distribution is interpolated as

$$\hat{T}(x) = \sum_{i=0}^{p} N_i(x) T_i \tag{16}$$

Using the shape functions, $N_i$ of Demkowicz et al.:[10]

$$N_i(\xi) = \begin{cases} 0.5(1-\xi), & i = 0 \\ 0.5(1+\xi), & i = 1 \\ \xi^i - 1, & i = 2, 4, 6, \ldots \\ \xi^i - \xi, & i = 3, 5, 7, \ldots \end{cases} \tag{17}$$

where $\xi$ is a mapping of the element coordinates onto a standard domain, $\xi \in [-1, 1]$. Note that the higher order shape functions have zero values at the end nodes. Hence, the coefficients of the linear shape functions, $T_0$ and $T_1$, commonly used in an *h*-version heat transfer element, do represent the temperatures at the nodes, while the coefficients of higher order shape functions do not have physical meaning. It should also be noted that T(x) is a $C^0$ function throughout the temperature domain.

The Galerkin method can be used here to obtain the matrix equation that approximates the solution of Eq. (11).

$$\boldsymbol{K}\boldsymbol{T} = \boldsymbol{q} \tag{18}$$

As described in Section 3, once a computer code is developed to solve the above equation, the ADIFOR-produced code can be used to solve Eqs. (9) and (10) for first- and the second-order derivatives, $d\boldsymbol{T}/d\boldsymbol{b}$ and $d^2\boldsymbol{T}/d\boldsymbol{b}^2$, of the discrete temperature vector, $\boldsymbol{T}$.

To study accuracy, the heat transfer problem is discretized into 10 elements with 11 nodes. The total length, $l$, of the domain, is considered to be the design variable. The analytical shape derivatives of the temperature distribution with respect to the length of the domain can be found by Eqs.(14)-(15), which are plotted in Fig.2. On the other hand, the continuous expressions of the discrete temperature derivatives, $d\hat{T}/dl$ and $d^2\hat{T}/dl^2$, can be interpolated by

$$\frac{d}{dl}\hat{T}(x) = \sum_{i=0}^{p} N_i(x)\frac{dT_i}{dl} \tag{19}$$

and

$$\frac{d^2\hat{T}}{dl^2}(x) = \sum_{i=0}^{p} N_i(x)\frac{d^2T_i}{dl^2} \tag{20}$$

where $N_i$ is defined in Eqs. (7). The result shows in Fig. 2 that with the order of polynomials in $N_i$ greater than 3, the continuous expressions of the numerical derivatives are matched very well with those of analytical derivatives.

The equations for the errors between the exact derivatives, dT/d$l$ and $d^2T/dl^2$, and the *p*-version finite element solutions, $\frac{d}{dl}\hat{T}$ and $\frac{\partial^2}{\partial l^2}\hat{T}$, are given by

$$\dot{e}(x) = \frac{d}{dl}T(x) - \frac{d}{dl}\hat{T}(x) \tag{21}$$

$$\ddot{e}(x) = \frac{d^2}{dl^2}T(x) - \frac{d^2}{dl^2}\hat{T}(x) \tag{22}$$

To study the convergence of the finite element solution and derivatives with different meshes and approximation, the $H^1$ norms of the errors are introduced. For example, the $H^1$ norm of the error of the first order derivative is defined as

$$\|\dot{e}\|_1 = \sqrt{\sum_k \|\dot{e}\|_{1,k}^2} \tag{23}$$

where $\|\dot{e}\|_{1,k}$ is an error in element k given by

$$\|\dot{e}\|_{1,k} = \sqrt{\left(\int_{\Omega_k}\left[\left(\frac{d}{dx}\dot{e}\right)^2 + \dot{e}^2\right]dx\right)} \tag{24}$$

similar definitions can be made for $\|e\|_1$ and $\|\dot{e}\|_1$. Three figurers are presented for errors of each type of solutions. The first two figures indicate the total $H^1$ errors of the solution when one of the parameters; the order of polynomials in approximation or the number of meshes, increases and the other remains unchanged. The third one is a picture of elemental errors in individual elements while the mesh size is being changed.

Figures 4 to 6 show the convergence of the temperature solution. As discussed in Ref. 9, it is observed that convergence rate of *p*-version is faster than that of h-version finite element. Furthermore, Fig.6, shows that elemental errors are not distributed uniformly. Particularly, high error is found near the neighborhood of high temperature gradient.

Figures 7 to 9 show the convergences of the first-order temperature derivatives. The trend of the convergent rate of the temperature derivatives is similar to that of the temperature. Again, high error in temperature derivative is found near the neighborhood of high temperature gradient. However, it seems that the convergent rate of the temperature derivative is one order slower than that of the temperature solution and reducing mesh size may not improve the accuracy of the temperature derivative in some part of the domain, as shown in Fig. 9.

Figures 10 to 12 show that neither reduction of mesh sizes nor increase of polynomial order can improve the convergence of the second-order temperature derivatives, measured by the $H^1$ error norm. This results are unexpected and require further study to fully understand such decay of convergence in high-order derivatives.

## 5. Concluding Remarks

The paper first proposes an implementation procedure to develop an ADIFOR-enhanced code for first- and second-order sensitivity analysis and then investigates the performance of the resultant code in terms of memory requirement, computational accuracy and efficiency. The results have demonstrated that an user can follow the proposed procedure to develop an ADIFOR-enhanced code for calculating first-order and second-order derivatives without much knowledge of the original code and with only a minimal coding effort. The resultant code can calculate the first-order derivatives very efficiently. However, it needs a moderate increase of computer resources to support the calculation of the second-order derivatives. The slow-down of the second-order sensitivity analysis is mainly caused by the computation of many second-order derivatives of elemental stiffness matrices, appearing on the right-hand side of the second-order sensitivity equation. The error analysis concludes that the convergence of the temperature derivatives show the same trend as that of the temperature solution. However, the convergence rate of the former is slower than that of the latter. Furthermore, in the cases studied, the h-adaptiveness fails to converge the temperature derivatives. Further research is needed to fully understand the error behavior of the temperature derivatives of which are calculated by the *p*-version finite elements.

## References


[1]Griewank, A. and Corliss, G. G., Eds., *Automatic Differentiation of Algorithms: Theory, Implementa-tion, and Applications* SIAM, Philadelphia, 1991.

[2]Griewank, A., "On Automatic Differentiation," *Mathematical Programming: Recent Developments and Applications*, M. Iri and K. Tanabe,eds., Kluwer Academic Press, Boston, 1989, pp. 83-108.

[3]Bischof, C. H. and Griewank, A., "ADIFOR: A Fortran System for Portable Automatic Differentia-tion," *Fourth AIAA/USAF/NASA/OAI Symposiumon Multidisciplinary Optimization*, AIAA 92-4744-CP, Cleveland, Sept. 1992, pp. 433-441.

[4]Barthelemy, J.-F.M. and Hall, L. E., "Automatic Differentiation as a Tool in Engineering Design," *Structural Optimization*, Vol. 9, 1995, pp. 76-82.

[5]Green, P., Newman, P. and Haigler, K., "Sensitivity Derivatives for Advanced CFD Algorithm and Vis-cous Modelling Parameters via Automatic Differentiation, 1993; *Journal of Computa-tional . Physics*, Vol. 125, No.2, 1996, pp. 313-324.

[6]Green, L., Bischof, C., Carle, A., Griewank, A., Haigler, K. and Newman, P., "Automatic Differentiation of Advanced CFD Codes With Respect to Wing Geometry Parameters for MDO," *Abstracts from Second U.S. National Congress on Computational Mechanics*, Washington, D.C., August, 1993, p. 136.

[7]Sherman, L. L., Taylor, A. C. III, Green, L. L., Newman, P. A., Hou, G. J.-W. and Korivi, V. M., "First- and Second-Order Aerodynamic Sensitivity Derivatives via Automatic Differentiation with Incremental Iterative Methods," *Journal of Compu-tational Physics*, Vol. 129, 1996, pp. 307 – 331.

[8]Issac, J. C. and Kapania, R. K., " Aeroelastic Sensi-tivity Analysis of Wings Using Automatic Differen-tiation," *AIAA Journal*, Vol. 35, No. 3, 1997, pp. 519-525.

[9]Bey, K.S., and Wilson, N. M., "Finite Element Thermal Modeling Using Hierarchial Shape Functions," private communication, NASA Langley, Hampton, VA.

[10]Demkowicz, L, Oden, J. T., Rachowicz, W., and Hardy, O., "Toward a Universal *h-p* Adaptive Finite Element Strategy, Part I Constrained Approximation and Data Structure," *Computer Methods in Applied Mechanics and Engineering*, Vol.77, 1989, pp. 79-112.

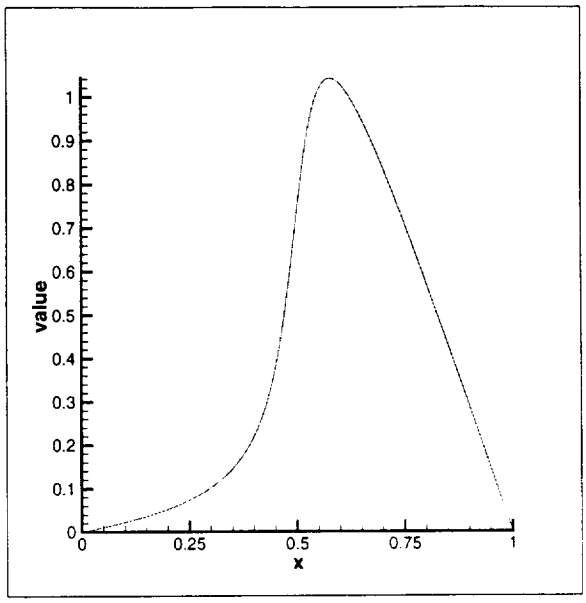
[11]Nguyen, D. T.,"Finite Element Software for Multidisplinary Design Optimization," Final Report, Master Contract NAS1-19858, NASA Langley Research Center, 1995

[12]Tessler, A., "A $C^0$ Anisoparametric Three-Node Shallow Shell Element For General Shell Analysis", U.S. Materials Technology Laboratory, Mechanics & Structures Branch, MTL TR 89-72, Watertown, MA Aug. 1989.

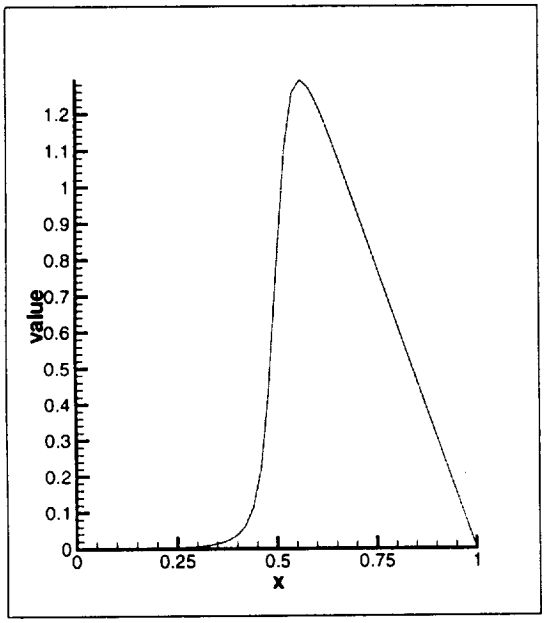
[13]Krishnamoorthy, C. S., *Finite Element Analysis: Theory and Programming*, $2^{nd}$ , Tata McGraw-Hill Publishing Company Limited, New Delhi, India, 1987

[14]Haug,E.J., Choi, K.K. and Komkov, V., *Design Sensitivity Analysis of Structural Systems*, Academic press, New York, N.Y., 1986
*Interantional Journal*, 1998.

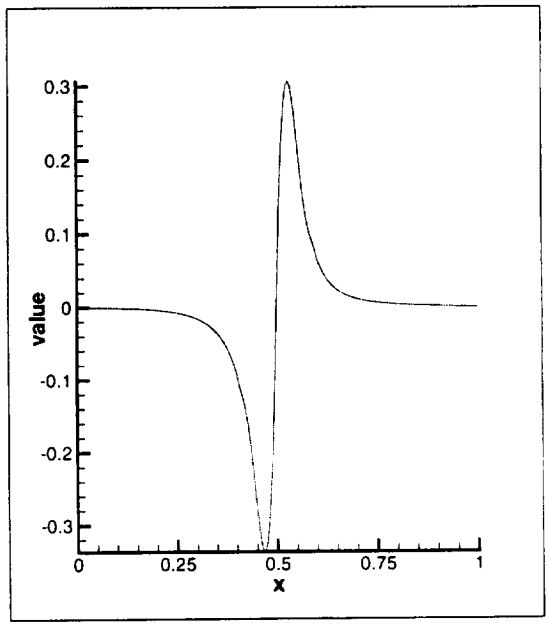
[15]Newman, J.C. III, Newman, P.A, Taylor, A.C.III and Hou, G.J-W., "Efficient Nonlinear Static Aeroelastic Wing Analysis", to appear in *Computers and Fluids*,




(a) Temperature distribution




(b) First order derivative




(c) Second order derivative

**Fig. 3. Analysis and sensitivity analysis of one-dimensional heat transfer problem.**

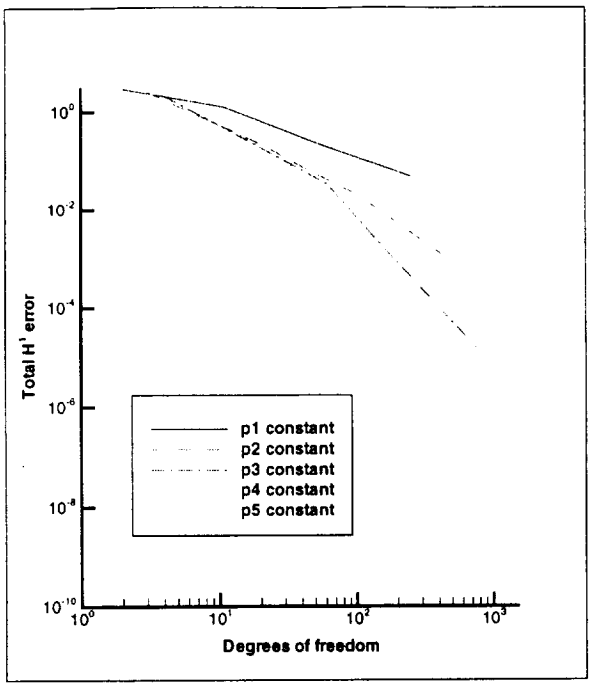



Fig. 4: Total $H^1$ error keeping p fixed and uniformly increasing h

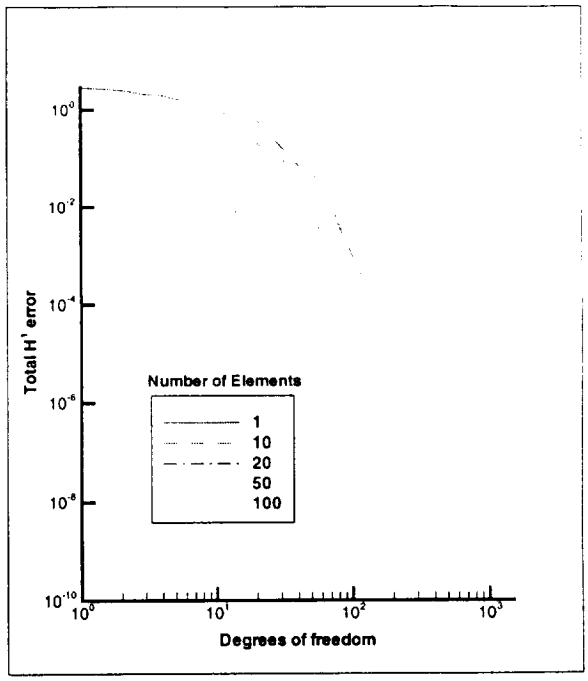



Fig. 5: Total $H^1$ error keeping h fixed and uniformly increasing p

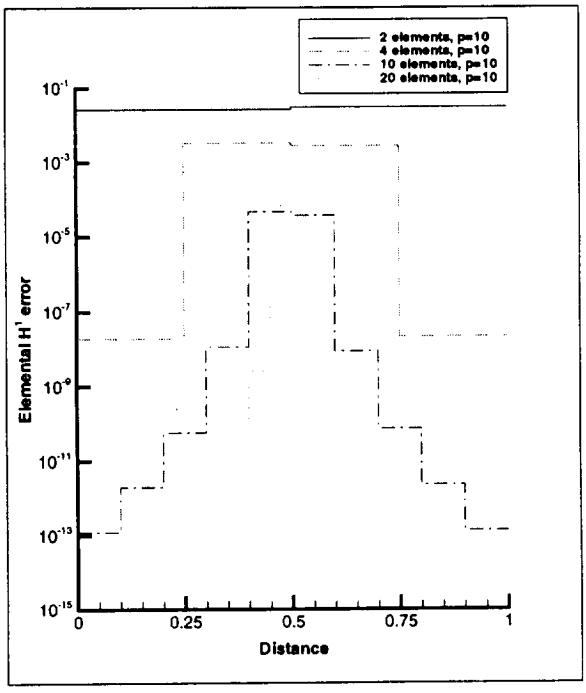



Fig. 6: Elemental error

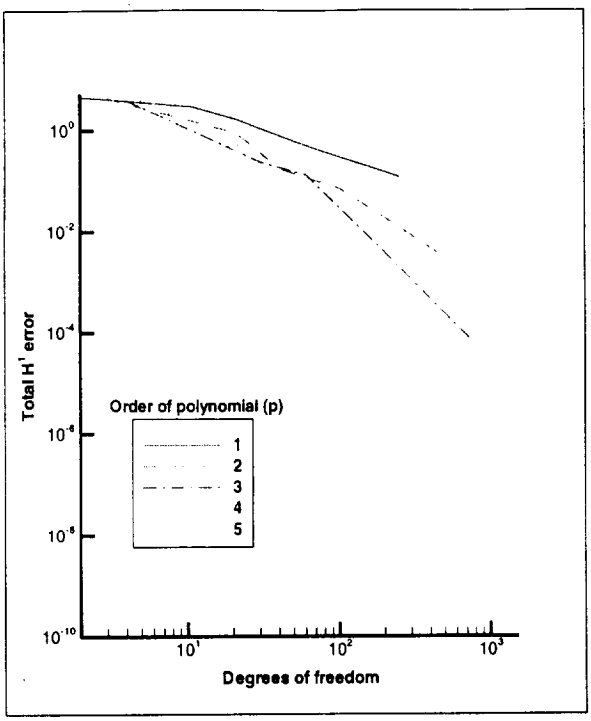



Fig. 7: Total $H^1$ error keeping p fixed and uniformly increasing h

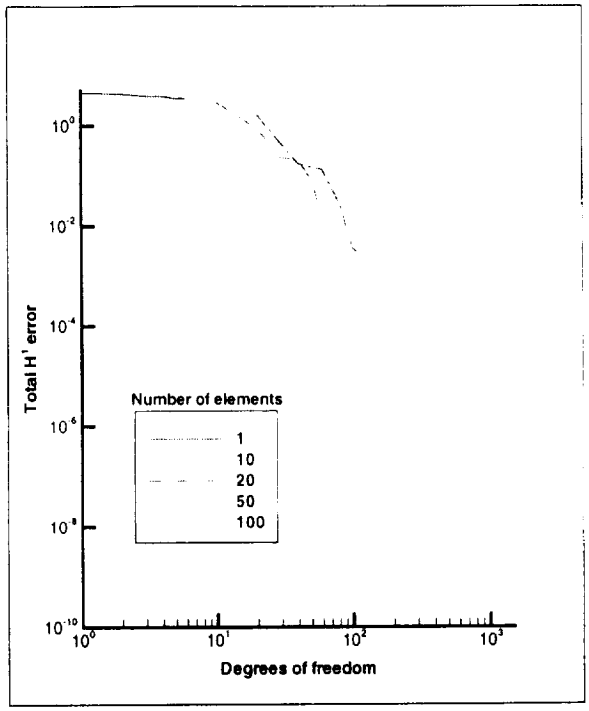



Fig. 8: Total $H^1$ error keeping h fixed and uniformly increasing p

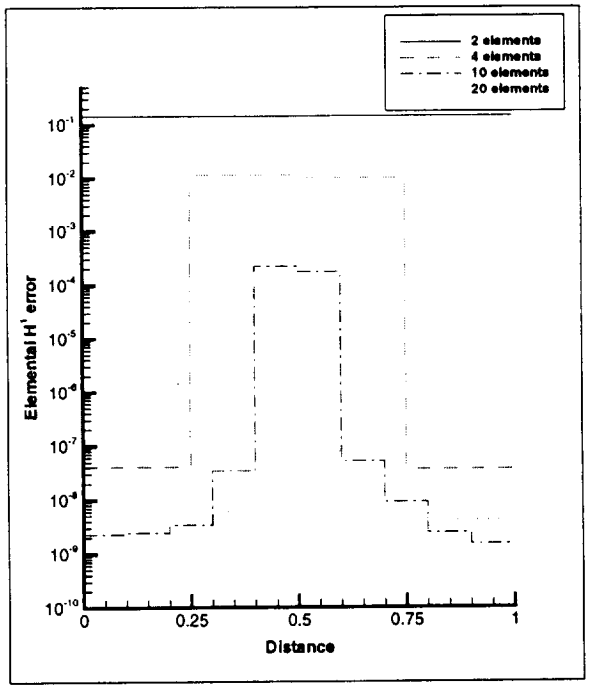



Fig. 9: Elemental error

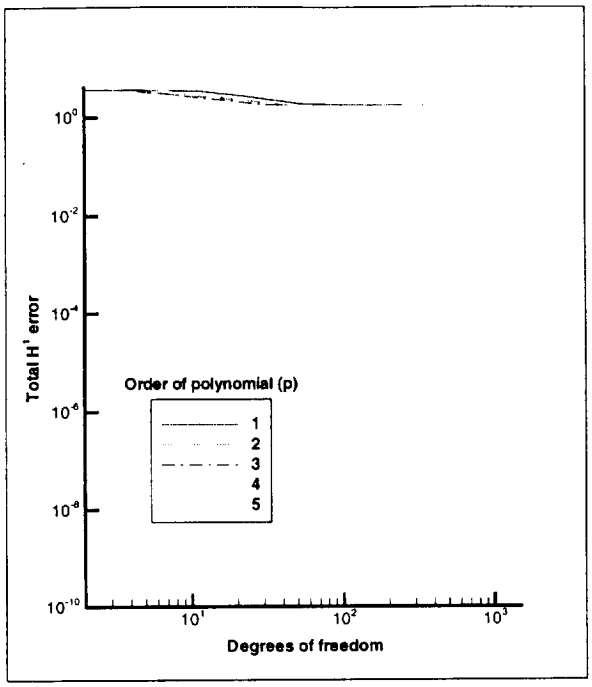



Fig. 10: Total $H^1$ error keeping p fixed and uniformly increasing h

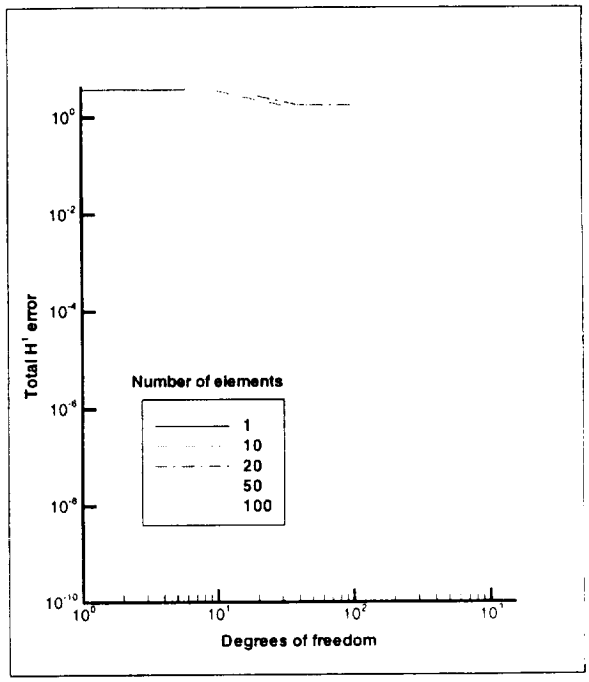



Fig. 11: Total $H^1$ error keeping h fixed and uniformly increasing p

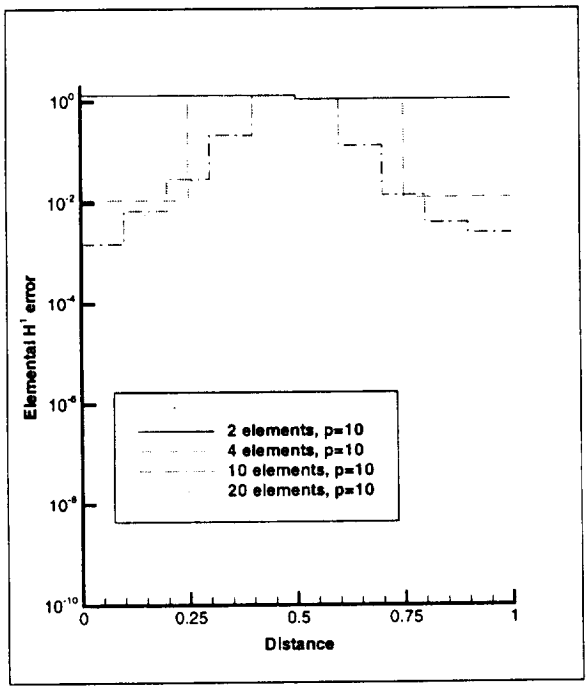



Fig. 12: Elemental error