110047

N95- 16462

Automatic Differentiation for Design Sensitivity Analysis of Structural Systems Using Multiple Processors

Duc T. Nguyen*, Olaf O. Storaasli†, Jiangning Qin*, and Ramzi Qamar*
Multidiscplinary Design Optimization Branch
Fluid Mechanics and Acoustics Division

Automatic differentiation tools (ADIFOR) is incorporated into a finite element based structural analysis program for shape and non-shape design sensitivity analysis of structural systems. The entire analysis and sensitivity procedures are parallelized and vectorized for high performance computation. Small-scale examples to verify the accuracy of the proposed program and a medium-scale example to demonstrate the parallel-vector performance on the multeiple Cray-C90- processors are included in the paper.

* Multidisciplinary Parallel-Vector Computation Center, 135 KDH Building, Old Dominion University, Norfolk VA 23529-0241

† Computational Mechanics Branch, NASA Langley Research Center, Hampton, VA 23681

Automatic Differentiation for Design Sensitivity Analysis of Structural Systems Using Multiple Processors

by

Duc T. Nguyen[†], Olaf O. Storaasli[§], Jiangning Qin[†], and Ramzi Qamar[†]

[†] Multidisciplinary Parallel-Vector Computation Center, 135 KDH Building, Old Dominion University, Norfolk, VA 23529-0241

[§] Computational Mechanics Branch, NASA Langley Research Center, Hampton, VA 23681

Abstract

Automatic differentiation tools (ADIFOR) is incorporated into a finite element based structural analysis program for shape and non-shape design sensitivity analysis of structural systems. The entire analysis and sensitivity procedures are parallelized and vectorized for high-performance computation. Small-scale examples to verify the accuracy of the proposed program and a medium-scale example to demonstrate the parallel-vector performance on the multiple Cray-C90 processors are included in the paper.

I. Introduction

Using the familiar finite element procedure[1], the static equilibrium equations for a structural model can be expressed as

$$[K(b)]_{n x n} \{z\}_{n x 1} = \{F\}_{n x 1} \tag{1}$$

where [K (b)], {z} and {F} are referred to the stiffness matrix, nodal displacement vector and nodal force vector, respectively. In Eq. (1), "n" represents the active degree-of-freedom of the discretized structural model.

The stiffness matrix [K (b)], in general, is a function of design variable vector {b} (where b $\in R^k$). As an example, {b} may represent the cross-sectional areas of various truss members, or thickness of plate members (for non-shape type of design variables), or it may also represent the joint coordinates of various nodes of a structure (for shape type of design variables).

A typical constraint, involving a limit on a displacement or a stress component, may be written as

$$g(z, b) \leq 0 \tag{2}$$

For the sake of simplified notation, it is assumed that g depends on only a simple design variable b (i.e. b $\in R^{k=1}$). Using the chain rule of differentiation, one obtains

$$\frac{dg}{db} = \frac{\partial g}{\partial b} + x^T \frac{dz}{db} \tag{3}$$

where x is a vector with components

$$x_i = \frac{\partial g}{\partial z_i} \tag{4}$$

The first term on the right-hand-side of Eq. (3) is usually zero or easy to obtain, thus one discusses only the computation of the second term.

Differentiating Eq. (1) with respect to b, one obtains

$$K * \frac{dz}{db} = \frac{\partial F}{\partial b} - \frac{dK}{db} * z \tag{5}$$

Premultiplying Eq. (5) by $x^T K^{-1}$, one obtains

$$x^T \frac{dz}{db} = x^T K^{-1} \left(\frac{\partial F}{\partial b} - \frac{dK}{db} * z \right) \tag{6}$$

Numerically, the computation of $x^T \frac{dz}{db}$ can be performed in two different ways. The first, called the "direct method", consists of solving Eq. (5) for $\frac{dz}{db}$ and then taking the scalar product with x. The second approach, called the "adjoint method"[2, 3], defines an adjoint vector λ which is the solution of the system

$$K \lambda = x \tag{7}$$

or

$$\lambda = K^{-1} x \tag{8}$$

or

$$\lambda^T = x^T K^{-1} \quad (\text{since matrix K is symmetric}) \tag{9}$$

and thus, Eq. (3) can be re-written as

$$\frac{dg}{db} = \frac{\partial g}{\partial b} + \lambda^T \left(\frac{\partial F}{\partial b} - \frac{dK}{db} * z \right) \tag{10}$$

The solution of Eq. (7) for λ is similar to a solution for displacement under a "dummy" load vector {x}.

Once, the sensitivity information $\frac{dg}{db}$ has been computed, any gradient based optimization softwares[4, 5] can be used to obtain a new, improved design.

The focus of this paper is in the parallel computation of $\frac{dz}{db}$ as shown in Eq. (5), and particularly, the computation of the term $\frac{dK}{db}$.

Since in the finite element procedure

$$[K] = \sum_{e=1}^{\#\text{ elements}} [K^{(e)}] \tag{11}$$

Therefore, computation of $\frac{d[K]}{db}$ involves with computation of $\frac{d[k^e]}{db}$ and the latter can be obtained either by

(i) Finite Difference Method

or

(ii) Analytical Method

In the finite difference method, a small perturbation of a design variable is first applied, then approximate derivative (which can be affected by round-off and truncation errors[3]) can be generated. The analytical method tends to generate very cumbersome expressions for the derivatives. Thus, the objectives of this paper is to use automatic differentiation (ADIFOR) tools[6] to compute the derivatives of $\frac{d[k^e]}{db}$ in a parallel-vector computer environment.

A brief review of ADIFOR tools[6] is given in Section 2. Parallel generation and assembly [7] of the stiffness matrix [K] is presented in Section 3. Parallel-Vector equation solver [8] which will be used to solve system of Eq. (5) is summarized in Section 4. Numerical examples are presented in Section 5, and conclusions are drawn in Section 6.

II. A Brief Review on Automatic Differentiation[6]

Automatic Differentiation (AD) is essentially an automatic implementation of the chain rule of differentiation based on tracking the connection between the dependent (or output) and independent (or input) variables.

Typically, to calculate the derivative of any output variable in a computer program with respect to any input variable, one modifies the original program by inserting of specialized instruction which identify the relevant output and input variables.

Automatic differentiation produces exact derivatives, limited only by machine precision. There are two modes of AD. In the forward mode, the chain rule is evaluated from the input to the output. In this mode, the computational cost increases with the number of input variables. In the reverse mode, the chain rule is evaluated from the output to the input.

In order to understand the forward mode in AD, let's refer to Figure 1 where the computation flow to evaluate

$$y_3 = \frac{-20\, b_2}{\left(2\, b_1\, b_2 + \sqrt{2}\, b_1^2\right)} = \frac{-20\, b_2}{b_1\left(2\, b_2 + \sqrt{2}\, b_1\right)}$$

is shown in a form of the directed graph.

The derivatives of $\frac{dy_3}{db_2}$ and $\frac{dy_3}{db_1}$ are also shown in a form of the directed graph in Figure 2.

In Figure 2, the connecting link between any 2 vertex represents the chain-rule derivatives. As an example, $\frac{\partial a}{\partial b_2} = 2\,b_1$ and $\frac{\partial d}{\partial a} = 1$.

On the other hand, if the reverse mode of differentiation is used to calculate $\frac{dy_3}{db_2}$, then the chain-rule of differentiation will start with the output variable y_3, and then proceed as following:

$$\frac{dy_3}{dd} = \frac{20\, b_2}{\left(2\, b_1\, b_2 + \sqrt{2}\; b_1^2\right)^2}$$

$$\frac{dy_3}{da} = \frac{dy_3}{dd}\,\frac{\partial d}{\partial a} = \frac{20\, b_2 * 1}{\left(2\, b_1\, b_2 + \sqrt{2}\; b_1^2\right)^2}$$

$$\frac{dy_3}{dx_2} = \frac{dy_3}{da}\,\frac{\partial a}{\partial b_2} + \frac{\partial y_3}{\partial b_2}$$

$$= \frac{20\, b_2 * 1 * 2\, b_1}{\left(2\, b_1\, b_2 + \sqrt{2}\; b_1^2\right)^2} + \frac{-20}{\left(2\, b_1\, b_2 + \sqrt{2}\; b_1^2\right)}$$

It has been concluded from earlier research works[6, 9, 10] that using automatic differentiation (AD) method, such as ADIFOR tool [6], will be more computationally efficient than the finite difference method. In most problems, however, analytical method is more efficient than ADIFOR tool (but at the expense of assuming there is no human errors in deriving analytical derivative expressions).

The comparisons of computational costs and the accuracy to evaluate derivative information between the Finite Difference, Analytical and ADIFOR have been discussed[6, 9, 10]. This paper, therefore, will focus on the issue of incorporating derivative calculation subroutines (generated by ADIFOR) in a parallel-vector high-performance computer environment.

III. Parallel Generation and Assembly on Distributed- and Shared Memory Computers[7]

The choice of the storage scheme for the global stiffness matrix in any finite element analysis code is based on whether it will save the memory or it will enhance the vector speed, or both. The row-oriented storage scheme[8] is good for saxpy operation and shared memory type computers, while the skyline storage is good for dot product (daxpy) operation. Moreover, the skyline storage scheme requires less memory and this feature is important for computers with distributed-memory (since each processor usually has less memory capacity as compared to shared-memory computers). Fortunately, the Intel iPSC/860 computers have good vector performance for daxpy operation. In order to use the vector-unrolling technique to improve the vector performance, a block-skyline columns storage and block rows storage schemes for the stiffness matrix is used on the Intel and Cray type computers, respectively (as shown in Figure 3). To simplify the discussion, assuming the global matrix is full and three processors are used to store different portions of the global stiffness matrix.

The size of the block is called k if there are k columns (or k-rows) in each block. It is realized that the choice of k will have the effects on

1. the in-core memory requirement,
2. the vector performance,
3. the communication performance.

For the Intel iPSC/860 parallel computers, the block size in MPFEA is set to be 8. Since each processor only has certain *block-columns* (or block rows) of the global stiffness matrix, the generation and assembly of this matrix can be done in parallel without any communications among processors. The work involved in the generation and assembly procedure can be summarized as (for each processor i, where i = 1, 2, ... , NP):

Task 1. To identify (but not to search for!) the elements that contribute to the columns (or rows) which belong to processor i.
Task 2. To generate these elements stiffness matrices.
Task 3. To assemble the global stiffness matrix with these element stiffness matrices.

It should be noted here that even for the case of nonlinear structural analysis, Task 1 of the above procedure needs to be done only once, while Task 2 and Task 3 have to be performed repeatedly since the global matrix will be updated in each nonlinear iteration.

IV. Parallel-Vector Choleski Method Development[8]

In the sequential Choleski method, a symmetric, positive-definite stiffness matrix, [K], can be decomposed as

$$[K] = [U]^T [U] \tag{12}$$

with the coefficients of the upper-triangular matrix, [U]:

$$u_{ij} = 0 \quad for \quad i > j \tag{13}$$

$$u_{11} = \sqrt{K_{11}}\ ; \ u_{1j} = \frac{K_{1j}}{u_{11}} \quad for \quad j \geq 1 \tag{14}$$

$$u_{ii} = \sqrt{K_{ii} - \sum_{k=1}^{i-1} u_{ki}^2} \quad for \quad i > 1 \tag{15}$$

$$u_{ij} = \frac{K_{ij} - \sum_{k=1}^{i-1} u_{ki}\, u_{kj}}{u_{ii}} \quad for \quad i, j > 1 \tag{16}$$

For example, u_{57} can be computed from Eq. (18) as:

$$u_{57} = \frac{k_{57} - u_{15}\, u_{17} - u_{25}\, u_{27} - u_{35}\, u_{37} - u_{45}\, u_{47}}{u_{55}} \tag{17}$$

The calculations in Eq. (17) for the term u_{57} (of row 5) only involve columns 5 and 7. Furthermore, the "final value" of u_{57} cannot be computed until the final, updated values of the first four rows have been completed. Assuming that only the first two rows of the factored matrix, [U], have been completed, one still can compute the second partially-updated value of u_{57} as designated by superscript (2):

$$u_{57}^{(2)} = k_{57} - u_{15}\, u_{17} - u_{25}\, u_{27} \tag{18}$$

If row 3 has also been completely updated, then the third partially-updated value of u_{57} can be calculated as:

$$u_{57}^{(3)} = u_{57}^{(2)} - u_{35}\, u_{37} \tag{19}$$

This observation suggests an efficient way to perform Choleski factorization in parallel on NP processors. For example, each row of the coefficient stiffness matrix, [K], is assigned to a separate processor.

From Eq. (17), assuming NP = 4, it is seen that row 5 cannot be completely updated until row 4 has been completely updated. In general, in order to update the ith row, the previous (i-1)rows must already have been updated. For the above reasons, any NP consecutive rows of the coefficient stiffness matrix, [K], will be processed by NP separate processors. As a consequence, while row 5 is being processed by a particular processor, say processor 1, then the first (5-NP) rows have already been completely updated. Thus, if the ith row is being processed by the pth processor, there is no need to check every row (from row 1 to row i-1) to make sure they have been completed. It is safe to assume that the first (i-NP) rows have already been completed as shown in the triangular cross-hatched region of Figure 4.

Synchronization checks are required only for the rows between (i-NP + 1) and (i-1) as shown in the rectangular solid region of Figure 4. Since the first (i-NP) rows have already been completely factored, the ith row can be "partially" processed by the pth processor as shown in Eq. (18, 19).

V. Numerical Applications

Different finite element types (such as 2-D Truss, and Plate/Shell elements) and different type of design variables (such as cross-sectional areas, joint coordinates of truss elements and thickness of plate elements) are considered in this section. The first two examples are small-size for the purpose of verifying the accuracy of derivatives (d $[k^{(e)}]$ / d b) generated by ADIFOR[6] as compared to the ones obtained by finite difference technique. The last example is medium-size for the purpose of evaluating the parallel-vector performance of the entire finite element and Design Sensitivity Analysis (DSA) process.

Example 1: Plate-Structure With (Non-Shape) Thickness Design Variable

In this example, 32 plate elements[11] are used, a point force is applied at the center of the fixed plate (see Figure 5). Thickness of a plate is selected as (non-shape) design variable in this case. The original thickness is 0.03 and a perturbation of 0.5% is used in the finite (central) difference scheme.

The derivatives of element stiffness matrix (in global reference and using ADIFOR) with respect to the thickness t for typical members such as members 5, 12, and 19 ar presented in Table 1. These derivatives are in good agreement with the ones obtained by finite (central) difference scheme.

Example 2: Truss-Structure With (Shape) Joint Coordinate Design Variables

In this example, a 1 bay x 1 story truss structure is shown in Figure 6. This small-scale structure has 4 joints and 5 members. All joint x-coordinates of this structure are selected as (shape) design variables. A horizontal force F is applied at node 1. The dimensions for each base and height of this structure are 12" and 9", respectively. Young modulus and cross-sectional area are 29000 Ksi and 4 in^2, respectively. A perturbation of 1% is used in the finite (central) difference scheme. The derivatives of element stiffness matrix (in global reference and using ADIFOR) with respect to a typical x-coordinate of joint 2 for members 1 and 5 are presented in Table 2. Again, these derivates are in good agreements with the ones obtained by finite (central) difference scheme.

Example 3: A 2-D Truss Structure With 80 Bays and 190 Stories

In this example, a 80 bay x 190 story truss structure is also shown in Figure 6. A horizontal force F is applied at node 100. All other datas are the same as in Example 2. There are 96 cross-sectional areas selected as (non-shape) design variables in this example. This structure has 60,990 elements. The resulted structural stiffness matrix has 30,780 degree-of-freedom. Using the variable bandwidth storage scheme[8] will require a real 1-dimensional array with 5,171,574 words to store the stiffness matrix in the core memory. The average bandwidth for this stiffness matrix is 168.

The performance of the entire finite element analysis and design sensitivity analysis (using ADIFOR tool) on 1, 8, and 16 Cray-C90 processors are shown in Table 3. The total speed-up for the ENTIRE PROCESS are 7.32 and 12.93 when 8 and 16 Cray-C90 processors are used, respectively.

VI. Conclusions

Based upon the numerical results presented in this paper, the following conclusions can be made:

1. Automatic Differentiation (ADIFOR)[6] tool has been successfully applied to both simple (TRUSS) and complex PLATE/SHELL[11] finite elements.

2. Both *non-shape* and *shape* design variables can be successfully treated.

3. For the first time (to the authors' knowledge), ADIFOR tool can be applied in a parallel-vector computer environment for *non-shape* and *shape* sensitivity analysis.

4. The *entire* finite element and sensitivity analysis can be done with excellent parallel and vector speed (using all 16 Cray-C90 processors).

VII. Acknowledgments

The financial support from NASA grant NAG1-858 are acknowledged. The authors are also deeply indebted to Drs. L. Green, P. Newman, J. Barthelemy (all from NASA Langley Research Center), C. Bischof (from Argonne National Laboratory) and A. Carle (from Rice University) for helpful discussions during the ADIFOR user workshop (September 13-14, 1993), held at Building 1192C-E, the CFD Laboratory, NASA LaRC). Helpful discussions with Dr. A. Tessler on using his plate/shell element (NASA Langley Research Center) is also appreciated.

VIII. References

1. T.J.R. Hughes, *The Finite Element-Method*, Prentice-Hall, Inc., (1987).

2. J.S. Arora and E.J. Haug, *Applied Optimal Design*, John Wiley & Sons, Inc., (1979).

3. R.T. Haftka, Z. Gürdal, and M.P. Kamat, *Elements of Structural Optimization*, Kluwer

Academic Publishers (1990).

4. R. Thareja and R.T. Haftka, "A Modified Version of NEWSUMT For Inequality and Equality Constraints," VPI Report 148, (March 1985).

5. G.N. Vanderplaats, "CONMIN: A Fortran Program for Constrained Function Minimization", NASA-TM X-62282, (1973).

6. C.H. Bischof and A. Griewank, "ADIFOR: A Fortran System For Portable Automatic Differentiation", Proceedings the 4th AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization, Cleveland, OH, pp. 433-441, AIAA 92-4744-CP, (September 1992).

7. J. Qin and D. T. Nguyen, "A New Parallel-Vector Finite Element Analysis Software on Distributed Memory Computers," Proceedings of the AIAA/ASME/ASCE/AHS 34th SDM Conference, La Jolla, CA (April 19-22, 1993).

8. T.K. Agarwal, O.O. Storaasli, and D.T. Nguyen, "A Parallel-Vector Algorithm for Rapid Structural Analysis on High-Performance Computers," Proceedings of the AIAA/ASME/ASCE/AHS 31th SDM Conference, Long Beach, CA (April 2-4, 1990).

9. J.F. Barthelemy and L.E. Hall, "Automatic Differentiation As A Tool In Engineering Design," NASA-TM 107661, (August, 1992).

10. C. Bischof, G. Corliss, L. Green, A. Griewank, K. Haigler and P. Newman, "Automatic Differentiation of Advanced CFD Codes for Multidisciplinary Design," Computing Systems in Engineering, Vol. 3, No. 6, pp. 625-637, (1992).

11. A. Tessler, "A C° Anisoparametric Three-Node Shallow Shell Element for General Shell Analysis," MTL-TR-89-72, (August 1989).

Figure 1: Computational Graph for $y_3 = \frac{-20\,b_2}{\left(2\,b_1\,b_2 + \sqrt{2}\;b_1^2\right)} = \frac{-20\,b_2}{b_1\left(2\,b_2 + \sqrt{2}\;b_1\right)}$

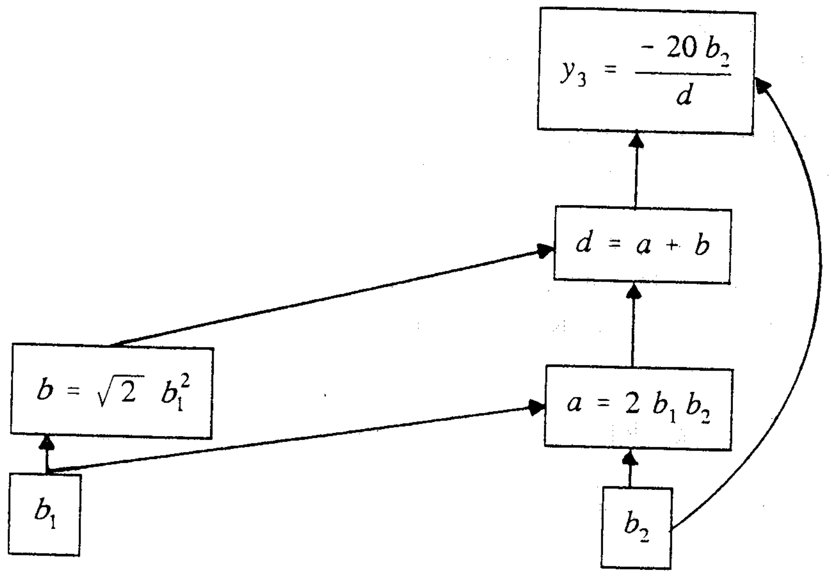


Figure 2: Computational Graph for

$$\frac{d\,y_3}{d\,b_2} = \frac{40\,b_2\,b_1}{\left(2\,b_1\,b_2 + \sqrt{2}\;b_1^2\right)^2} + \frac{-20}{\left(2\,b_1\,b_2 + \sqrt{2}\;b_1^2\right)}$$

$$\frac{d\,y_3}{d\,b_2} = \left(\frac{20\,b_2}{d^2}\right)(1)\,(2\,b_1) + \left(-\frac{20}{d}\right)$$

$$\frac{d\,y_3}{d\,b_1} = \frac{(20 * b_2)\,(2\,b_2 + 2\sqrt{2}\;b_1)}{[\,b_1\,(2\,b_2 + \sqrt{2}\;b_1)\,]^2}$$

$$\frac{d\,y_3}{d\,b_1} = \left(\frac{20\,b_2}{d^2}\right)(1)\,(2\,b_2) + \left(\frac{20\,b_2}{d^2}\right)(1)\,(2\,\sqrt{2}\;b_1)$$

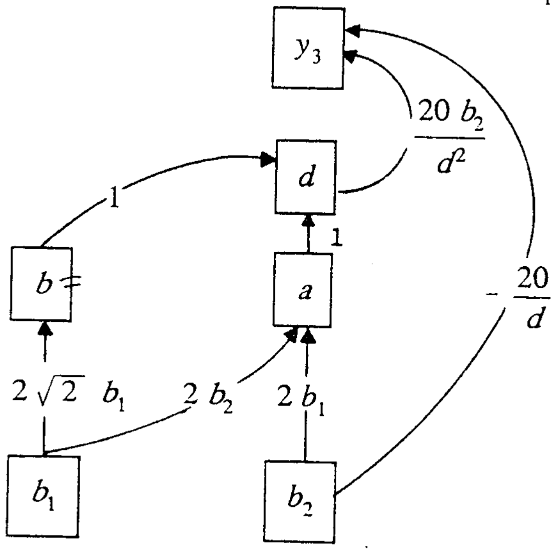


Figure 3. Block-skyline columns storage and block rows storage schemes

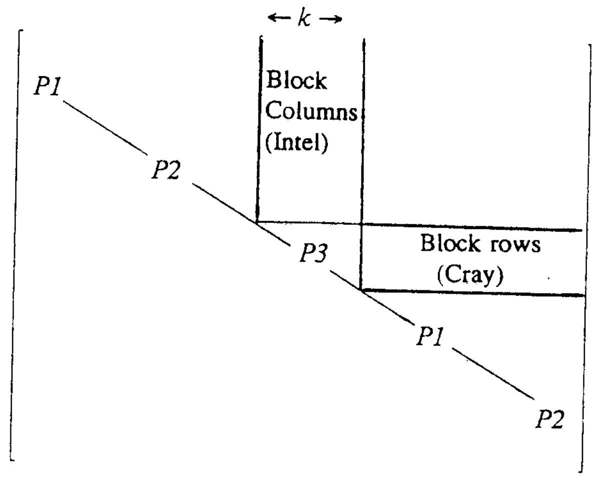


Figure 4: Information required to update row i

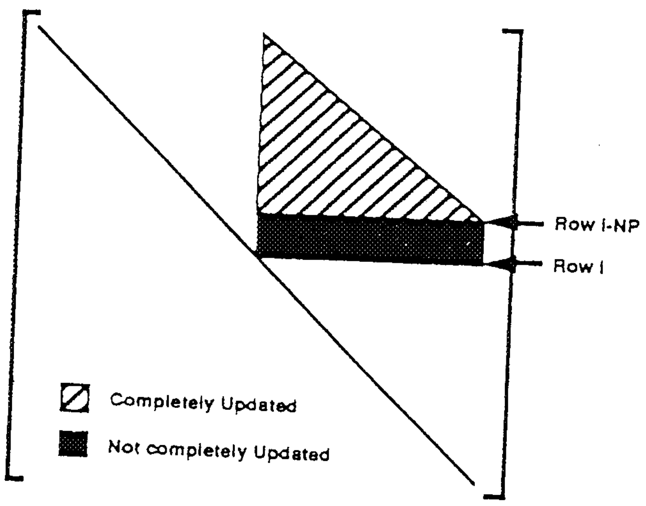


Figure 5: Clamped Plate - Structure

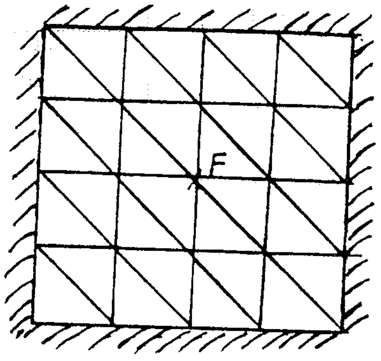
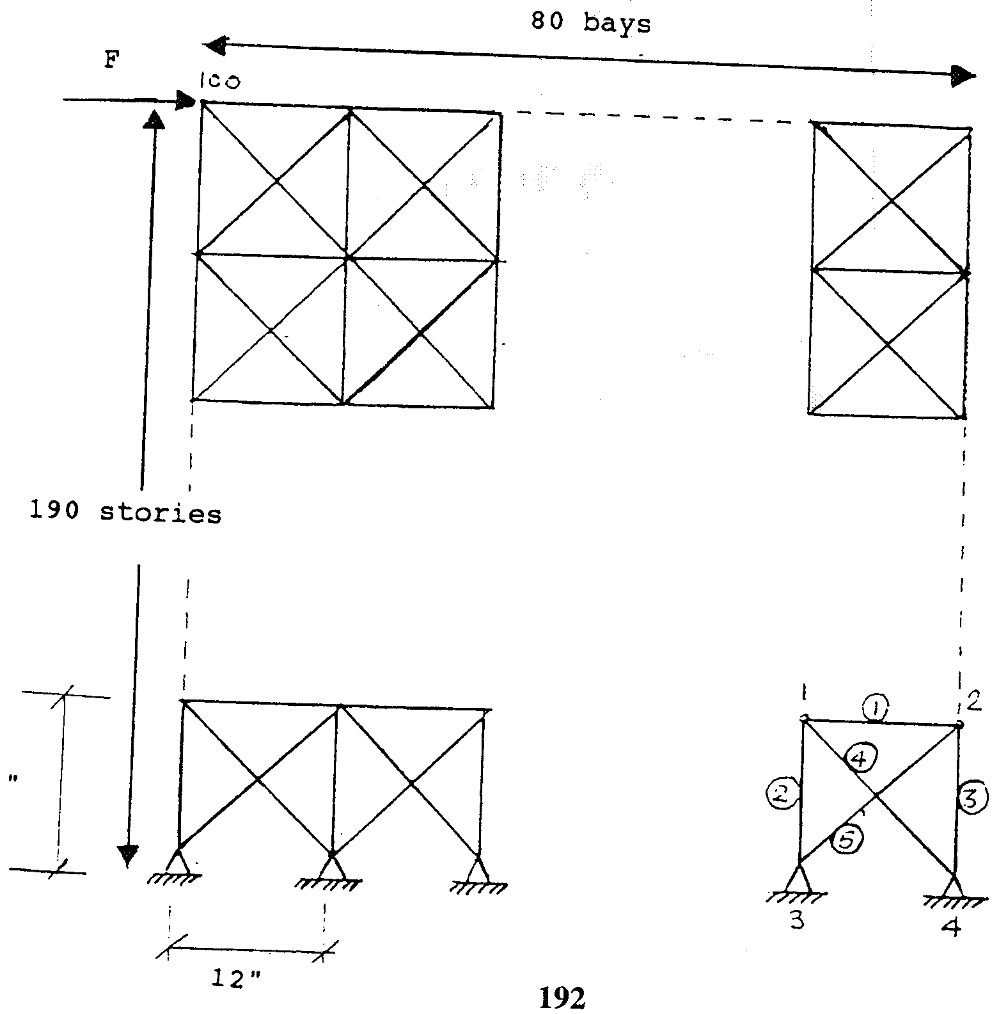


Figure 6: 2-D Truss Structure



$$\frac{\partial\,[k^{(5)}]}{\partial\,t} = [\,5576.925\,,\ -4780.2198\,,\ 0\,,\ 0\,,\ 0\,,\ -15934.066\,,\ \ldots\ldots\,]$$

$$\frac{\partial\,[k^{(12)}]}{\partial\,t} = [\,15934.068\,,\ 5576.923\,,\ 0\,,\ 0\,,\ 0\,,\ -5576.923\,,\ \ldots\ldots\,]$$

$$\frac{\partial\,[k^{(19)}]}{\partial\,t} = [\,21510.99\,,\ 5576.925\,,\ 0\,,\ 0\,,\ 0\,,\ 8.268E\text{-}12\,,\ \ldots\ldots\,]$$

Table 1: ADIFOR Derivatives of Plate Element Stiffness Matrix with Respect to Thickness (Non-shape) Design Variable

Table 2: ADIFOR Derivatives of Truss Element Stiffness Matrix with Respect to x-coordinate of Joint 2 (Shape) Design Variable.

```
el. stiff [k] for member 1
 0.966667E+04  0.000000E+00 -0.966667E+04  0.000000E+00
 0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00
-0.966667E+04  0.000000E+00  0.966667E+04  0.000000E+00
 0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00
```

Gradient of stiff [k] w.r.t DV 2 = $\frac{\partial [k^{(1)}]}{\partial x_2}$

```
   -805.55555555556       0.    805.55555555556   0.
0.      0.   0.    0.
    805.55555555556       0.   -805.55555555556   0.
0.      0.   0.    0.
```

```
el stiff [k] for member 5

 0.494933E+04  0.371200E+04 -0.494933E+04  -0.371200E+04
 0.371200E+04  0.278400E+04 -0.371200E+04  -0.278400E+04
-0.494933E+04 -0.371200E+04  0.494933E+04   0.371200E+04
-0.371200E+04 -0.278400E+04  0.371200E+04   0.278400E+04
```

Gradient of stiff [k] w.r.t DV 2 = $\frac{\partial [k^{(5)}]}{\partial x_2}$

```
   32.995555555555  -284.58666666667   -32.995555555555   284.58666666667
 -284.58666666667   -445.44000000000   284.58666666667    445.44000000000
  -32.995555555555   284.58666666667    32.995555555555  -284.58666666667
  284.58666666667    445.44000000000  -284.58666666667   -445.44000000000
```

Table 3: Parallel-Vector Performance For DSA of 80 Bays x 190 Stories Truss Structure Using ADIFOR Tool on Multiple Cray-C90 Processors

Tasks	Number of Cray-C90 Processors			Speed-Up Factors	
	1 proc.	8 proc.	16 proc.	8 proc.	16 proc.
(A)	0.4855^{sec}	0.09954^{sec}	0.07854^{sec}	4.88	6.18
(B)	0.9582^{sec} (0.9906*)	0.1320^{sec} (0.1433*)	0.0731^{sec} (0.1547*)	7.26	13.11
(C)	2.6290^{sec}	0.3568^{sec}	0.2026^{sec}	7.37	12.98
(D)	0.1019^{sec}	0.1015^{sec}	0.1018^{sec}	N/A	N/A
(E)	2.3717^{sec}	0.3034^{sec}	0.1558^{sec}	7.82	15.22
(F)	9.6934^{sec}	1.2128^{sec}	0.6047^{sec}	7.99	16.03
Entire Process	16.2740^{sec}	2.2221^{sec}	1.2591^{sec}	7.32	12.93

Notes:
(A) To generate column heights of stiffness matrix
(B) To generate and assemble stiffness matrix
(C) To factorize stiffness matrix
(D) To get static (forward/backward) solution (sequential computation)
(E) To generate the right-hand-side vectors for sensitivity equations
(F) To solve for displacement sensitivity vectors
* Wall-Clock-Time

Automatic Differentiation for Design Sensitivity Analysis of Structural Systems Using Multiple Processors

by

Duc T. Nguyen[†], Olaf Storaasli[§], Jiangning Qin[†], and Ramzi Qamar[†]

[†]Multidisciplinary Parallel-Vector Computation Center,
135 KDH Building,
Old Dominion University,
Norfolk, VA 23529-0241

[§]Computational Structures Branch,
NASA Langley Research Center,
Hampton, VA 23681

OBJECTIVES

1. To obtain <u>accurate</u> derivatives of complex finite elements and/or complex design variables

2. Design variables can be either <u>non-shape</u> (such as areas, thickness) or <u>shape</u> types (such as joint coordinates)

3. The <u>entire</u> solution <u>process</u> should be <u>parallelized</u> and <u>vectorized</u> to reduce solution time

4. Numerical validation and performance evaluation for the proposed procedure.

MOTIVATION

1. Analytical (hand-coded) derivatives are <u>not feasible</u> for complex finite elements and <u>shape variables</u>

2. Finite difference derivatives are <u>expensive</u> and can be <u>inaccurate</u>

APPROACH USED

1. Utilizing Automatic Differentiation (ADIFOR) tool

   Developed by ANL & CRPC at Rice

   Applications Feed back from AAOB & CSB at LaRC

2. Parallelizing and Vectorizing every step of the <u>entire</u> solution <u>process</u>

UNIQUE FEATURES OF THIS WORK

1. Both simple and complex finite elements (2-D truss, and 3-D plate/shell) are treated.

2. Both (non-shape) design variables (such as areas of truss members, or thickness of plate and shell members), and shape design variables (such as joint coordinates) are considered

3. The entire solution process has been parallized and vectorized

4. EXCELLENT speed-up has been achieved even on "small-scale" example.

GENERAL FORMULATION FOR DESIGN SENSITIVITY ANALYSIS (D.S.A.)

- Equilibrium Equations

$$[K]\{Z\} = \{F\} \tag{1}$$

Take derivatives of both sides of Equation (1) with respect to design variable vector b

$$\frac{\partial[K]}{\partial b} * \{Z\} \div [K] * \frac{\partial\{Z\}}{\partial b} = [0] \tag{2}$$

- Sensitivity Equations

using "NEW" parallel Gen. + Assem. technique!

$$[K]\frac{\partial\{Z\}}{\partial b} = \underbrace{-\frac{\partial[K]}{\partial b} * \overset{\text{Vector}}{\{Z\}}}_{\text{Matrix}} \tag{3}$$

Note:

Parallel-Vector Eq. Solver is needed for CSM & CFD !

D.S.A.
FOR 2-D TRUSS ELEMENT

$$[k^{(e)}]_{Global} = \frac{EA}{L}\begin{bmatrix} C_x^2 & C_xC_y & -C_x^2 & -C_xC_y \\ & C_y^2 & C_xC_y & -C_y^2 \\ & & C_x^2 & C_xC_y \\ & & & C_y^2 \end{bmatrix}_{4x4} \quad where \quad \begin{cases} L = \sqrt{(x_k - x_j)^2 + (y_k - y_j)^2} \\ C_x = \dfrac{x_k - x_j}{L} \\ C_y = \dfrac{y_k - y_j}{L} \end{cases}$$

$$= \frac{EA}{L} * [T]_{4\times 4}$$

(A) Non-Shape Design Variables

$$\frac{\partial [k^{(e)}]}{\partial \vec{b}} = \frac{\partial \left[k^{(e)} \right]_{Global}}{\partial \vec{A}} = \frac{E}{L} * \left[T \right] = EASY\ !$$

(B) Shape Design Variables

$$\frac{\partial \left[k^{(l)} \right]_{Global}}{\partial \begin{Bmatrix} x_j \\ x_k \\ y_j \\ y_k \end{Bmatrix}} = VERY\ TEDIOUS\ !$$

D.S.A. FOR TRIANGULAR PLATE BENDING ELEMENT

(Reference: Preziemienicki's Book)

$$\left[k^{(e)}\right]_{Local} = \int_{\upsilon} D^{T} \kappa D \, dV = \frac{Et^{3}}{12\left(1 - \nu^{2}\right)}$$

$$* \int_{x_1}^{x_2}\int_{y_1}^{y_2} \left[\begin{array}{c|cccccc} & & & 0 & & & \\ \hline & 4 & & & & & \\ & 0 & 2(1-\nu) & & \text{Symmetric} & & \\ & 4\nu & 0 & 4 & & & \\ & 12x & 0 & 12\nu x & 36x^2 & & \\ & 4(\nu x + y) & 4(1-\nu)(x+y) & 4(x+\nu y) & 12x(\nu x + y) & \begin{array}{c}(12-8\nu)(x+y)^2 \\ -8(1-\nu)xy\end{array} & \\ & 12\nu y & 0 & 12y & 36\nu xy & 12(x+\nu y)y & 36y^2 \end{array}\right] dx\,dy \qquad (5.249)$$

Then

$$\left[k^{(e)}\right]_{Global} = \left[k^{(e)}\right]_{Local} * \left[T\right]$$

Functions of element nodal coordinates also!

(A) <u>Non-Shape</u> <u>Design Variables</u>

$$\frac{\partial \left[k^{(e)}\right]_{Global}}{\partial t} = E\ A\ S\ Y\ !$$

(B) <u>Shape</u> <u>Design Variables</u>

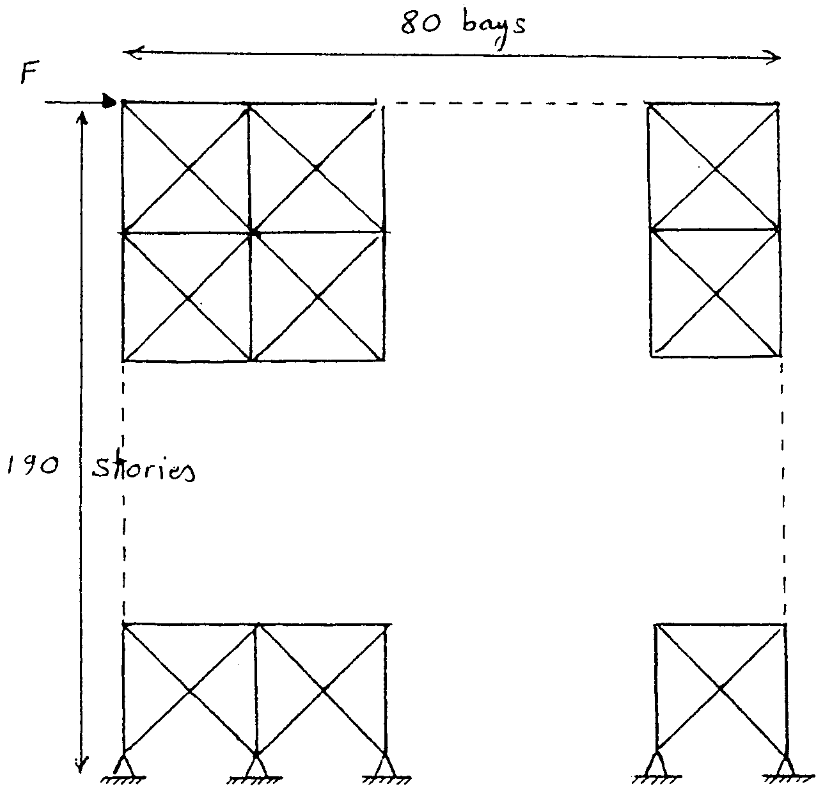
$$\frac{\partial \left[k^{(e)}\right]_{Global}}{\partial \begin{Bmatrix} x_i \\ x_j \\ x_k \\ . \\ . \\ . \\ . \\ z_i \\ z_j \\ z_k \end{Bmatrix}} = V\ E\ R\ Y\ \ T\ E\ D\ I\ O\ U\ S\ !$$

D.S.A. FOR TRIANGULAR PLATE/SHELL BENDING ELEMENT

(Reference: Alex Tessler's Element)

<u>Note:</u> In this work, Alex Tessler's Plate/Shell finite element is used.

APPLICATIONS



Example 1 (80 bays, 190 stories

NEL = 60,990 elements (TRUS

NEQ = 30,780 equations

NTERMS = 5,171,574 terms

AVEBW = 168 average bandwidth

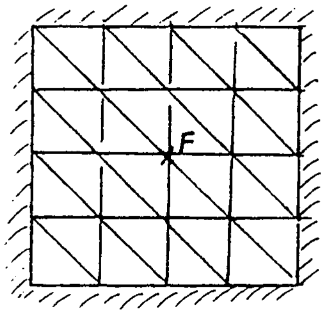
NDV = 96 Non-shape design variables

Example 2 (1 bay, 1 story)

NEL = 5 elements (TRUSS)

NEQ = 4

NDV = 8 Shape design variab



Example 3 (Plate structure)

NEL = 32 elements (Alex Tessler's Plate/Shell)

NDV = 75 shape design variables

NUMERICAL RESULTS AND PARALLEL PERFORMANCE

Tasks	Number of Cray - C90 Processors			Speed-Up Factors	
	1 proc.	8 proc.	16 proc.	8 proc.	16 proc.
(A)	0.4855^{sec}	0.09954^{sec}	0.07854^{sec}	4.88	6.18
(B)	0.9582^{sec} (0.9906*)	0.1320^{sec} (0.1433*)	0.0731^{sec} (0.1547*)	7.26	13.11
(C)	2.6290^{sec}	0.3568^{sec}	0.2026^{sec}	7.37	12.98
(D)	0.1019^{sec}	0.1015^{sec}	0.1018^{sec}	N/A	N/A
(E)	2.3717^{sec}	0.3034^{sec}	0.1558^{sec}	7.82	15.22
(F)	9.6934^{sec}	1.2128^{sec}	0.6047^{sec}	7.99	16.03
Entire Process	16.2740^{sec}	2.2221^{sec}	1.2591^{sec}	7.32	12.93

```
nel,ndofpe,nodes,ndofpn,nunrol,nummat,ireal,nbays,      nstory,ndv
7,  4,   2*2,  8,  1000,  0,  10,  300,  1000
+design variable, total memory needed= 1000,  7881649
 max. wall clock timef for gen+assem = 0.250323822
 (z)/d(b) with respect to DV # 1000
 d(z)/d(b) =  0.463915E-02 0.197448E-03 0.463915E-02 0.127947E-03 0.463915E-02
 d(z)/d(b) =  0.584470E-04 0.463915E-02-0.110535E-04 0.463915E-02-0.805540E-04
 d(z)/d(b) =  0.463915E-02-0.150054E-03 0.463915E-02-0.219555E-03 0.463915E-02
 d(z)/d(b) = -0.289055E-03 0.463915E-02-0.358556E-03 0.463915E-02-0.428056E-03
ME,  time for generate SD=1,  2.465475E-2
ME,  time for generate K =1,  0.250328442
ME,  time for Factori.   =1,  0.234019218
ME,  time for Solution   =1,  2.2428906000002E-2
ME,  time for (dK/db)*X  =1,  6.526009938
ME,  time for dX/db      =1,  21.726178002
**  Time in boundc  =4.737786E-3
**  Time in jointc  =3.0296399999999E-4
**  Time in apload  =4.8300000000001E-5
**  Time in elconn  =3.33129E-3
**  Time in materp  =4.7050644E-2
**  Time in colht   =0.152899086
TOTAL TIME:(nel,neq,ielm,nterms)28.991996244,  12300,  6600,  12300,  186532
-------------------------------------------------------------------------------
```

```
nel,ndofpe,nodes,ndofpn,nunrol,nummat,ireal,nbays,      nstory,ndv
 ,  4,   2*2,  8,  1000,  0,  10,  300,  1000
 design variable, total memory needed= 1000,  7881649
 max. wall clock timef for gen+assem = 0.143736066
d(z)/d(b) with respect to DV # 1000
 d(z)/d(b) =  0.463915E-02 0.197448E-03 0.463915E-02 0.127947E-03 0.463915E-02
 d(z)/d(b) =  0.584470E-04 0.463915E-02-0.110535E-04 0.463915E-02-0.805540E-04
 d(z)/d(b) =  0.463915E-02-0.150054E-03 0.463915E-02-0.219555E-03 0.463915E-02
 d(z)/d(b) = -0.289055E-03 0.463915E-02-0.358556E-03 0.463915E-02-0.428056E-03
ME,  time for generate SD=1,  2.466528E-2
ME,  time for generate K =1,  0.125706372
ME,  time for Factori.   =1,  0.141787986
ME,  time for Solution   =1,  2.2426734E-2
ME,  time for (dK/db)*X  =1,  3.275844468
ME,  time for dX/db      =1,  10.83518028
ME,  time for generate SD=2,  1.8168000000429E-5
ME,  time for generate K =2,  0.12609603
ME,  time for Factori.   =2,  0.141645408
ME,  time for Solution   =2,  5.1287999999872E-5
ME,  time for (dK/db)*X  =2,  3.28376691
ME,  time for dX/db      =2,  10.836083088
**  Time in boundc  =4.723914E-3
**  Time in jointc  =3.0290400000002E-4
**  Time in apload  =4.8348000000004E-5
**  Time in elconn  =3.331998E-3
**  Time in materp  =4.7076264E-2
*  Time in colht   =3.3666912000001E-2
OTAL TIME:(nel,neq,ielm,nterms)14.514775074,  12300,  6600,  6150,  186532
.OTAL TIME:(nel,neq,ielm,nterms)14.528120334,  12300,  6600,  6181,  186532
----------------------------
```

```
 el,ndofpe,nodes,ndofpn,nunrol,nummat,ireal,nbays,     nstory,ndv
7,  4,  2*2,  8,  1000,  0,  10,  300,  1000
#design variable, total memory needed= 1000,  7881649
 max. wall clock timef for gen+assem = 0.106087758
d(z)/d(b) with respect to DV # 1000
 d(z)/d(b) =  0.463915E-02 0.197448E-03 0.463915E-02 0.127947E-03 0.463915E-02
 d(z)/d(b) =  0.584470E-04 0.463915E-02-0.110535E-04 0.463915E-02-0.805540E-04
 d(z)/d(b) =  0.463915E-02-0.150054E-03 0.463915E-02-0.219555E-03 0.463915E-02
 d(z)/d(b) = -0.289055E-03 0.463915E-02-0.358556E-03 0.463915E-02-0.428056E-03
ME,  time for generate SD=3,  2.441805E-2
ME,  time for generate K =3,  8.4435756E-2
ME,  time for Factori.   =3,  9.3954744000001E-2
ME,  time for generate SD=1,  2.2068600000003E-4
ME,  time for generate SD=2,  2.5296000000008E-5
ME,  time for generate K =1,  8.4171846000001E-2
ME,  time for generate K =2,  8.452527E-2
ME,  time for Factori.   =1,  9.3851616E-2
ME,  time for Factori.   =2,  9.3939270000001E-2
ME,  time for Solution   =1,  6.4373999999923E-5
ME,  time for Solution   =2,  5.441399999917E-5
ME,  time for (dK/db)*X  =1,  2.156280738
ME,  time for (dK/db)*X  =2,  2.168834262
ME,  time for dX/db      =1,  7.22125122
 E,  time for dX/db      =2,  7.196945016
 E,  time for Solution   =3,  2.2304856E-2
 E,  time for (dK/db)*X  =3,  2.163674568
ME,  time for dX/db      =3,  7.211424354
**  Time in boundc  =4.723506E-3
**  Time in jointc  =3.0207599999998E-4
**  Time in apload  =4.8342000000007E-5
**  Time in elconn  =3.328746E-3
**  Time in materp  =4.6941834E-2
**  Time in colht   =1.0283598E-2
TOTAL TIME:(nel,neq,ielm,nterms)9.621492162,  12300,  6600,  4100,  186532
TOTAL TIME:(nel,neq,ielm,nterms)9.607627608,  12300,  6600,  4131,  186532
TOTAL TIME:(nel,neq,ielm,nterms)9.721047816,  12300,  6600,  4131,  186532
```

CONCLUSIONS

1. Automatic Differentiation (ADIFOR) tool has been successfully applied to both simple (TRUSS) and complex (Alex Tessler's PLATE/SHELL) finite elements

2. Both non-shape and shape design variables can be successfully treated.

3. For the first time (to the author's knowledge), ADIFOR tool can be applied in a parallel-vector computer environment for non-shape and shape sensitivity analysis.

4. The entire finite element + sensitivity analysis can be done with excellent parallel and vector speed (using all 16 Cray-C90 processors)

SESSION 6 Mosaic and the World Wide Web

Chaired by

Clyde R. Gumbert and John W. McManus