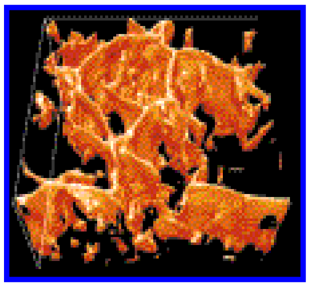
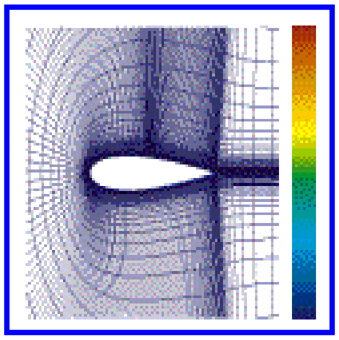
January-February 1997
Vol2. No. 22



New Tool Provides Accurate Derivatives for Multidisciplinary Applications

by Alan Carle



As part of the consortium headed by IBM Corp. under the High Performance Computing and Communications (HPCC) Program's Testbed-1 Cooperative Research Announcement (CRA), and in conjunction with the Multidisciplinary Design Optimization Branch at NASA Langley Research Center, Rice University has developed ADIFOR-MP, a new version of the ADIFOR Automatic Differentiation software for Fortran. ADIFOR is used to augment sophisticated computer simulation codes to compute derivatives of their outputs with respect to their inputs for use in advanced numerical optimization procedures.

Multidisciplinary design optimization (MDO) promises to revolutionize the design of complex vehicles, such as the High Speed Civil Transport, by applying numerical optimization to computer simulations of multiple interacting physical phenomena, including fluid flow, thermodynamics, and structures. Biannual AIAA symposiums on Multidisciplinary Analysis Optimization attest to the importance of MDO to the aerospace community. Derivatives, also known as "sensitivities," play a critical role in MDO. They allow researchers to understand how small changes to the inputs of one component of a complex system affect that component's output, or how small changes to the inputs of the whole system affect the system's outputs.

To obtain these derivatives, researchers have traditionally had two choices: use finite-difference approximations, or develop a program to compute the analytic derivatives. Finite differences are typically used in order to avoid the tedious, time-consuming, and error-prone programming required to construct code that computes the derivatives analytically. Unfortunately, it is usually very difficult to judge the accuracy of finite-difference approximations.

Fortunately, engineers can avoid these nuisances through the use of Automatic Differentiation (AD). An AD tool automatically applies the derivative chain rule to all of the expressions in a program to generate another program that computes -- in addition to its usual outputs -- the analytic derivatives of those outputs with respect to a set of input parameters. The ADIFOR 2.0 System, developed in a collaboration between Argonne National Laboratory and Rice University, implements AD for Fortran 77.

Automatic Differentiation & Parallelism

The computational expense of calculating derivatives of complex processes easily dominates the cost of programs that use derivatives. To make the use of such programs feasible, it is important to take advantage of all available parallelism. (The same conclusion was stated in "The Virtual Skeleton: Modeling Human Movements," NAS News, September -- October 1996, where derivatives are required for the solution of an optimal control problem.)

Researchers at Rice have investigated two techniques for applying parallelism to the computation of derivatives: (1) compute derivative "strips" in parallel, and (2) compute derivatives of explicitly parallel codes. These techniques are described in the next section.

Derivative Stripmining

AD, as implemented by ADIFOR, computes a gradient for each intermediate value computed by the original program. Each of these gradients is a vector whose elements are the derivatives of the intermediate value with respect to the input parameters of the program. If there are a sufficiently large number of inputs to the program, then it may be useful to divide up the calculation of these derivatives and assign them to multiple processors. For example: for six inputs, one processor can be used to compute all six derivatives; two processors can be simultaneously used to compute three derivatives each; or six processors can be used to compute one derivative each. In this manner, parallelism can be used to compute derivatives of sequential codes. Note, however, that this method redundantly recomputes the function value on each processor doing derivative work, so this technique fails to scale perfectly.

Differentiation of Explicitly Parallel Codes

The team of researchers at Rice have recently developed a prototype implementation of ADIFOR-MP, a "message-passing aware" version of ADIFOR. ADIFOR-MP is capable of augmenting explicitly parallel programs written using a subset of MPI or PVM message-passing routines. This new functionality makes it possible to take an MPI- or PVM-based simulation and incorporate it into an MDO framework. Stripmining can be used in conjunction with ADIFOR-MP to further increase the use of parallelism in the computation of derivatives.

To date, the ADIFOR-MP prototype has been applied to two explicitly parallel CFD solvers; the PVM implementation of OVERFLOW, developed at Ames, and the MPI implementation of TLNS3D, developed at NASA Langley Research Center. Preliminary results on OVERFLOW were presented by Rice's Mike Fagan at the 1996 Computational Aerosciences Workshop held in October at Ames.

`Even More' ADIFOR-MP Users Needed

The goal from the beginning of the ADIFOR project has been to construct a robust implementation of AD that can be used to construct derivative-enhanced versions of real codes. Since ADIFOR 2.0 became available in the summer of 1995 it has been requested by more than 200 users. To drive additional ADIFOR-MP development efforts, the ADIFOR team is interested in finding and collaborating with even more users who need accurate derivatives for explicitly parallel codes. Only through such collaborations will researchers achieve a good understanding of how MPI and PVM are used in practice and ensure that ADIFOR-MP is truly usable.

ADIFOR Availability

More information about the ADIFOR project:

- http://www.mcs.anl.gov/adifor
- http://www.cs.rice.edu/~adifor

To obtain the ADIFOR 2.0 System, email request to adifor@mcs.anl.gov or to adifor@cs.rice.edu.

Next Article Contents Main Menu NAS Home

Unstructured Meshes Play Role in Device Modeling

by Subhash Saini

This is the second part of "Semiconductor Device Modeling for 21st Century at ARC," an article that appeared in the November-December issue. The first installment addressed NASA's future unique computing needs and described a new program at Ames Research Center to simulate semiconductor devices (in partnership with Stanford University and UC Berkeley) using the HPCC-funded highly parallel computer systems at the NAS Facility.

Essential to both process modeling and device modeling is the capability of grid generation and parallel high-performance computing. Grid generation poses a daunting challenge to the development of Technology CAD tools (TCAD). The level of grid generation technology currently used in TCAD tools is relatively primitive compared with the CFD community's technology. NASA Ames, a pioneer in state-of-the-art grid generation, is uniquely positioned to transfer its expertise to the TCAD community.

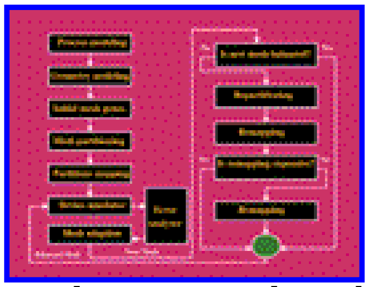
For example, device simulation requires sophisticated technology for interface layers. CFD researchers developed this technology for wing-fuselage interfaces for multidisciplinary simulation, such as fluid structures, which is routinely used in the aerospace industry and can be passed on to the TCAD community.

Impact on Accurate Solutions

Grid technology can significantly impact the accuracy of numerical solutions by improving grid generation for representing physical phenomenon with controlled and known errors arising from numerical discretization. Process simulation needs robust and efficient methods of grid generation to account for grids moving due to surface reactions. Again, the CFD community is well experienced in this technology for helicopter rotor flows. Furthermore, adaptive grid technology that is routinely used in aerospace design problems has yet to be fully developed for TCAD -- especially for use with high-performance parallel computers.

Unstructured grids have two advantages over structured grids for device modeling and process modeling. Unstructured meshes are flexible and enable grid generation around highly complex geometries, and unstructured data structures can be locally adapted to the solution. Three-dimensional grids are necessary for ultra-large-scale integration devices because of fringe and parasitic effects.

Parallel Adaptive Device Simulation



The overview of a parallel adaptive device simulator to be developed at the NAS Facility is shown in the figure. It consists of an interface to process modeling, geometry modeling, an initial mesh generator, mesh partitioner, mapper, device solver (either hydrodynamic or diffusion-drift), and adapter.

The way it might work is as follows: With input from the geometry modeler, an initial unstructured tetrahedral or hexahedral mesh is generated. This mesh is first partitioned, then mapped to the nodes of the parallel computer. A good partitioning tool balances the computational load and minimizes interprocessor communication. The device solver is used to precondition the matrix and then performs a few iterations to update solution variables. These variables are stored, after which the mesh adaptive procedure is invoked to refine and/or coarsen the computational mesh, based on an error-estimating algorithm in the device solver. Next, the old mesh is adapted locally, and a new mesh is generated.

At this point a decision is made as to whether the newly computed mesh is properly balanced among the procedures, with respect to the computational load. If it is unbalanced, repartitioning is performed to divide the new mesh into subgrids. The partitioned subgrids are then reassigned to the processors to minimize the cost of data movement. If the cost of remapping exceeds the computational gain, the new partitions are discarded and the device solver continues to work on the old partitions.

More about progress on this research will be discussed in future issues, as information becomes available.

Subhash Saini heads the NAS algorithms, architectures and applications group. He holds a Ph.D. in computational physics from University of Southern California and pursued postdoctoral studies at UC Berkeley and UCLA. Saini has been a researcher at the NAS Facility since 1989.

Next Article Contents Main Menu NAS Home

Managing Memory Problems on the CRAY C90

by George B. Myers

With increasing regularity, NAS users are experiencing one of two problems regarding memory utilization on the Cray systems. In the first case, users don't ask for enough memory in their batch requests, with the result being that jobs abort. This is frustrating because users then have to adjust the jobs and resubmit -- which means working their way back up the queue. In the second case, a job doesn't request enough initial memory when it starts up. The result is that it must move (swap) out to a temporary device each time it requests more memory, which can adversely affect throughput for both the job and the system. This article discusses how to manage these situations.

Not Enough Memory Requested

When the user hasn't requested enough memory, the seemingly simple solution is to increase the request to PBS. However, this method may require several attempts to get the right amount of memory. One way to determine how much memory a program needs is to use the size command. This command breaks a code down into three values, which represent the text (code segment), the initialized variables (data segment), and the uninitialized memory (called BSS on the Cray systems). The size command displays the total of these three values, which represents the amount of memory that will be initially requested at job startup. For example:

```
% size ptoy
ptoy: 116208 + 26941 + 81037367 = 81180516
```

This turns out to be 77.4198 megawords (MW). Setting the memory request to 78 MW is enough to correct the problem, in this example. However, with many programs today this increase will not be enough, and it does not correct the problem with swapping.

Adjusting HEAP Size Stops Swap

If the ptoy program (from the example above) was run in a batch job requesting 77.42MW it would abort. Why? Because the program has some automatic arrays that get expanded at run time -- a frequent problem with larger codes. As grid sizes grow, so do the memory requirements of the programs that process them. Many programs use dynamic memory allocation to save overall space. This places a greater demand on the space allocated as BSS, which is largely composed of the HEAP. (The HEAP is where program memory management takes place, including STACK allocation.) When dynamic arrays are in the megaword range, the initial space set aside for the HEAP may be inadequate. This is significant because when a program runs out of HEAP space it makes a request to the system for more memory. Then, that job -- and frequently, several others -- is swapped to a temporary disk while memory is rearranged to allow the job to grow. When this happens with a large job, the system experiences idle time, which impacts all users. This situation can be fixed by adjusting the initial HEAP to the maximum size needed to run the job without further memory requests.

Take the following steps to determine the initial HEAP setting. Using the ptoy program as an example:

1. Make an exaggerated guess on memory requirements based on the results of the size command, as with the example above.

2. The ja command provides data on the maximum memory used by a process. Encompass the executable with ja in the script:

   ```
   ja
   ptoy
   ja -st
   ```

3. The output from ja is displayed in megawords, which must be converted to words because that is the unit needed. One megaword equals 1,048,576 words. The ja memory value for the ptoy program converts to 81,199,104 words, which is larger than the value returned by the size command.

4. Subtract the value produced by size from that produced by ja converted to words:

   ```
   81,199,104 - 81,180,516 = 18,588
   ```

5. Increase the initial size of the HEAP space by at least this amount -- rounding up to the nearest 10,000 is not unreasonable. To adjust the HEAP, add this value to the BSS field from the size command:

   ```
   81,037,367 + 18,588 = 81,055,955
   (81,060,000 rounded)
   ```

Setting Initial HEAP Size

Memory allocation is set up by the loader. One way to set the initial HEAP size is to use the segldr command parameter -H in the form:

```
-H init,inc
```

where init is the initial size, and inc is the amount to increment the HEAP if additional memory is required. Use a number that is large enough to reduce memory requests, yet small enough to not waste memory. Using this with the example code:

```
% cf77 -Wl"-H81060000+20000" -o ptoyf toy.f
% size ptoy_fixed
ptoy_fixed:     107600 + 22784 + 81086708 = 81217092
```

In this example, the -H parameter was passed to the loader using the cf77 compiling systems.

Symptoms of Insufficient HEAP

Suppose that you asked for enough memory the first time. How do you know if your program is swapping out? One way to find out if a job is requesting more memory is to use procstat to see if that job is making calls to SSBREAK, which is a call to the system for more memory. For more information on the procstat command, see the man pages or the tutorial "Performance Monitoring Tools".

If your program uses a large amount of memory and requires swapping, the systems staff will usually notice and then contact you.

SciCon Tutorials

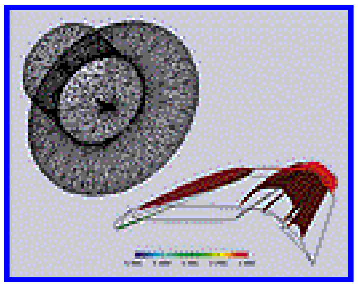
All tutorials presented by the NAS scientific consulting group during the 1997 New Operational Period training sessions given last November are available scicon@nas.nasa.gov.

George Myers works in the NAS scientific consulting group. His recent efforts have involved disseminating information through the World Wide Web, including designing the new "SciCon Tutorial" format and presenting tutorials on multitasking and performance monitoring tools. Myers has provided user consulting at Ames since1988.

Next Article Contents Main Menu
NAS Home

3D Feature Extraction Technique Helpful for Structured and Unstructured Grids

by Deborah Silver



Among the visualization tools that assist scientists in identifying phenomena in numerical simulations are those allowing users to extract and quantify certain regions within large datasets. This article discusses basic algorithms for "feature segmentation" techniques, which extract coherent amorphous regions from two- and three-dimensional scalar structured and unstructured grids. The techniques are applied to both CFD- and finite element analysis-based datasets.

Visualization provides access to new domains and allows researchers to identify observed phenomena interactively. The persistence of an unexpected effect -- an object or structure with space-time coherence that is identifiable and has a finite lifetime -- is usually a manifestation of something new. Scientists will then attempt to extract these regions of interest, visualize them, perform local measurements, classify, catalog, and abstract them for simplified representations (reduced models), and then track their evolution over time. For example, in meteorology, a scientist will track the progression of a storm or eddy.

The aim in most disciplines is to study the evolution and essential dynamics of these observable regions and describe them for modified time periods in order to obtain a partial solution, or at least a simpler model of the original problem.

Although each scientific domain has its own set of "interesting" features, generic procedures can be used to extract certain common features found in most disciplines; that is, connected components which satisfy a set of predefined criteria. The criteria can be based on any quantities, such as threshold interval, shape, vector direction, and neighborhood connectivity. This type of segmentation algorithm (similar to those used in Computer Vision) separates the dataset into "objects" and background.

To extract features in a 3D dataset, values are selected to meet a threshold criteria. A seed-growing technique can then be used to locate the features; that is, regions are grown around seeds. The initial seeds can either be user selected or be based on local extremal values. For regular gridded datasets, features can be stored in octrees, and properties such as the centroid, mass (integrated content), moments, and volume can be easilty computed. These features can be visualized using volume rendering or by fitting a surface around the boundary. If the latter approach is used, the surface can be colored according to one of the properties computed above.

For varying threshold values, the seed-growing algorithm can use the previous threshold value to grow the region (just around the boundary values). This is especially efficient when a researcher wants to determine the connectivity characteristics of the entire dataset for all threshold values; that is, number of objects versus threshold value. This can give insight into which threshold value is most appropriate and the "noise" level of the dataset.

For a curvilinear grid dataset (that is, a three-dimensional rectilinear grid in computational space that is "warped" in physical space so as to wrap around a region of interest), the object segmentation technique described above must include the periodicity of the dataset in the connectivity definition. An object extending along a boundary would be split into two objects if periodicity is not enforced. This segmentation algorithm has been implemented, and is described fully in "Object-Oriented Visualization," IEEE Computer Graphics and Applications, 15(3), 1995.

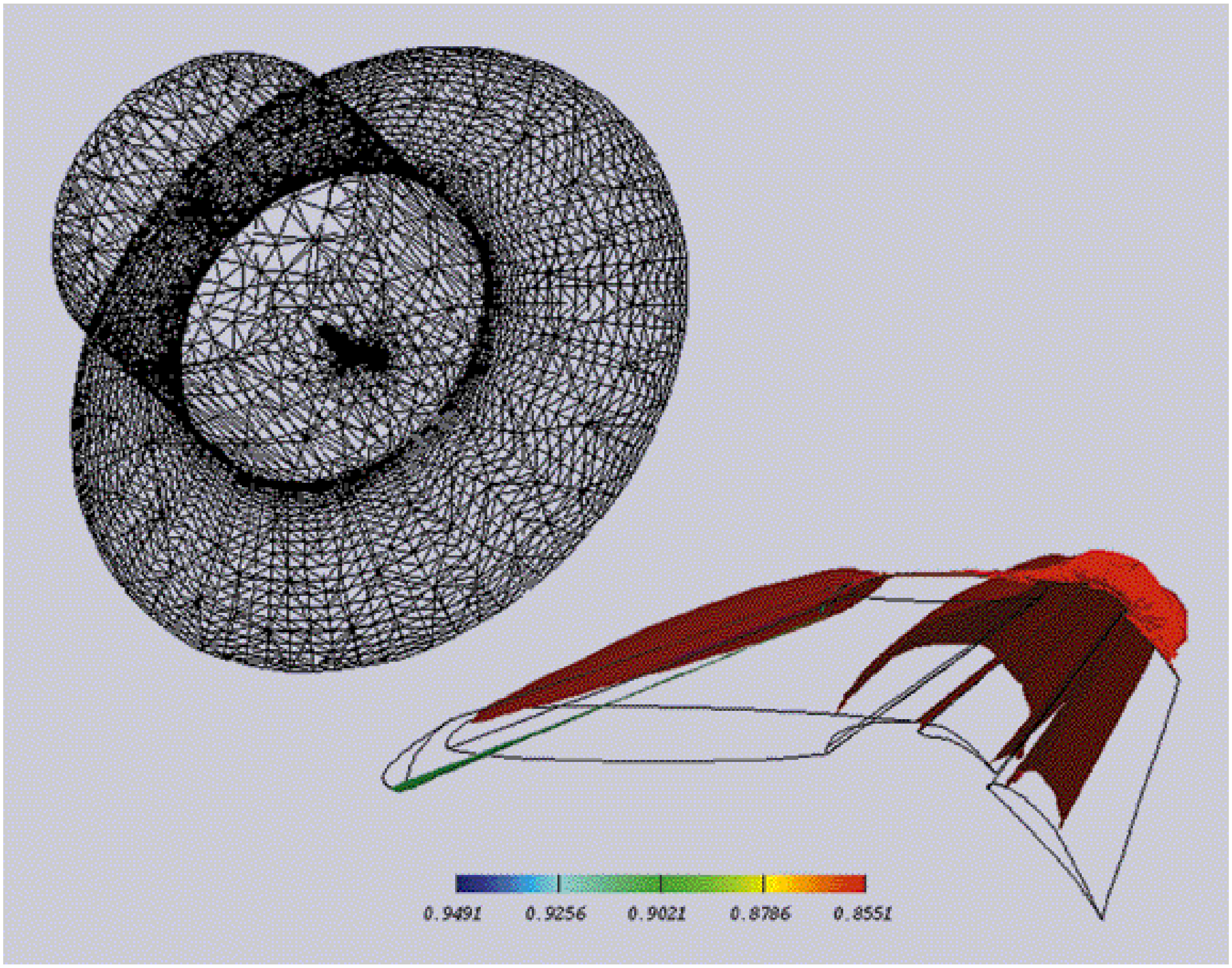
Segmenting unstructured (tetrahedral) grids is a bit more difficult because the position of the neighboring values is not implicit as with a regular dataset. The above algorithm can be implemented with the adjacency information stored explicitly. In this work, a combination of four lists has been used: the Data-List, which contains the data values; the Position-List and Cell-List, which contain indexed neighborhood information; and the Object-List, which contains a list of tetrahedra comprising each object segmented by the algorithm. The elements in each list can be indexed to the respective arrays. Examples are shown in the image above.

Features are crucial to understanding the results of complex numerical simulations. The methods discussed here allow simple feature extraction from both structured and unstructured grids. These features can now be measured and quantified to obtain meaningful results about the observable phenomena in the datasets. Most important, once features are isolated they can be tracked in time. A tracking algorithm for regular datasets has been implemented (described in two papers, "Visualizing Features and Tracking Their Evolution," IEEE Computer, Volume 27, Number 7, pp. 20-27, July 1994; and "Volume Tracking," IEEE Visualization '96 Proceedings, October 1996. Current work includes extending this algorithm to unstructured datasets.

The work described here was performed at the Laboratory for Visiometrics and Modeling, CAIP Center, Rutgers University.

Deborah Silver is an Associate Professor in the Department of Electrical and Computer Engineering at Rutgers University. This work was performed as part of a research grant at Ames Research Center.

Next Article Contents Main Menu NAS Home



This dataset was taken from the NASA Web site provided by Tim Barth and Sam Linton at Ames Research Center. It has 595,536 tetrahedra and 112,551 points. The dataset consists of data for a multiple component wing computation of inviscid compressible flow (Mach = .2 and alpha = 0 deg) and is represented in the upper left of the image. The outline of the wing is shown below. The variable being visualized is density at a threshold of .97 Kg/m^3 (and below). The surfaces are colored according to the local minima within each region.

to the article

Latest NAS Technical Reports are Online

Here are summaries of the some of the latest NAS Technical Reports, now available online.

"Performance Evaluation of Communication Software Systems for Distributed Computing," by Rod Fatoohi. Object-oriented distributed computing is well equipped to handle complex systems, and several high-speed network technologies have emerged for LANs and WANs. However, the performance of networking software is not improving as fast as hardware and workstation microprocessors. This paper gives an overview and evaluates the performance of the Common Object Request Broker Architecture (CORBA) standard in two distributed computing environments at the NAS Facility, consisting of SGI workstations connected by four networks: Ethernet, FDDI, HiPPI, and ATM. Results show that high-level communication interfaces, such as CORBA and PVM, can achieve reasonable performance under certain conditions. (NAS-96-006)

"The EMCC/ARPA Massively Parallel Electromagnetic Scattering Project," by Alex Woo and Kueichien Hill. The Electromagnetic Code Consortium (EMCC) was sponsored by the Advanced Research Project Agency (ARPA) to demonstrate the effectiveness of massively parallel computing in large-scale radar signature predictions. The EMCC/ARPA project consisted of three parts: a two-phased Program Research and Development Announcement (PRDA) to parallelize computational electromagnetics codes; a simultaneous technology development effort to complement the algorithm development of PRDA codes; and the upgrade of a massively parallel processing computer to 208 i860 GP nodes for code development and demonstration. Phase I has been completed. Overall, the project has greatly increased the computational electromagnetics capability in both algorithmic and parallel computing aspects. Parallel computing has become a common and useful tool for large-scale radar signature predictions by several U.S. aerospace companies. (NAS-96-008)

"Evaluation of Job Queuing/Scheduling Software: Phase 1 Report," by James P. Jones. The recent proliferation of high-performance workstations and the increased reliability of parallel systems have illustrated the need for robust job management systems to support parallel applications. To address this issue, a requirements checklist for job queuing/scheduling software was compiled and an evaluation begun of the leading job management system software applications against the checklist. This report describes the three-phased evaluation process, and presents the Capabilities versus Requirements results for Phase 1. Job management system support for parallel applications running on workstation clusters and parallel systems is still insufficient. However, the data provided will be useful to other sites in selecting a job management system. (NAS-96-009)

"Parallel Implementation of an Adaptive Scheme for 3D Unstructured Grids on the SP," by Leonid Oliker, Rupak Biswas, and Roger Strawn. Dynamic mesh adaption on unstructured grids is a powerful tool for computing unsteady flows requiring local grid modifications to efficiently resolve solution features. This experience of parallelizing an edge-based adaption scheme has shown good single-processor performance on the CRAY C90. For the IBM SP2, results show a 47.0X speedup on 64 processors when 10 percent of the mesh is randomly refined. Performance deteriorates to 7.7X when the same number of edges is refined in a highly localized region. However, by repartitioning the mesh immediately after targeting edges for refinement -- but before the actual adaption takes place -- the speedup improves dramatically to 43.6X. (NAS-96-011)

Next Article Contents Main Menu NAS Home

Announcing a new publication...

NAS News has been retired. In its place, a new publication called Gridpoints has been created. During the 13-year run of NAS News, the NAS Systems Division has contributed to astonighing advances in the power of high-performance computers and netqorks, the fidelity of computational fluid dynamics simulations, and the versatility of scientific visualization tools.

Now the division is looking toward a new future as NASA's primary supercomputing facility and as the lead organization for the Information Power Grid--a multidisciplinary effort to unite high-performance computers, scientific instruments, mass storage devices, and collaboration tools into a transparent, nationwide network. The scope of Gridpoints reflects this shift.

You can register for a subcription via an online form or by sending a request to gridpoints@nas.nasa.gov.

Read the Winter 1999 issue online now. (PDF, 822KB, Adobe Acrobat Reader required)

Past issues of NAS News are archived and accessible online.

`Hot Off The Press' - NPB Performance Results on Newest Systems

These figures show the results of the latest NAS Parallel Benchmarks (Version 1.0) BT (Class B).

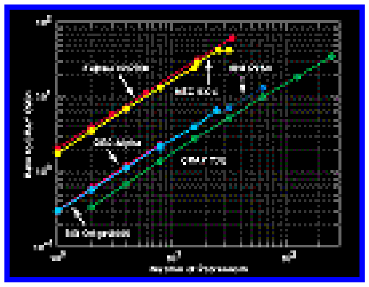


Figure 1 shows performance for the DEC Alpha Server 8400 5/440, Fujitsu VPP series (VX, VPP300, and VPP700), IBM RS/6000 SP P2SC (120 MHz), NEC SX-4/32, SGI/CRAY T3E, and SGI Origin 2000. All results have been normalized to the wall-clock time of one processor of the CRAY C90. For details see "NAS Parallel Benchmark (Version 1.0) Results 11-96," by Subhash Saini and David Bailey (Technical Report # NAS-96-18, November 1996).

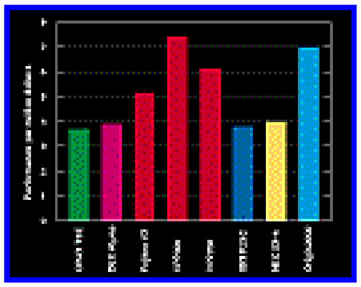
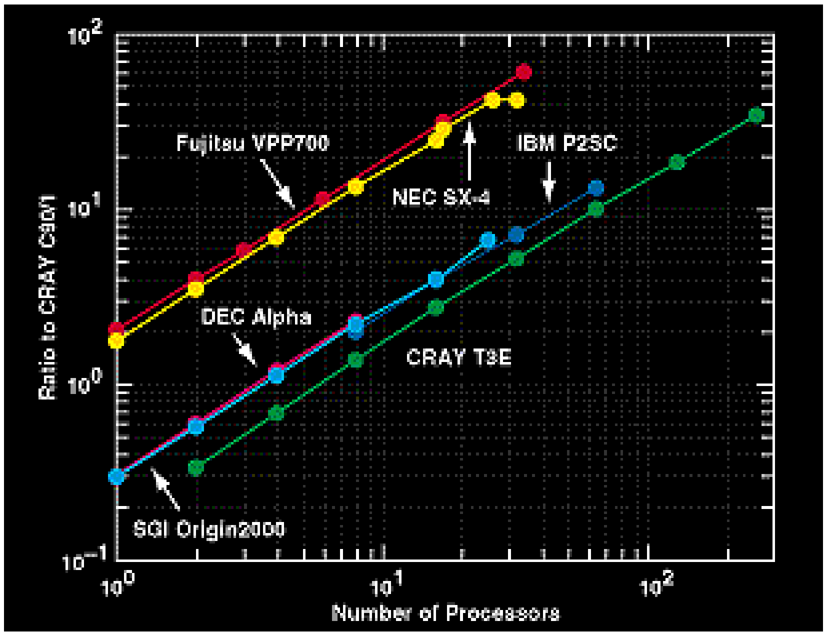
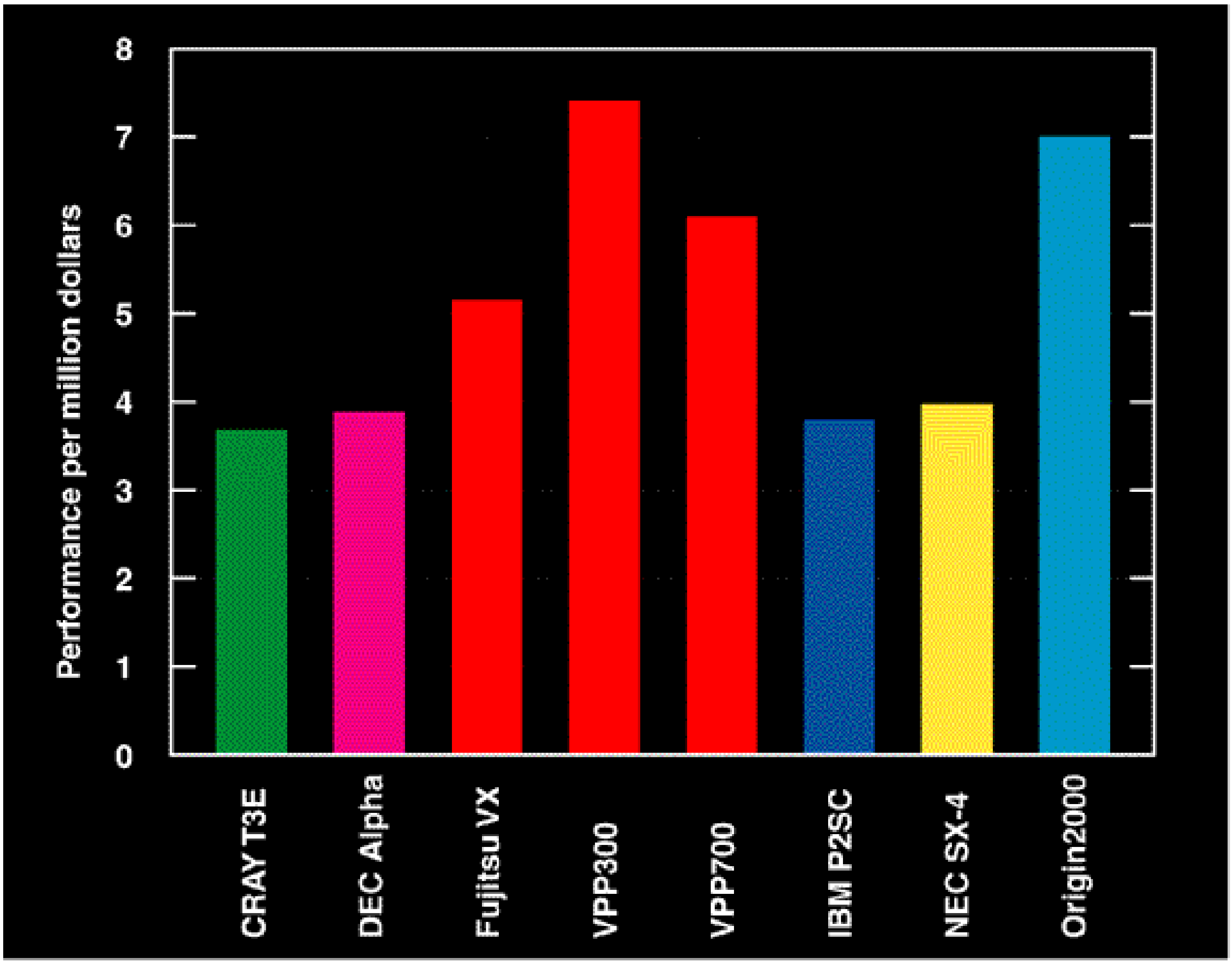


Figure 2 shows performance results per million dollars spent for various systems. This number is computed by dividing performance (ratio to CRAY C90/1) by the vendor list price in millions of dollars to run the Class B size NAS Parallel Benchmarks.

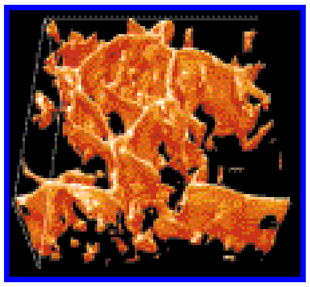
Graphics by Subhash Saini.


Ratio to CRAY C90/1
Number of Processors
Fujitsu VPP700
NEC SX-4
IBM P2SC
DEC Alpha
CRAY T3E
SGI Origin2000
10^{-1}
10^0
10^1
10^2



Performance per million dollars
0
1
2
3
4
5
6
7
8
CRAY T3E
DEC Alpha
Fujitsu VX
VPP300
VPP700
IBM P2SC
NEC SX-4
Origin2000

New Collection of Technical Seminar Videotapes Available



Here are brief descriptions of some of the more recent and well received seminars held at the NAS Facility. Videotapes of all those lised here are accessible from the NAS Documentation Center videotape loan program (send email to doc-center@nas.nasa.gov).

"Gigabit CORBA -- An Architecture for High-performance Distributed Object Computing," presented by Douglas C. Schmidt, Assistant Professor, Departments of Computer Science and Radiology, Washington University. Common Object Request Broker Architecture (CORBA) is an emerging standard for distributed computing that supports: platform, language, and network heterogeneity; flexible object location, activation, and selection; and high-level application services. However, studies show that conventional implementations of CORBA incur considerable overhead when used for bandwidth-intensive and latency-sensitive applications over high-speed ATM networks. To alleviate these performance problems, researchers are examining techniques for developing high-performance, real-time I/O subsystems, flexible and adaptive light-weight communication protocols, and streamlined ORB implementations.

"The Globus Metacomputing Environment," presented by Carl Kesselman, Information Sciences Institute, University of Southern California. Emerging high-performance applications require the ability to exploit complex supercomputing environments constructed dynamically, in real time, from geographically distributed resources. These applications use high-speed networks to integrate high-speed computers and storage devices, large databases, advanced visualization devices, and/or scientific instruments to construct networked "virtual supercomputers." The challenges arising in this environment are being addressed by the Globus project, a collaborative effort being led by Kesselman and Ian Foster (Argonne National Laboratory), which defines a two-level approach to metacomputing. A discussion of GUSTO, a medium scale testbed environment being constructed using Globus technology, concludes the talk.

Clinton Groth, Departments of Atmospheric, Oceanic, and Space Sciences, and Aerospace Engineering, University of Michigan (in joint work with Phil Roe, University of Michigan) presented "Numerical Extended Hydrodynamics for Non-equilibrium Flows at Micron-scales Encountered in MEMS/Semiconductor Design and Manufacturing." A classical problem of fluid mechanics is the satisfactory mathematical description of gasses and plasmas not in thermodynamic equilibrium, but whose particles are sufficiently numerous that tracking them individually, even in a probabilistic sense, is impractical. The mathematical features of Levermore's models are discussed, along with some newly-derived extended hydrodynamic (EHD) models based on perturbative approximations to the nonperturbative closures. Also described is the speaker's recent computational experience in applications to various large Knudsen number flow problems including shock structure and blunt body flow prediction.

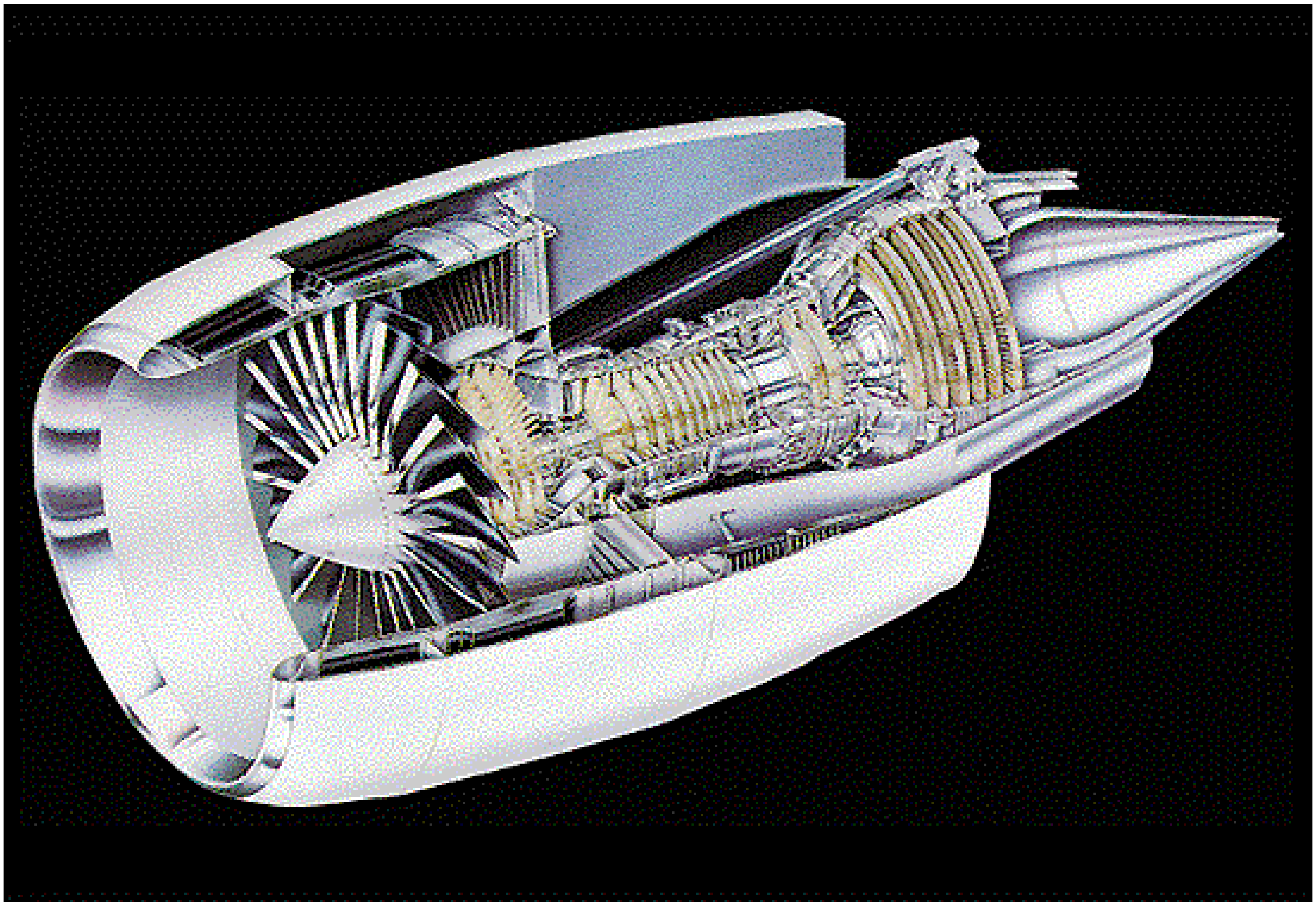
"Efficient Fair-Queueing Algorithms for ATM and Packet Networks," presented by Anujan Varma, Department of Computer Engineering, University of California, Santa Cruz. Broadband networks require the use of traffic scheduling algorithms to provide Quality-of-Service (QoS) guarantees to individual sessions. Weighted Fair Queueing (WFQ) has been considered the ideal traffic scheduling algorithm in terms of delay and fairness properties. However, the computations in a WFQ scheduler under certain conditions make its implementation difficult. A novel class of scheduling algorithms is presented. Network designers can use this methodology to implement efficient fair-queueing algorithms. Two specific algorithms in this class are especially attractive for use in high-speed networks. The underlying theory behind these algorithms and their implementation in an ATM testbed are discussed.

"The Role of Large-scale Simulations in the Understanding of the Turbulent Dynamics of the Sun," presented by Andrea Malagoli, Department of Astronomy and Astrophysics and Enrico Fermi Institute, University of Chicago. The scientific and computational challenges that must be attacked when modeling the turbulent dynamics of the solar interior, whose structure can only be inferred indirectly from helioseismological data or from observations of the solar surface are presented and discussed. The major focus is on how significant progress in this Grand Challenge problem requires a multidisciplinary approach, where advanced scientific analyses and state-of-the-art computational technology are combined to develop very-high-resolution hydrodynamical and magnetohydrodynamical numerical simulations.

"A Programming Environment for Distributed Memory Machines," presented by Alexey Lastovetsky, Lead Researcher, Institute for System Programming, Russian Academy of Sciences. The mpC language was developed to write efficient and portable programs for a wide range of distributed-memory machines. It supports task and data parallelism, allows static and dynamic process and communication structures, enables optimizations aimed at communication and computation, and supports modular parallel programming and the development of a library of parallel programs. The language is an ANSI C superset. The talk presents most principal features of mpC and its programming environment, allowing for the development of efficient and portable programs for distributed memory machines -- in particular, for workstation networks.

"Simulation of Turbomachinery Flows," presented by Mark L. Celestina, NASA Lewis Research Center. CFD is routinely used in the analysis and design of fan rotors in aeroengines. However, multistage components of aeroengines such as compressors and turbines are more complex due to the interaction of neighboring blade rows and are more difficult to analyze and design. This talk discusses the motivation and development of the Average-passage equation system that extends the use of CFD to multistage flows. An overview of the Average-passage model and its implementation in a computer code, APNASA, is presented. Details on the exploitation of the inherent parallelism, including timings from both the HPCC-funded IBM SP2 and the CRAY C90. A strategy for achieving overnight turnaround of an analysis in support of a design application of a typical multistage compressor, is also presented.

Next Article Contents Main Menu NAS Home



Cutaway view of a GE90 fan engine. Part of the presentation, "Simulation of Turbomachinery Flows", presented by Mark L. Celestina of NASA Lewis Research Center.

to the article

Next Article | Contents | Main Menu | NAS Home

January -- February 1997
Vol. 2, No. 22

Executive Editor: Marisa Chancellor

Editor: Jill Dunbar

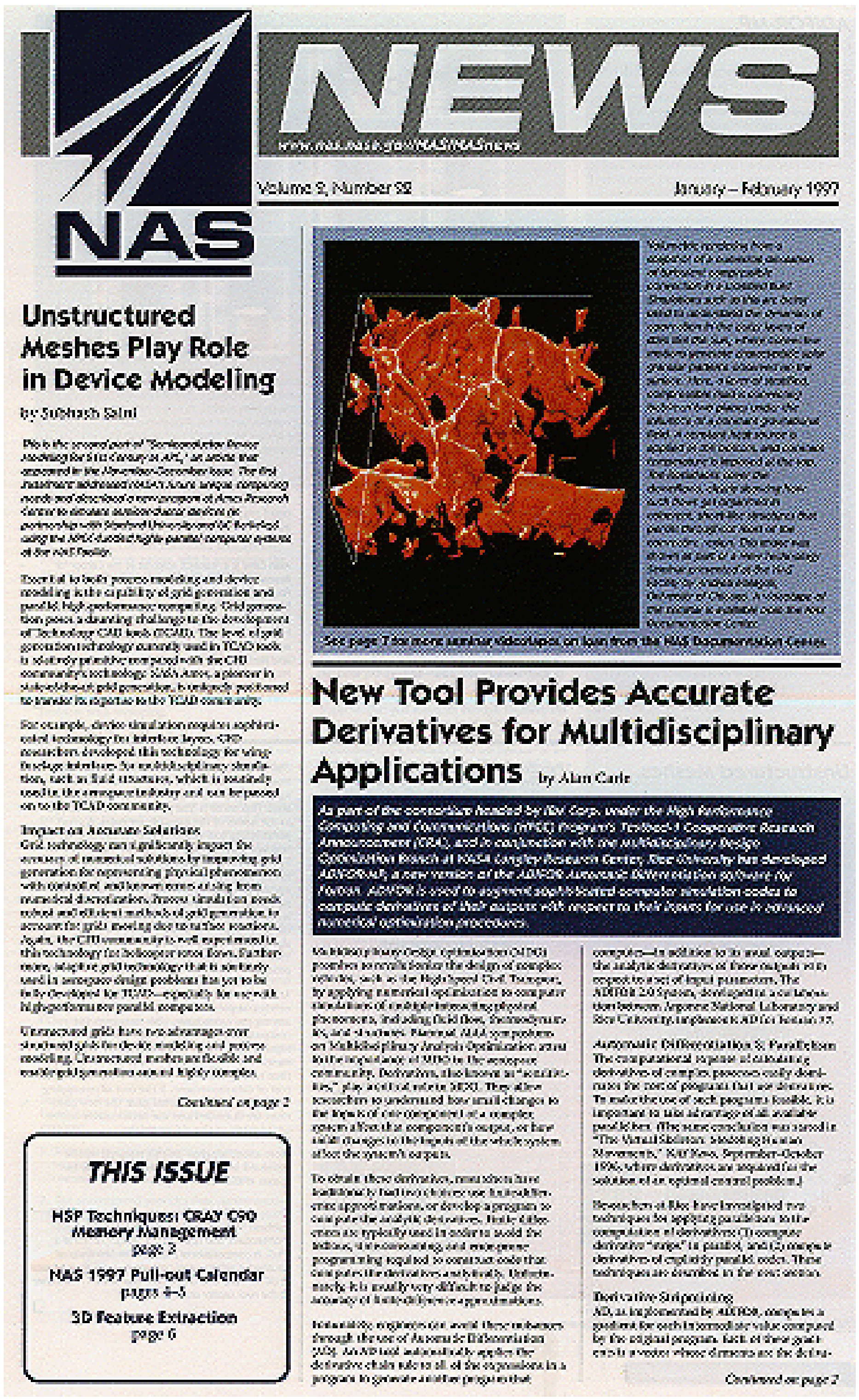
Senior Writer: Elisabeth Wechsler

Contributing Writers: Alan Carle, George B. Myers, Subhash Saini, Deborah Silver

Image Coordinator: Chris Gong

Other Contributors: Richard Anderson, Mark Celestina, James Donald, Mike Fagan, Gretchen Jacobson, Andrea Malagoli, Terry Nelson, Bill Nitzberg, Marcia Redmond, Leigh Ann Tanner, Dani Thompson, Tom I-P. Shih, Val Watson

Editorial Board: Nicholas Cardo, Marisa Chancellor, Jill Dunbar, Chris Gong, Mary Hultquist, David Kao, Chuck Niggley, Elisabeth Wechsler



NAS NEWS
www.nas.nasa.gov/NAS/NASnews
Volume 2, Number 22 January – February 1997

Unstructured Meshes Play Role in Device Modeling

by Subhash Saini

This is the second part of "Semiconductor Device Modeling for the 21st Century at ARC," an article that appeared in the November–December issue. The first installment addressed TCAD's future unique computing needs and described a new program at Ames Research Center to simulate semiconductor devices in partnership with Stanford University and UC Berkeley using the NAS-funded highly parallel computer systems at the NAS Facility.

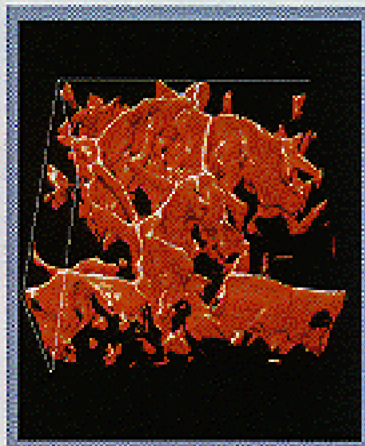
Essential to both process modeling and device modeling is the capability of grid generation and parallel, high-performance computing. Grid generation poses a daunting challenge to the development of Technology CAD tools (TCAD). The level of grid generation technology currently used in TCAD tools is relatively primitive compared with the CFD community's technology. NASA Ames, a pioneer in state-of-the-art grid generation, is uniquely positioned to transfer its expertise to the TCAD community.

For example, device simulation requires sophisticated technology for interface layers. CFD researchers developed this technology for wing-fuselage interfaces for multidisciplinary simulation, such as fluid structures, which is routinely used in the aerospace industry and can be passed on to the TCAD community.

Impact on Accurate Solutions
Grid technology can significantly impact the accuracy of numerical solutions by improving grid generation for representing physical phenomena with controlled and known errors arising from numerical discretization. Process simulation needs robust and efficient methods of grid generation to account for grids moving due to surface reactions. Again, the CFD community is well experienced in this technology for helicopter rotor flows. Furthermore, adaptive grid technology that is routinely used in aerospace design problems has yet to be fully developed for TCAD—especially for use with high-performance parallel computers.

Unstructured grids have two advantages over structured grids for device modeling and process modeling. Unstructured meshes are flexible and enable grid generation around highly complex

Continued on page 2



Volumetric rendering from a snapshot of a numerical simulation of turbulent, compressible convection in a stratified fluid. Simulations such as this are being used to understand the dynamics of convection in the outer layers of stars like our Sun, where [illegible] This image was shown as part of a NAS technology seminar [illegible] University of Chicago. A videotape of the seminar is available from the NAS Documentation Center.

See page 7 for more seminar videotapes on loan from the NAS Documentation Center.

New Tool Provides Accurate Derivatives for Multidisciplinary Applications

by Alan Carle

As part of the consortium headed by IBM Corp. under the High Performance Computing and Communications (HPCC) Program's Testbed-1 Cooperative Research Announcement (CRA), and in conjunction with the Multidisciplinary Design Optimization Branch at NASA Langley Research Center, Rice University has developed ADIFOR/MP, a new version of the ADIFOR Automatic Differentiation software for Fortran. ADIFOR is used to augment sophisticated computer simulation codes to compute derivatives of their outputs with respect to their inputs for use in advanced numerical optimization procedures.

Multidisciplinary Design Optimization (MDO) promises to revolutionize the design of complex vehicles, such as the High Speed Civil Transport, by applying numerical optimization to computer simulations of multiple interacting physical phenomena, including fluid flow, thermodynamics, and structures. Biennial AIAA symposiums on Multidisciplinary Analysis Optimization attest to the importance of MDO in the aerospace community. Derivatives, also known as "sensitivities," play a critical role in MDO. They allow researchers to understand how small changes to the inputs of one component of a complex system affect that component's output, or how small changes to the inputs of the whole system affect the system's outputs.

To obtain these derivatives, researchers have traditionally had two choices: use finite-difference approximations, or develop a program to compute the analytic derivatives. Finite differences are typically used in order to avoid the tedious, time-consuming, and error-prone programming required to construct code that computes the derivatives analytically. Unfortunately, it is usually very difficult to judge the accuracy of finite-difference approximations.

Fortunately, engineers can avoid these [illegible] through the use of Automatic Differentiation (AD). An AD tool automatically applies the derivative chain rule to all of the expressions in a program to generate another program that computes—in addition to its usual outputs—the analytic derivatives of these outputs with respect to a set of input parameters. The ADIFOR 2.0 System, developed in a collaboration between Argonne National Laboratory and Rice University, implements AD for Fortran 77.

Automatic Differentiation & Parallelism
The computational expense of calculating derivatives of complex processes easily dominates the cost of programs that use derivatives. To make the use of such programs feasible, it is important to take advantage of all available parallelism. (The same conclusion was reached in "The Virtual Skeleton: Modeling Human Movements," NAS News, September–October 1996, where derivatives are required for the solution of an optimal control problem.)

Researchers at Rice have investigated two techniques for applying parallelism to the computation of derivatives: (1) compute derivative "strips" in parallel, and (2) compute derivatives of explicitly parallel codes. These techniques are described in the next section.

Derivative Stripmining
AD, as implemented by ADIFOR, computes a gradient for each intermediate value computed by the original program. Each of these gradients is a vector whose elements are the deriva-

Continued on page 2