Zeitschrift: Helvetica Physica Acta
Band: 61 (1988)
Heft: 1-2

Artikel: ASTRID : a family of scientific applications programs
Autor: Gruber, Ralf
DOI: https://doi.org/10.5169/seals-115929

Helvetica Physica Acta
Vol. 61 (1988)
0018-0238/88/020024-06$1.50+0.20/0

ASTRID : A family of scientific applications programs

ASTRID group, presented by Ralf GRUBER, GASOV-EPFL, Bat. DMA, 1015 Lausanne.

Abstract:

ASTRID is an interdisciplinary project that aims at implementing a set of numerical experimentation programs adapted to the architecture of parallel vector computers with huge central memories. These supercomputers are directly linked to graphic workstations or PC's which are used to input data and output results. Particular emphasis is put on designing user friendly, easily maintainable applications programs.
In a first step, four pilot applications will be realized; eight others will follow in the near future.

1. INTRODUCTION

ASTRID (French acronym for Scientific Applications related to Research, Industry and Development) designates an interdisciplinary project under way at E.P.F.-Lausanne. It aims at implementing simulation codes that model realistic three dimensional physical phenomena.
The purpose of ASTRID is to centralize the efforts now distributed in the scientific community in the development of simulation software. Software systems for simulation and modeling for different physical phenomena usually have a similar structure. The goal of ASTRID is to concentrate on the common elements of such systems: grid generation, numerical analysis, parallel algorithms, graphical visualisation and user interface and to design and implement a system which unifies these elements into a coherent package.
The ASTRID members act also as consultants for research groups which submit applications to be analyzed. Each application submitted to ASTRID is split into an application specific part (which remains the responsibility of the original design team) and a part common to the original design team and the ASTRID nucleus of programmers. The latter will advise, counsel, design and implement solutions in close co-operation with the former team. The ASTRID nucleus is then responsible for the advancement of the common part of the project (as an example, the common part in a 3-D fluid flow simulation comprises of the mesh generator, the solver and the optimized matrix operations package).
Conceptually speaking, an ASTRID program runs three virtual machines : the input computer (PC for instance), the computational unit (such as a supercomputer for instance) and an output machine.with sophisticated graphic capabilities. Given a common resolution technique, most parts of various codes will be identical or similar. The basic belief of ASTRID is that all the computationally intensive parts of simulation programs are adapted to parallel vector computer architectures. Natural parallel processing can be achieved by decomposing the computational domain into sub-domains subject to geometrical, physical or computational arguments. In the case of a discrete approach such as finite differences, finite elements or finite volumes, a structured mesh is applied to discretize each of the subdomains leading to structured matrices. In order to avoid matrix structure modifications, boundary conditions are introduced via matrix transformations and by adding dummy variables. The system of linear equations is solved iteratively by a conjugate gradient method with à sub-domain preconditionning. Within one subdomain, the construction of the mesh and the matrices aa well as the matrix resolution are performed in vector mode. Much effort is done to build large tasks by concentrating most of the computations within each of the subdomains.

Since the ASTRID concept is organized around an input, a computation and an output virtual machine, databases are required to be easily transportable from one computer to the other. Great attention is given to achieve a high level of convenience in the use of the codes. This implies highly interactive menu driven man-machine interfaces to define the input quantities

and to select the graphical modules that display the solutions. An interactive documentation system directly linked to the subroutines provides the user with a HELP facility and eases maintenance of the system. Specific graphical representations and animation are developed for each application. Standard graphical and mathematical libraries can be used to interpret the results.
We present in section 2 the applications submitted for realization within the ASTRID project and in section 3 the computer science and mathematical approaches developed by the ASTRID nucleus for their solution. In the closing section, we address some ideas about future applications of simulation software in industry.

2. THE ASTRID APPLICATIONS

The main goal of ASTRID is to develop the common mathematical analysis and and computer science parts of computer simulation programs of different research groups. At the moment, twelve groups have submitted proposals. They are:

A1 : Three-dimensional viscous flow modeling. Turbulent boundary layers.
. Optimization of a water turbine including water entrance, flow preparation (turbo), flow through the turbine and water exit.
. Study of reentrant phase of the european space shuttle HERMES.

A2 : Three-dimensional incompressible stationnary flow of two fluids separated by a free surface.
. Study of the magnetohydrodynamic instability in aluminium ovens in order to to reduce energy consumption.

A3 : Numerical simulation of electronic properties in materials.
. Study of mobility of electrons in periodic systems with a surface boundary. Simulation of semiconductors and superconductors.

A4 : Solidification processes.
. Free boundary electromagnetic moulding of aluminium.

A5 : Numerical concrete.
. Diffusion of acid rain into concrete.

A6 : Three-dimensional stochastic simulation of polycrystals.
. Crystal structures as a function of cooling in the moulding process.

A7 : Energy and air flows in buildings.
. Optimization of heating, wall insulation and ventilation.

A8 : Three-dimensional simulation of nonaxisymmetric fusion machines.
. Study of the stability behaviours of Stellerators and non-axisymmetric Tokamaks.

A9 : Laser-material interaction.
. Simulation of a laser fusion experiment. Surface treatment by lasers.

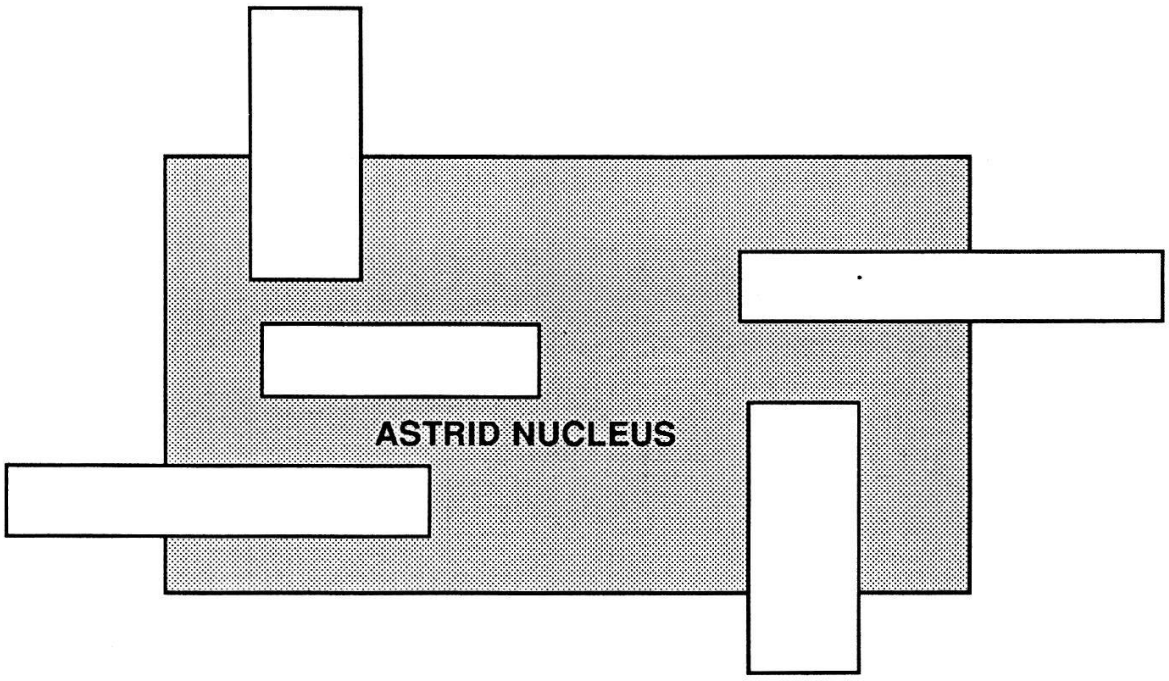
A10 : Plasma particle simulation.
. Fundamental study of the behaviour of a three-dimensional dense plasma in an electromagnetic field.

A11 : Three phase porous media.
. Security study of nuclear power plants.

A12 : Adaptative solution for elastic problems.
. To improve precision, higher order finite elements are locally included.

Applications A1, A2, A3 and A6 have been selected as "pilot projects". They are presently under development by the ASTRID nucleus. They reflect the variety of simulation programs which range from finite element and finite volume methods to global expansion functions and Monte Carlo simulations; these projects require the solution of stationary time-dependent and eigenvalue problems, and enable us to create a common software adapted to most of the proposed applications.

The development work on these proposed applications is done jointly by the original design team and the ASTRID group. The figure below illustrates this cooperation: parts of the applications are realized by the ASTRID nucleus whereas the regions situated outside the frame represent the specific parts of the application related to each research institute. It is obvious that such an ambitious project only can succeed through a very close co-operation between the different groups.



3. THE ASTRID NUCLEUS

Here we describe the different activities of the engineers who form the ASTRID nucleus in a logical order :

(a) Description of the objects and the subdomains

The first step

Simulation codes presume a pre-defined physical domain of computation. This definition of the object is often done using a CAD system, which is, in general, difficult to interface. Since we prefer not to depend on such expensive and very "non-standard" software, we decided to

develop our own "mini-cad". In this small system we ask for the minimum amount of information to describe the objects in a precision which does not exceed that of the simulation result or that given by manufacturing limits. The geometry is presented to the meshing algorithm as a dense grid of points - in such a way the representation is independent of any particular object modeling technique and is general enough to allow for an interface to any CAD system The object is decomposed into several subdomains, each of which is topologically a cube. This decomposition is done according to three main criteria: the geometry of the object, the physics of the problem being solved and the architecture of the target machine. The geometrical arguments are mostly given by the automatic mesh generator which does not like too exotic forms. Physical reasons are the existence of shock fronts, boundary layers or boundary conditions. If the computer to be used is highly parallel it is evident that the number of subdomains will be adjusted to the number of available processors.

(b) Adaptive structured mesh

Each deformed cube representing a subdomain is discretized with a structured mesh. This implies that the mesh has to be created hierarchically. First the 12 edges of the cube are discretized in such a way that the number of intervals is the same for 4 "parallel" edges. In a second step, each of the 8 surfaces is discretized with a rectangular grid using a method based on molecular dynamics [1]. The discretized surfaces are stored in the database and are not further modified. This is necessary in order to guarantee continuity of the mesh between the subdomains. Finally, a three-dimensional structured mesh is generated for each of the subdomains. Since these operations are independent, they can be performed in parallel.
Once the system has been solved with an initial mesh, the solution vector is used to define an adequate mesh density function. This function is then used to generate a second mesh, better adapted to the final solution. The ultimate goal is to have a mesh leading to a homogeneous error distribution. Remeshing is also necessary if the physical quantity determining a subdomain boundary varies, as in a time dependent or an iterative process.

(c) Matrix construction

The possibility of adjusting the subdomains and the meshes during computation makes it necessary to have a very efficient matrix construction algorithm. Instead of sequentially computing the matrix contributions for one cell after another, we calculate one single contribution after another for all the cells at once. Such a procedure needs much more memory space but is highly vectorizable. We do not see memory size as a restriction for such a procedure since we believe that there will soon be in Switzerland a supercomputer with a huge central memory.

(d) Matrix solvers

In all finite element or finite volume approaches time steps are handled implicitely. This implies that one has to solve a system of linear equations

$$\mathbf{A}\mathbf{x} = \mathbf{b} \qquad (1)$$

In practice, $\mathbf{A}$ consists of as many submatrices $\mathbf{A}_i$ as there are subdomains and $\mathbf{x}$ and $\mathbf{b}$ have as many subvectors $\mathbf{x}_i$ and $\mathbf{b}_i$ as subdomains. The connectivity conditions between subdomains are given by a special vector $\mathbf{x}_0$ which includes all the variables belonging to at least two subdomains.

If system (1) is solved by a direct matrix solver, the connectivities have to be included in the matrix leading to

$$\begin{array}{|ccc|c|} \multicolumn{3}{|c|}{\begin{matrix} A_1 & & \\ & \ddots & \\ & A_i & \\ & & \ddots \\ & & A_N \end{matrix}} & \begin{matrix} B_1 \\ \vdots \\ B_i \\ \vdots \\ B_N \end{matrix} \\ \hline C_1 \; . \; . \; C_i \; . \; . \; C_N & & & A_O \\ \hline \end{array} \quad \begin{array}{|c|} \hline x_1 \\ \vdots \\ x_i \\ \vdots \\ x_N \\ \hline x_O \\ \hline \end{array} \quad = \quad \begin{array}{|c|} \hline b_1 \\ \vdots \\ b_i \\ \vdots \\ b_N \\ \hline b_O \\ \hline \end{array} \qquad (2)$$

The vectors $\mathbf{x}_i$ include all the internal variables of the i th subdomain. System (2) is solved in three steps. First we perform Gauss elimination by

$$\mathbf{D} = \mathbf{A}_O - \sum_{i=1}^{N} \mathbf{C}_i \, \mathbf{A}_i^{-1} \, \mathbf{B}_i \, , \quad \mathbf{d} = \mathbf{b}_O - \sum_{i=1}^{N} \mathbf{C}_i \, \mathbf{A}_i^{-1} \, \mathbf{b}_i \, , \qquad (3)$$

then solve for

$$\mathbf{D} \, \mathbf{x}_O = \mathbf{d} \qquad (4)$$

and by back substitution compute

$$\mathbf{x}_i = \mathbf{A}_i^{-1} (\mathbf{b}_i - \mathbf{B}_i \, \mathbf{x}_O). \qquad (5)$$

The computationally intensive operations in (3) and all of the operations in (5) can be performed in parallel. Instead of inverting the A_i in (3), a matrix decomposition is performed which keeps the band structure unchanged. However, the zero diagonals in between a band are filled in. In particular, in the three-dimensional case, the band becomes very wide. In addition, the number of connectivity points can become large if many subdomains are chosen to reduce the computational effort in (3). Then most of the computing time has to be invested to solve the full system (4). In view of these problems, we are convinced that in practice, direct matrix solvers are not adequate for three-dimensional problems; therefore we choose to investigate iterative techniques.
To solve (1) iteratively we apply a **conjugate gradient (CG) method** with a domain decomposition preconditioning technique. The major operation in a CG method is a matrix-vector multiply

$$\mathbf{v} = \mathbf{A}\underline{\mathbf{y}} \qquad (6)$$

where $\mathbf{y}$ is an estimate of $\mathbf{x}$. This operation (6) is done in three stages. First execute in parallel

$$\mathbf{v}_i = \mathbf{A}_i \, \mathbf{y}_i \qquad (7)$$

disregarding the connectivities.Then, sum up into $\mathbf{v}_O$ all the connectivity terms of all the $\mathbf{v}_i$ and, finally, redistribute back the right $\mathbf{v}_O$ components into the $\mathbf{v}_i$. This means that the connectivity conditions are applied to the partial vectors $\mathbf{v}_i$ and not to the matrices $\mathbf{A}_i$.

(e) Graphical representations

The graphical display chosen for the interpretation of physical results depends on the specific application solved. The main representations will be x - y plots, hidden line x - y - z plots, colouring of objects and particle simulation in a force or velocity field.

(f) Database structure

The database for finite element or finite volume applications stores the definition of the object specified by the user, the grid generated by the meshing algorithm, the matrices and vectors specific to the method of resolution and the final results, in a form convenient for graphical presentation. Some of this data must flow from one machine to the other. Typically, the grid data must be available on all three virtual machines.
We are using the MEMCOM [2] database system .

(g) Documentation system

Since the software developed in ASTRID is a cooperative effort of many research scientists, programming norms have been set to make for easier integration and maintenance of the codes. A subroutine consists of an initial documentation section followed by the subroutine body. The documentation section gives a detailed description of the subroutine, the name of the author and a log of version updates. An interactive documentation system has been written which allows for the retrieval of the program documentation sections by keywords.

4. Outlook

The school project ASTRID started in September 1986 and will last for 4 years. Altogether, 10 engineers are involved in the development of the software package. In close co-operation with the ASTRID nucleus, about two engineers per application are also involved in the realization of their projects.
A first prototype of the entire ASTRID system will be available in March 1988. From there on, more emphasis will be put on the validation and the user-friendliness of the applications so that the system can reach industrial standards for simulation software. We intend to develop an easy input of data, simple to modify thus allowing a fast product optimization; moreover, highly evolved HELP systems (e.g. interactive documentation) and an expert system will be developed.
The expert system we have in mind would be a learning system for the efficient use of a given code. The idea is that the achieved knowledge of an experienced user of a simulation program should not be lost if the user leaves. Informations on convergence properties, interpretation of solutions and their behaviour as a function of different parameters as well as a database-oriented result storage system should be easily accessible by a non-experienced user. He will then have the opportunity to rapidly learn how to use such a code and how to interpret the computed results.
In order to become as independent as possible from a specific parallel vector computer architecture we decided to concentrate most of the computational effort in small modules. These modules can then be highly optimized for each computer by the computer manufacturers. This is our way to become computer independent.

References

]1] E. Bonomi, "Generation of structured adaptive grids based upon molecular dynamics".
[2] S. Merazzi, P. Stehlin, MEM-COM User Manual, Version 5.4, SMR Corp., Bienne, Switzerland (1987).

RG/av/Mac192/October 28, 1987