Ferrocene: An example of the use of a multiview visualization system (Hersch et al., p.12).
DATA-PARALLEL MOLECULAR DYNAMICS: 1-D HARD-CORE FLUID

By Ernesto Bonomi — CFD/MPC Project, IMHEF-EPFL
& Marco Tomassini, Section Logiciels, SIC-EPFL

Les méthodes de dynamique moléculaire permettent de simuler par ordinateur des systèmes physiques complexes. Avec la disponibilité actuelle de machines massivement parallèles, la voie est ouverte pour des simulations comportant des centaines de milliers de particules, ce qui devrait améliorer la fiabilité des résultats. La dynamique moléculaire de sphères rigides offre des avantages mais étant intrinsèquement asynchrone, elle est plus difficile à paralléliser ou vectoriser. Nous avons modélisé un système simple, le gaz monodimensionnel de corps durs, sur l'ordinateur massivement parallèle Connection Machine CM-2.

Molecular dynamics constitute a family of techniques for solving large classical N-body problems under a variety of physical conditions. Modern massively parallel machines allow us to approach the simulation of matter at the atomic level, thus enlarging the scope of computer modelling. Hard-sphere impulsive evolutions are intrinsically asynchronous and do not parallelize easily on multiprocessors or vector machines. Using the 1-D hard-core fluid as a paradigm, a data-parallel molecular dynamics algorithm with at least 16384 particles has been implemented on a Connection Machine CM-2.

INTRODUCTION

Molecular dynamics are well-known computational techniques that have gained wide acceptance in the simulation of gases, liquids and solids, both in and out of statistical equilibrium. This approach consists of numerically solving the simultaneous equations of motion of many interacting particles for a dynamical evolution which conserves total energy. On a finite number of time steps, the simulation explores a subset of admissible configurations. According to ergodic theory, time averages on the phase space trajectory allow the evaluation of the expectation values of any dynamical observable at equilibrium.

To be faithful to the macroscopic descriptions of matter, molecular dynamics studies must use a large number of particles and must take into account boundary effects which play an important role, for instance in the study of critical phenomena. It is interesting to explore the applicability of parallel computer architectures to such simulations since even on modern workstations these simulations can take days, if not weeks.

A coarse-grain view of molecular dynamics suggests the decomposition of the container into spatial subdomains, each one of them managed by a dedicated processor. This approach is well adapted to the simulation of regularly distributed populations of particles with short range interactions, giving rise to a loosely synchronous parallelism in which the integration time step acts as a macroscopic synchronization barrier [1]. When the distribution of particles is non homogeneous, such as in a fluid, problems of load balancing begin to appear, requiring dynamic reallocation of particles among processors [2].

Hard-core molecular dynamics cannot be treated by the above synchronous approach because of the discontinuous nature of the binary potential [3]: the attractive part of the two-body potential is simply neglected and the repulsive contribution is reduced to a discontinuity representing the spatial extension of the core of a rigid sphere. Hence each particle is driven by a constant force except at the instant of an elastic collision with another particle. The search for the colliding pairs and the computation of the elapsed time between subsequent collisions makes a hard-core fluid simulation an asynchronous, event-driven process difficult to parallelize efficiently. Hard-core molecular dynamics imply no intervening numerical methods and, consequently, the simulation is exact up to the computer precision. To our knowledge, no satisfactory parallel simulation of this kind has been published to date.

In an attempt to contribute to the study of parallel hard-core molecular dynamics simulations, we set out to investigate a very simple system, the finite one-dimensional hard-core fluid [4], using a highly parallel machine: the CM-2. This system has the advantage of being simple and well understood. Although the model is probably the simplest one in equilibrium statistical mechanics, it is interesting as a prototype because it has many of the features of more realistic 2-D or 3-D systems without their added complexities. It is thus an ideal testbed for studying the implementation of a new parallel approach to a hard-core molecular dynamics simulation. Moreover, the generalization to higher dimensions is straightforward.

THE N-particle dynamics

In a one dimensional hard-core fluid confined in a linear box [0,1], N particles of diameter R move freely between collisions without overlapping. For any admissible initial configuration of positions and velocities, the time to impact
EDITORIAL

The «EPFL Supercomputing Review» is published annually with the purpose of presenting to academia, industry, and the general public, in Switzerland and abroad, the work carried out by researchers in Switzerland on supercomputers. In this, the fourth year of its publication, we are seeing that while «traditional» vector computers such as our own Cray-2 still account for the lion’s share of supercomputing in this country, massively parallel and specialized computer architectures are becoming more and more popular amongst researchers. You will find more articles on these various architectures than in previous issues, plus we have tried to put more detail into the algorithms and calculations used.

The articles presented here represent only a fraction of the computation-intensive research taking place in Switzerland. The future of supercomputers in this country is directly dependent on the health of its research, and vice versa.

We would like to thank both Jacqueline Dousson and Aymar de Brossin of the Section Assistance for their help in reproducing some of the colour images displayed in this review. We would also like to thank Cray Research (Switzerland) SA for its financial contribution to the printing of this issue.

CONTENTS

Data-Parallel Molecular Dynamics: 1-D Hard-Core Fluid

Dynamique moléculaire massivement parallèle: fluide de corps durs monodimensionnel

Applications of Supercomputers in the Development of Gyrotrons

Les superordinateurs appliqués au développement des gyrotrons

Performances of a Multiprocessor Multiwindow Visualization Subsystem

Performances d’un sous-système de visualisation et de fenêtrage à processeurs multiples

Network Supercomputing

Superordinateurs en réseau

Computational Fluid Dynamics on Massively-Parallel SIMD Computers

Calculs d’écoulement sur ordinateurs massivement parallèles SIMD

Tokamak Disruptions and Modelling of Complex Nonlinear Plasma Evolution by Supercomputers

Disruption dans les tokamaks et modélisation de l’évolution complexe et non linéaire du plasma par superordinateurs

TERPSICHORE: a three-dimensional ideal magnetohydrodynamic stability program

TERPSICHORE: un programme de stabilité magnétohydrodynamique idéale tridimensionnelle

EDITOR:

STEWART WILLIAMS, SIC-EPFL

CO-EDITOR:

MARIE-CATHERINE STIBEL, SIC-EPFL

TEXT PROCESSING AND LAYOUT:

APPARO INSPEKT/SIC-EPFL

ADDRESS:

SERVICE INFORMATIQUE, CENTRAL EPFL

MA-EPFL, CH-1015 LAUSANNE

PHONE:

(021) 693 22 11

FAX:

(021) 693 22 20

E-MAIL:

williams@sic.epfl.ch
for collisions between each pair of particles is the smallest, non-negative root of a quadratic equation. The minimal time over this collection of potential collisions determines the moment of the next event. Each particle moves then by an amount determined by the product of its current velocity with the computed time to collision. At the end of this step, the particles in the colliding pair simply exchange velocities, conserving energy and momentum. In case of contact of a border particle with the rigid wall, the particle velocity is simply reversed.

The simulation proceeds collision-by-collision, according to the following sequential algorithm outline:

0: Initialize all particle positions \( x_i \) and velocities \( v_i \).
1: Schedule in a list all admissible collision times between pairs of nearest particles.

loop while not done
2: Find the first minimum scheduled time to impact \( t_{\text{min}} \).
3: Move all particles: \( x_i \leftarrow x_i + v_i t_{\text{min}} \).
4: Compute new velocities for the colliding pair.
5: Update the collision list.
6: Calculate interesting observables for future use.

end loop

A priori, there is little room for parallelization in the above algorithm. Actually, time intervals between subsequent collisions are randomly distributed and only one collision event occurs at a given time, step 4. If we try to process a whole block of events in the collision list at once, there is a risk of altering the dynamics, since scheduled events are interdependent. However, hard-core fluid molecular dynamics can efficiently and naturally be simulated in parallel, provided that we use a different point of view, taking into account the novel features offered by massively parallel machines.

**Data-Parallel Approach**

Before going into the details of the parallel algorithm, it is useful to take a brief look at the Connection Machine architecture and its programming model [5]. Here we will simply stress the main points.

- The CM-2 is a SIMD (Single Instruction Multiple Data) computer with many, from 4K to 64K, 1-bit processors. Moreover, a fully configured CM-2 contains 2048 floating-point chips and a local memory of 64 or 256 Kbits per processor. The same program instructions are broadcast to all the processors by a sequencer and executed in lockstep. There is no globally accessible memory.
- The CM-2 features fast data communications between neighbouring processors in regular patterns (NEWS communications).
- Due to the hypercube network interconnection, any processor can communicate with any other, although these general communications are slower than using NEWS since they require routing and congestion control. These control activities are transparent to the application programs.

- A variable that is spread throughout the processor memories is called a parallel variable. An example would be a vector or a matrix. Since there are many processors available, the user is encouraged to associate one datum to a single processor.
- The programming model is based on the so-called data-parallel paradigm, which essentially consists of performing the same operation synchronously on many different data embodied by a parallel variable.
- The concept of a virtual processor helps when there are more data than processors, a common situation in large problems. Actually, a physical processor can simulate a number of virtual processors. A program can use any suitable number of these virtual processors in a transparent manner. It can then be executed unchanged on CM-2 systems of different sizes with only a performance penalty when there are fewer physical processors.
- Input-output to parallel files is provided, as well as the possibility of rapidly visualizing large data sets. These are important features for computer simulations producing a large quantity of results.

The Connection Machine includes software support of a geometry: an organization of virtual processors into an \( n \)-dimensional lattice, \( n=1,2,\ldots,31 \). In this way many common data layouts can be easily accomodated. For the 1-D hard-core fluid, the natural choice of geometry is a string of virtual processors, each of them controlling a particle with its attributes: position, velocity and the various flags required by the simulation.

The implementation strategy for our molecular dynamics in parallel is somewhat different from the sequential point of view, where the collision schedule must be maintained. On the CM-2, it is easier and more natural to re-evaluate in parallel the next collision event at each iteration. This is not only more efficient for the CM-2, it also simplifies the design of the algorithm. In addition, the possibility of multiple binary collisions occurring simultaneously, normally difficult to deal with in the sequential scheme, is automatically solved.

**Parallel Algorithm**

The following is a pseudo-code description of the parallel algorithm which has been written in C/Paris, an intermediate-level language for programming the CM-2:

```plaintext
create a string of virtual processors: each processor controls a particle; the first and last processors represent the left and right boundaries.
for all processors do in parallel
  0: initialize particle positions \( x_i \) and velocities \( v_i \)
  1: initialize flags
end parallel do
loop while not done
  for all processors do in parallel
    2: get the position and the velocity of the left and right particles
```
We will now discuss the above parallel algorithm step by step. In step 0 we generate in parallel a regular distribution of particles with randomly chosen initial velocities. In step 1 a number of Boolean flags representing particle attributes are initialized for each processor. One flag is used to identify the rigid boundaries; this wall flag is set on for the end processors (the walls) and set off everywhere else. In addition, there are two flags, right and left, that are set to 0 everywhere before each iteration and are used to identify the colliding pairs of particles during the iteration. The operations on those 1-bit fields are efficiently and conveniently implemented in CM programming.

![Figure 1: NEWS communications of particle positions and velocities](image1)

The molecular dynamics begin with step 2. To decide if and when a given particle collides with a neighbour, a processor must know the relative position and velocity of its particle with respect to its nearest neighbours. Every processor, including those representing the walls, receives the left and right particle positions and velocities in parallel via the NEWS primitives. This synchronous communication is done quickly and without contention. Figure 1 depicts the operation. With this information each particle (processor) computes in parallel its relative position and velocity and in step 3 the collision time with its left neighbours. Step 4 is simplified by another invaluable operation, Figure 2. With this strategy and at the price of introducing some additional computation, the context, namely the set of active processors, is never reduced and the various cases need not be handled separately. The computing time for a single parallel update (steps 2-7 of the algorithm) of the 16384-particle system is 4.6 milliseconds on a 16K processor CM-2. To give an idea of the computing times required for a simulation run, two million iterations took about 3 hours on a 16K CM-2. This includes all phases: CM-2 time, front-end communication, data reduction and start-up.

The next section discusses step 7, the calculation of physical observables which lends itself to an interesting parallel approach.

**Sampling from the Dynamics**

Ultimately, the goal of a molecular dynamics simulation is to compute the macroscopic behaviour of the system. Indeed, apart from the collision prediction, most of the computer time is spent collecting the particle trajectories and then processing the information to determine the system properties. In the usual approach, quantities of interest are periodically written out to disk files to be post-processed. In general this is time consuming and thus
motivates a parallel approach, especially justified for high performance computers. Thinking Machines has attempted to solve the problem with a parallel disk array called the Data Vault which makes possible high speed two-way communication between the disks and the CM-2 processors [5].

In step 7 of the parallel algorithm, the phase space trajectories, Figure 3, are stored on the Data Vault at regular intervals for future use. In our typical runs, files of the size of about 10 MBytes are produced and efficiently stored during a simulation. However the problem remains of transferring even larger volumes of data and processing them on a remote workstation. Note that because of the large number of particles involved in our system, the mean intercollision time is very small and a faithful simulation of a macroscopic sample requires many more iterations than for a small system to obtain statistically significant quantities. Consequently, for a large system it is worth doing a maximum of data reduction and on the fly analysis during the simulation, that is, letting the CM do the work in parallel rather than leaving it to post-processing. Two examples, \( \sigma = 1.0 \) and \( \beta N/L = 0.5 \), are the calculation of the pressure through the Virial Theorem giving the correction to the perfect gas equation of state, Figure 4, and the one-particle density function, Figure 5.

To construct the one-particle density function we must decompose the domain \([0;L]\) into small segments and count how many particles have «fallen» into each of the segments at the end of the simulation. This information, suitably scaled, gives us a plot that approximates the density function. This operation can be done easily and efficiently on the Connection Machine. First, at each iteration step, each processor calculates in parallel the label of the segment visited by its attached particle. Then each sends a vote to the processor whose coordinate in the string is the same as the computed segment label. Figure 5 shows the resulting density of particles, independent of the temperature, near the left boundary. Note the characteristic «wall effect» [4].

**Conclusion**

We found that the CM programming model was easy to understand and use, although it took some time to learn C/Paris, the intermediate CM-2 language. This is because it is strongly tied to the detailed architecture of the machine, allowing closer control over geometry definition, memory allocation and interprocessor communications.
Nevertheless there are other programming environments, based on Fortran, Lisp and C that are easier to assimilate. Our problem was easy to map into the architecture and entailed a natural usage of the powerful CM-2 parallel primitive operations. On the other hand, although the printed documentation is good, there is a lack of easy-to-use debugging and parallel data visualizing tools. This poses a problem for the developer, especially when working from a remote site.

Large-scale molecular dynamics are a promising simulation tool in many situations, e.g. modelling of fluid flow at the atomistic level [7],[8]. Our hard-core fluid implementation on the CM-2 can be extended to two and three dimensions with a number of bodies on the order of tens of thousands, allowing the modelling of more realistic systems and the detection of complex flow patterns.

REFERENCES


APPLICATIONS OF SUPERComputERS IN THE DEVELOPMENT OF GYROTRONS

BY T.M. TRAN, M.Q. TRAN & R. GUBER, EPFL – CENTRE DE RECHERCHES EN PHYSIQUE DES PLASMAs – ASSOCIATION EURATOM-CONFEDÉRATION SUISSE

Le gyrotron est actuellement la source micro-onde la plus déveIlOpée utilisée pour le chauffage des plasmas de fusion par les ondes cyclotroniques électroniques. Les fréquences considérées varient de 100 à 300 GHz, à une puissance atteignant 1 megawatt ou plus. Le développement de ces sources nécessite des études d’analyse et de conception qui se basent essentiellement sur des codes de simulation utilisant des particules afin de modéliser d’une manière précise le transport et l’interaction du faisceau d’électrons dans les gyrotrons. Une présentation des modèles utilisés dans ces codes, qui exigent souvent d’importantes ressources informatiques, constitue le sujet principal de cet article.

The gyrotron is presently the most developed microwave source for the heating of fusion plasmas, using the electron cyclotron waves in the frequency range of 100–300 GHz at power levels exceeding one megawatt. The development of such devices necessitates analysis and design efforts which rely mostly on the simulation codes using the particles for the accurate modelling of the electron beam transport and interaction in gyrotrons. The presentation of the models employed in these codes, which often require large computer resources, will be the main subject of the present article.

INTRODUCTION

The physics modelling and the engineering design of high power electron tubes nowadays rely strongly on numerical computations; the cost of building and testing a tube limits the number of prototypes for a given development. Illustrative of this trend is the research and development programme undertaken by the Centre de Recherches en Physique des Plasmas in the gyrotron field, a high power (up to 1 MW per unit) electron tube operating in the centimetre to millimetre wavelength (frequency from 8 to 300 GHz).

DESCRIPTION OF THE GYROTRON

In a gyrotron the rotational energy of an ensemble of relativistic (energy < 100 keV) electrons spiralling in a magnetic field $B_0$, is converted into electromagnetic energy in an appropriate electrodynamic structure, such as a resonant cavity or even a simple waveguide. The electromagnetic (EM) wave frequency is determined by the relativistic electron cyclotron frequency \( \Omega_{ce} = \frac{eB_0}{\gamma m} \), where $\gamma$ is the relativistic factor \( \gamma = \left(1 - \frac{v^2}{c^2}\right)^{-1/2} \).

The main elements (Figure 1) in the gyrotron are:

- the electron gun, which creates the beam with a high content of rotational energy. The gun configuration which is widely used is known as the Magnetron Injection Gun (MIG). The beam parameters are typically: beam energy=80 keV, current=10-50 A, ratio $v_\perp/v_\parallel=$ 1.5-2, where $v_\perp$ and $v_\parallel$ refer to the velocity components in the direction perpendicular and parallel to $B_0$ respectively.

- the electrodynamic structure where the interaction between the EM wave and the beam occurs and from where the EM energy is extracted. For a gyrotron oscillator, it is either a cylindrical cavity operating in the $TE_{m,n,1}$ mode (m=azimuthal index, n=radial index) (Figure 1(a)), or a quasi-optical (Q.O.) Fabry-Pérot resonator (Figure 1(b)). In the first case, the EM wave is coupled out through one end of the cavity, whereas in the second case a diffraction grating could be used. Both the cylindrical cavity gyrotron and the Q.O. gyrotron are studied at the CRPP.

- the collector, where the energy of the spent electron beam is deposited or recovered (depressed collector).

- the microwave window, which separates the high vacuum (pressure < $10^{-8}$ mbar) region of the gyrotron from the atmosphere and through which the EM beam is coupled to the load.

The physics of the interaction between the EM wave and the electron beam can be understood by considering the...
variation of the electron momentum $p$ in the field of the wave. When the wave frequency is larger than $\Omega$, the majority of the electron loses energy to the wave. High speed computers are required to follow the non-linear stage of the evolution of the interaction of the beam with all the modes which can grow from the noise level in the resonant structure. Many complex phenomena must be included to correctly describe the physics of the electron beam from the cathode to the interaction region. The beam quality in phase space $(r, v_r, v_i)$ is affected by electric and magnetic fields either imposed externally or caused by the beam itself (AC and DC space charge effects, AC instabilities). A large effort has been undertaken to correctly model these phenomena. The main tool in this case is again numerical simulation.

The optimization of a gyrotron necessitates extensive calculations to determine:

- the electrodes of the MIG, with the constraints on high voltage stand-off (e.g., maximum DC electric field about 70 kV/cm in vacuo)
- the geometry of the resonant structure in order to maximize the conversion of the electron beam energy to the EM wave. The main constraint in this component is the power loss due to the skin effect of the metal which becomes particularly important at high frequency $f$ (the ohmic losses per unit area is proportional to $f^{5/2}$) and must not exceed 4 kW/cm$^2$.
- the collector and the magnetic field lines in this region. Again, the main constraint is the specific heat loss, which is 0.5 to 1 kW/cm$^2$ when a large area is to be cooled.
- the window structure, which needs to withstand the thermal stresses induced by the absorption of the microwave.

A set of computer programmes has been developed at the CRPP throughout the years to understand the physics and to address the technological issues. In this presentation, only the computer modelling of the electron gun and the physics of interaction will be given. Both models employ the particle simulation techniques to describe the transport of the electron beam as well as its interaction with the electromagnetic wave and therefore, require large computer resources.

**The electron gun (MIG)**

In the optimization of the electron gun, a 2d axisymmetric computational domain extending from the electron emitter to the electron collector has to be considered. Only the interactions of the particles with the static fields—the high frequency electromagnetic fields are neglected—are taken into account. These static fields include the external magnetic field used to guide the electron beam and the electrostatic field produced by the electrons. While the magnetostatic field can be readily calculated by the Biot-Savart equation, given a set of magnetic coils, the electrostatic field is determined by solving the Poisson equation with the boundary conditions specified by the complicated boundaries of the computational domain.

The DAPHNE gun simulation platform was developed, based on the model described above. It is embedded in the ASTRID programming environment [1]. Three modules of DAPHNE were written specially: the Biot-Savart program CMFI, the particle pusher PART and the electron beam collector module COLL. PART advances a given number of sampled electrons in the electromagnetic field to enable the computation of a precise charge density distribution, needed as the right hand side vector of the Poisson equation. The COLL module computes the time-dependent heat transfer. These modules communicate with each other and with the ASTRID modules through the data management system MEMCOM (see Figure 2). The inputs for CMFI and PART are given in the input datasets for the ASTRID module CASE. The input datasets for the ASTRID modules are interpreted by a specially conceived command language. The overall DAPHNE architecture is shown in Figure 2.

**Figure 2:** Software architecture of DAPHNE. The marks A denote modules coming from the ASTRID programming environment, the marks U denote user-defined modules.

The modules are executed in the standard UNIX way, the input datasets interpreted by the command language and data is read from and written back to the data base MEMCOM. The modules are executed in the following order:

- **MiniM** sets the subdomains defining the geometry of the gun
- **CASE** sets the Dirichlet boundary conditions for the gun
- **MESH** constructs an initial structured mesh in each subdomain of the gun
- **SOLVE** solves the Laplace equation to find the potential
- **MiniM** sets the magnetic field coils
- **CASE** sets the currents in the coils
- **CMFI** computes the magnetic field at each mesh point
- **PART** pushes the electrons and computes the charge density distribution
- **MESH** reconstructs the mesh using the charge density distribution as mesh density function
- **CMFI** recomputes the magnetic field at the new mesh points
At this point of the computation we fix the mesh and the magnetic field and iterate over the Poisson equation and the electron trajectories until convergence by executing:

SOLVE solves the Poisson equation to find the potential
PART pushes the electrons and computes the charge density distribution

After each of the steps described above, we can execute VIEW to study the results. If a steady state solution is reached, it is possible to compute the time-evolution of the collector temperature. The charge density and velocity distributions on the internal collector wall yield the energy deposition on the surface. The modules to be executed are:
CASE prescribes the time dependent beam characteristics and the cooling conditions
SOLVE solves the time dependent heat transfer equation
VIEW animates the temperature distribution as a function of time.

The DAPHNE program has been successfully benchmarked with the well-known EGUN code [2] as shown in Figure 3 which displays the evolution of the averaged electron velocity ratio produced by both codes. It should be noted that EGUN employs a regular square mesh over the whole 2d computational domain to discretize the Poisson equation which implies that large memory as well as CPU resources are needed for the whole gyrotron if we want to avoid loss of accuracy at certain regions of the gun. In DAPHNE, this problem is alleviated by decomposing the entire domain into several subdomains of different sizes and by the use of adaptive non-uniform meshes generated by the MESH module.

The physics of interaction

- Single mode interaction

The simplest description of the electron interaction with the high frequency field assumes a single cylindrical cavity $TE_{m,n,1}$ mode or a Q.O. Fabry-Pérot Gaussian $TEM_{m,n,1}$ mode. The frequency is fixed around the fundamental electron cyclotron frequency ($f = \frac{\Omega_e}{2\pi}$) or one of its harmonics ($f = n\Omega_e/2\pi$). Since only one wave frequency is considered, the motion of the electrons can be time-averaged around this frequency, thus simplifying greatly the electron equations. The initial conditions for the electrons, defined at the entrance of the interaction region, describe an annular beam with or without a spread of velocities. Usually when the electron beam is assumed to be cold (no spread of velocities) and of 'pencil' shape (a Dirac distribution on the transverse position), a sample of 16 particles is sufficient to represent the beam. Once the initial conditions for the electrons and the amplitude of the electromagnetic field are specified, the (averaged) equations of motion can be integrated across the interaction space, using for example a Runge-Kutta method. This model is straightforward and very fast, even when the annular shape of the beam and velocity spreads (requiring more than 100 sampled particles) are taken into account, especially for a vector computer since the vectorization can be done quite naturally. The main advantage of this model is to allow an extensive multi-dimensional parametric analysis of the gyrotron. A typical result of such an analysis is plotted in Figure 4 in which the gyrotron efficiency is shown versus the frequency and the electric field of the microwave. Using such a graph, one can estimate, for example, the optimum efficiency under the constraint of the maximum permissible electric field compatible with the specific heat load on the resonator wall.

In the model described above, the electromagnetic profile was assumed to be an eigenmode of the empty resonator. This approximation can fail when the quality factor $Q$ of the oscillator is low (especially for gyrotrons

![Figure 3](image_url): The electron beam velocity ratio versus the axial coordinate as obtained from the EGUN code and the DAPHNE code.

![Figure 4](image_url): The gyrotron efficiency versus the frequency and the wave electric field obtained by the single mode model for a Gaussian $TEM_{00}$ mode.
AppLtcnrrot's or SuprRcor,lpurERS rru tHr Drve LoprvENT oF GyRorRoNS using a cylindrical resonator) and becomes meaningless for a gyrotron amplifier or a backward wave oscillator. The next improvement to this model is thus to include a second order differential equation representing the wave profile variations along the interaction region, in the presence of the electron beam. The wave equation can be derived from the Maxwell equation and has boundary conditions at each end of the interaction region, reflecting the impedance matching conditions. The TWA code [3] was developed based on this model. It can be applied to gyrotron oscillators as well as gyrotron amplifiers. An example of output from TWA is shown in Figure 5.

**Figure 5:** The self-consistent field amplitude (dotted line . . .) and the electron efficiency (dashed line —) versus the beam propagation direction produced by TWA for a low Q cylindrical cavity gyrotron. The tapped cavity radius is also shown (continuous line) on the figure.

**Multimode interaction**

In the Q.O. gyrotron, the resonator is highly overmoded to accommodate the desired high microwave output power. As a result, the frequency spacing between two adjacent cavity modes can be small compared to the oscillation bandwidth of the gyrotron. In such circumstances, the competition between several longitudinal TEM_{0.0.9} modes could be severe (the higher transverse modes can be neglected because of their larger diffraction loss). In the simulation, the mode competition is followed in time, starting from a given wave spectrum of low amplitudes and random phases (to simulate the noise) until the nonlinear saturation state is reached [4]. In each time step, the electrons are pushed across the interaction region as in the single mode calculation, followed by the update of the power for each mode using the power balance between the particles and the waves. Typically 15 modes and 8000 sampled electrons are considered during several thousands of time steps, requiring about 2000 seconds of Cray2 CPU time. In the MULTIM code [4] developed at the CRPP, the competition of the fundamental (\( f = \Omega_{c}/2\pi \)) and the second harmonic (\( f = 2\Omega_{c}/2\pi \)) can also be simulated. In this kind of calculation, even more time steps have to be performed to attain the saturation. A comparison of the results produced by simulation and from experiment is shown in Figure 6, confirming that this type of simulation can be useful in predicting the gyrotron performances. It can serve to analyze different schemes to optimize the gyrotron efficiency, such as the one that uses a beam voltage ramp up in time as shown in Figure 7. The CPU cost, however, prevents the performance of very extensive parametric scans which can be done comfortably only with the single mode model.

**Figure 6:** The Q.O. gyrotron output power as a function of the electron beam current: comparison of the experimental results with the simulation results obtained from MULTIM with and without beam space charge effects.

**Figure 7:** Simulation of the time evolution of the Q.O. gyrotron output power when the beam voltage is suddenly increased at \( t = 2.2 \mu s \) from \( V_b = 70 \) kV to \( V_b + \Delta V_b \), showing that an 76% increase of the saturated power can be realized with a beam voltage ramp up scheme.
In the models considered so far, we have always assumed a given wave frequency spectrum chosen around the electron cyclotron frequency (or one of its harmonics) so that it is possible to perform a time-average of the electron motion. Moreover, the collective electrostatic interactions were either neglected or treated using more or less simplified models. In order to analyze more accurately the physics of the interaction, especially the effects of the electron beam collective oscillations on the gyrotron emission characteristics, we are developing 1d and 2d codes which take the rapid electron gyration into account (no time-averaging) and use the full Maxwell equations to compute the electromagnetic fields. This type of particle simulation, also called the Particles In Cell method (PIC) has been exploited successfully in Plasma Physics [5]. These codes are mainly aimed to study the collective effects in very high power microwave sources rather than to the design of existing gyrotrons.

**REFERENCES**


---

**PERFORMANCES OF A MULTIPROCESSOR MULTIWINDOW VISUALIZATION SUBSYSTEM**

*R.D. Hersch, B. Tonelli, B. Krummenacher — Laboratoire de Systemes Périphériques, EPFL*

Actuellement, les systèmes multiprocesseurs à vocation générale ne disposent pas de capacités d'affichage d'images couleur à grand débit. Dans le cadre d'un projet de recherche sur la parallélisation des systèmes d'entrée-sortie, un système d'affichage multiprocesseur-multifenêtre pour la visualisation rapide d'images couleur de 24 bits/pixel a été développé. Ce dispositif d'affichage peut être raccordé à un tableau de processeurs de traitement ou à des réseaux à large bande (FDDI, RNIS large bande). Grâce à son architecture à base de transputers comportant 3 processeurs et 12 liens de communication, ce système reçoit, découpe et affiche des images couleur de 24 bits/pixel à une vitesse de 18 MOctets/sec. Dans des conditions optimales, six images couleur de 1024x1024 pixels ou 24 images de 512x512 pixels sont affichées par seconde. Un tel dispositif de visualisation est 15 à 20 fois plus performant que le système de fenêtrage X-Window s'exécutant sur des stations de travail de type Sun SparcStation. Il trouve son application comme serveur de visualisation pour la gestion et l'affichage rapide d'images couleur (images produites par supercalculateur, cartographie, génie civil, biologie, médecine, etc.).

Multiprocessor systems are available at moderate costs, but they generally lack high-speed multiwindow colour image visualization capabilities. A multiprocessor multiwindow parallel visualization subsystem is presented which can be hooked onto parallel processor arrays or high-speed network interfaces (FDDI, ATM broadband). Thanks to its well-balanced triple transputer architecture, the multiwindow display subsystem is capable of simultaneous transfer, clipping and display of 24 bits/pixel colour images at a rate of 18 MBytes per second. Under optimal conditions, it is able to receive, clip and display six full screen 1024x1024 or twenty-four 512x512 full colour images per second. Such a display subsystem is an order of magnitude faster than X-window systems running on standard workstations. It satisfies the needs for browsing through image data bases and for fast visualization of uncompressed image sequences (scientific data produced by supercomputers, photogrammetry, civil engineering, biology, medicine, etc.).
PERFORMANCES OF A MULTIPROCESSOR MULTIWINDOW VISUALIZATION SUBSYSTEM

INTRODUCTION

Thanks to their special-purpose display pipeline, current state-of-the-art workstations are able to display synthesized colour images in real-time. However, they prove to be slow when executing general purpose 2-D image processing and visualization tasks which are necessary for visualizing existing images, like aerial photographs or medical images [1].

In the near future, high capacity image servers will be available for storing large quantities of pixel image data. These servers will be accessible through high-speed local area networks or through future broadband ISDN public networks. Browsing at high speed through random sequences of images will become as common as today’s fast visualization of text by high-performance editors [2]. Simulation programs running on supercomputers will generate image sequences for animated interactive user-driven visualization.

ARCHITECTURE

In order to offer high-speed image visualization interfaced with various devices such as processor arrays, high-speed local area (FDDI) and broadband communication networks, a multiprocessor transputer-based display subsystem was designed for fast interactive multi-window visualization of full colour 24 bits/pixel images (Figure 1).

This display subsystem consists of one separate processing unit for each of the basic Red, Green and Blue colours (Figure 2). Each processing unit consists of a T800 transputer with 4 MBytes RAM and 2 MBytes of video-ram. The processor also incorporates 4 serial links each offering an effective communication bandwidth of 1.5 MBytes/s.

The display subsystem incorporates a window manager responsible for the dialog with the application, for filtering input events and for the creation and maintenance of the window hierarchy. Window server processes running on the Red, Green and Blue processors are responsible for data exchange between the application and the display (Figure 3).
They receive colour image data in parallel from application processes which they clip and display in the target window. In the case of an event which changes the current state of window overlays, the window manager process updates the information record of each window specifying its visible rectangular parts and sends messages to the concerned application processes, asking them to regenerate their visible window image parts.

**Performances**

Global display subsystem performance depends on communication bandwidth, on the extent to which communication and display are executed in parallel and on the performance of the display processors during clipping and display operations.
Communication bandwidth is mainly limited by the bandwidth of the transputer serial communication links. But, since link controllers work by direct memory access (DMA), 4 transputer links can simultaneously transfer information into memory without requiring processor power. Communication and processing can therefore be overlapped. Figure 4 also shows that the message passing system responsible for transferring messages between any pair of processors does not significantly reduce communication bandwidth (less than 7%).

The clipping time necessary for displaying an image within a window depends on the complexity of overlapped window parts. It is proportional to the number of rectangles lying within the boundary of a Manhattan polygon representing the visible image part. Clipping time was measured to be 100 microseconds for a non-overlapped window and 300 microseconds for a partially covered window, resulting in a polygonal uncovered image part made up of 7 rectangles. Compared to image display time (1 to 100 ms, depending on image size), clipping time is negligible.

Display time includes clipping time and the time needed to transfer clipped rectangular image parts from the reception buffer into the corresponding parts of the image frame buffer. Measured maximal image display bandwidth by one T800 transputer is 9.5 MBytes/s (Figure 5). This relatively high speed is obtained thanks to the move pixel block instruction available in the transputer instruction set.

Display performances of the three-processor transputer-based display sub-system are compared with those of standard UNIX workstations running on Sun workstations (Figure 6). The parallel display sub-system visualizes colour images 10 to 15 times faster than UNIX workstations running on standard workstations.

Global performance tests including parallel communication, clipping and display operations show that the display sub-system is capable of displaying 24-bit image data at a rate of 18 MBytes/s (Figure 7). Communication is running in complete overlap with clipping and display. The display sub-system is extremely well balanced: at full speed, communication bandwidth per processor is 6.3 MBytes/s. With non-overlapping full image size windows, each display processor accesses main memory (read and write accesses) at a rate of 12.6 MBytes/s. The resulting total memory bandwidth of 20 MBytes/s per processor represents the maximum available bandwidth for a 32 bits wide memory bank having a cycle time of 200ns.

**Conclusions**

The display sub-system is driven by 3 transputers and offers the external processor array a 18 MBytes/s communication bandwidth through its 12 communication links. Display resolution is 1024x1024 pixels at 24 bits/pixel. Since transputers allow simultaneous communication and processing, previously received colour images are clipped and displayed while new images are being received. Extensive performance tests show that the display is able to receive and visualize colour image data at a rate of 18 MBytes per second. In an optimal configuration, one can either display six full screen 1024x1024 or twenty-four 512x512 24 bits/pixel images per second. Such speed is sufficient for real-time interactive visualization and manipulation of image sequences originating from a processor array, from fast network accesses (FDDI, ATM broadband) or from a supercomputer.

Communication, display and memory bandwidths are well balanced; at full speed, both the memory and the communication bandwidth represent the bottleneck of the system. The proposed multiprocessor multiwindow display sub-system is one order of magnitude faster than standard workstation-based multiwindow visualization systems. It satisfies the needs of fast image visualization (photogrammetry, civil engineering, biology, medicine, etc.). Such a display sub-system, hooked onto a high-speed network, offers high-performance image visualization at moderate costs.

**References**


The direct SCF algorithm to the solution of the Hartree-Fock equations, a widely used method for the calculation of molecular and electronic structures, has been parallelized such that the computation can be split over a number of machines. Since the communication and synchronization requirements are modest, computers that are geographically remote, but that can be accessed via a communication service like Internet, can be networked. Each server uses all of its processors in parallel to compute the tasks distributed by the client-computer, a mode of operation best described as distributed-concurrent parallel computing.

INTRODUCTION

Combining or networking two or more supercomputers to work on the same job is an obvious way to address the need for extreme computational performance. This allows us to carry out calculations too large to be executed on any single computer even if all processors of that machine were used. But it is also an attractive way to reduce turnaround times for very large applications in situations where more than one machine is available.

We have developed software that permits the distribution of a calculation over several machines (Sciddle). For an application like the Hartree-Fock calculation, as implemented in the program DISCO, the communication requirements are small, and it is possible to network machines which are not on the same floor-space, but which are connected by a public network like the EPFL Cray-2 and the ETHZ Cray Y-MP. Using this cluster, it is possible to obtain the response of up to eight (dedicated) Cray-2 processors on weekends or other periods of moderate user activity. The program described is now being used in applications that focus on properties of very large molecules like the one shown in Figure 1.

Figure 1: The accurate computation (Hartree-Fock and MP2 theory) of structural and electronic properties of a molecule of this size requires a massive computational resource, obtainable by networking multiprocessor supercomputers.
The servers perform their computation in shared memory parallel mode, thus using all of their processors as described in [1]. All data, including the input information are broadcast by the client. This makes the servers input/output-free and computation intensive. Therefore the server program only consists of the code necessary to compute a package of tasks, plus a few routines that handle the communication with the client.

In a network of computers that show different levels of response, the task-packages are issued in a round-robin fashion: once a server has completed a package, it signals its availability for another package to the client. If no more packages are available, the client will collect the partial result from that server. The machine that shows the best response (speed and availability of the CPUs, communication speed) will therefore execute the highest number of packages. The packages have to be built and distributed such that the server idle times are minimal, and also such that the individual processor idle times within a server are minimal (two-level load-balancing problem).

**CLIENT-SERVER COMMUNICATION**

To communicate data or messages between client and server(s), we have developed a small communication language and library, the «Sciddle system» [5], which is written in portable C language and which is based on Berkeley BSD sockets. The client process establishes a socket connection to all servers, and, after having initiated the computation, all data along with the list of tasks are broadcast through this channel. Server action is initiated by the client through a form of remote procedure call (RPC) which has been implemented such that it is non-blocking (asynchronous).

The Sciddle system provides a communication environment such that the programmer can call a server as if it were a regular subroutine or function call. Sciddle will create an interface between the client and a server based on user-defined procedure declarations. For example, the client/server interface procedure declaration for the distributed computation of the dot product of vectors a and b in the «Sciddle» language would look like:

![Figure 2: Sites connected in the various networks established with the client process hosted by one of the Minnesota machines.](image-url)
The Sciddle object code, together with the Sciddle library items, will be linked to the user program. The transfer of the data (density matrix (input), Fock matrix (output)) in the current version of DISCO ranges between 100,000 and 1,000,000 words. The contents of the shared variables needed by the servers to execute their code is less than 10 KWords. Only 16 words of information have to be transferred with each task-package. With sustained communication rates of 100 KBytes/sec between Lausanne and Zürich, 30 KBytes/sec over the Atlantic or 3 KBytes/sec over the Pacific, multi-continent networks of supercomputers can be set up (Figure 2). The machines involved in this project are listed in Table 1.

Table 1: From Lausanne to Seoul: list of the supercomputers networked and their locations. (MCC=Minniseota Supercomputer Center, NCSA=National Center for Supercomputer Applications in Urbana-Champaign, SDSC=San Diego Supercomputer Center, SERI=Systems Engineering Research Institute, Korea Institute of Science and Technology, Daeduk Science Town and Seoul).

<table>
<thead>
<tr>
<th>Machine</th>
<th>Label</th>
<th>Site</th>
<th>Processors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cray-2/4-226</td>
<td>c2</td>
<td>EPF Lausanne</td>
<td>4</td>
</tr>
<tr>
<td>Cray Y-MP/264</td>
<td>cy</td>
<td>ETH Zürich</td>
<td>2</td>
</tr>
<tr>
<td>Cray-2/4-512</td>
<td>sc</td>
<td>MSC (Minnesota)</td>
<td>4</td>
</tr>
<tr>
<td>Cray X-MP/464</td>
<td>sf</td>
<td>MSC (Minnesota)</td>
<td>4</td>
</tr>
<tr>
<td>Cray Y-MP/464</td>
<td>uy</td>
<td>NCSA (Illinois)</td>
<td>4</td>
</tr>
<tr>
<td>Cray-2/4-128</td>
<td>u2</td>
<td>NCSA (Illinois)</td>
<td>4</td>
</tr>
<tr>
<td>Cray Y-MP/864</td>
<td>yl</td>
<td>SDSC (California)</td>
<td>8</td>
</tr>
<tr>
<td>Cray-2/2-128</td>
<td>sc</td>
<td>SRI (Korea)</td>
<td>4</td>
</tr>
</tbody>
</table>

Results

On a network of five Cray Y-MP computers (dedicated mode) with a total of twenty processors, for an application similar to the one in Figure 1, a speedup factor of 16.6 was measured. This speedup reflects a degree of parallelization of close to 99%, and a performance of over 3.3 GFLOPS. The most important results of these measurements taken with Dr. Phuong A. Vu of the Software Division of Cray Research Inc., using machines of their Eagan (Minnesota) computer facility, are listed in Table 2.

Table 2: Results of the 20 processor (5 server) network obtained for a direct SCF Hartree-Fock calculation on an organic molecule of interest in materials science (Prof. U.W. Suter, ETH Zürich). The data refer to one single direct SCF iteration.

<table>
<thead>
<tr>
<th>Processor</th>
<th>Time (sec)</th>
<th>Speedup</th>
<th>MFLOPS</th>
<th>Parallelism</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Processor</td>
<td>2,144</td>
<td>1.00</td>
<td>199</td>
<td></td>
</tr>
<tr>
<td>20 processors</td>
<td>178</td>
<td>16.61</td>
<td>3,320</td>
<td>98.9% (Amdahl’s law)</td>
</tr>
</tbody>
</table>

Application:

Direct SCF Hartree-Fock calculation on Bis-(2,6-dimethylphenyl)carbonate (38 atoms, no symmetry, 314 contracted basis functions)

Network:

One Y-MP/8, one Y-MP/6, three Y-MP2E; 20 processors, Hyperchannel

THE «SWISS NETWORK»

For the same example, using the 6 processor cluster formed by the EPFL Cray-2 and the ETHZ Cray Y-MP, elapsed times between 666 and 890 seconds have been observed during normal (non-dedicated) service hours on the weekend of September 1, 1991. The fastest time measured, 666 seconds, corresponds to a performance of 640 MFLOPS. During the time of the measurement, one job on the Cray Y-MP, and three to four other jobs on the Cray-2 were competing for the CPUs (all foreign jobs were serial (single CPU) jobs). The theoretical peak performance for this example is slightly more than 800 MFLOPS. (DISCO is nearly twice as fast on a cy processor than on a c2 processor).
The «Trans Atlantic Network»

The above example only experienced a marginal speedup when the four processor Cray-2 (sc) of the University of Minnesota was added to the «Swiss Network». The best elapsed times recorded were 650 and 654 seconds. At the time of the computation, the sc machine was still under a significant user-load, and showed a poor CPU availability (0.92 CPUs; 23% of machine) compared to the Swiss machines (2.28 CPUs or 57% (c2); 1.96 CPUs or 98% (cy)). The sf machine was not available (operational reasons) at the time of the measurement. Depending on the times of day (e.g., daytime in Switzerland, nighttime in Minnesota), considerable speedup can be observed for this cluster.

The biggest network set up so far has been a cluster of 6 servers based on two continents with a total of 28 processors. In this configuration all machines of Table 1 except the ETH Zürich (remote access restrictions) and the SERI machines (communication rates) were involved. In this network one transatlantic Internet socket connection (sc to c2), three USA continental Internet connections (sc to u2, uy, and yl), plus an Ethernet connection (sc to sf) were established.

Even though this is the biggest cluster assembled so far, the best performances (speedups) have been obtained by networking the MSC with the NCSA Crays (4 machines, 16 processors). With a relatively small example, the pyridine molecule (C₆H₄N₃; 451 seconds CPU time on a Y-MP), the best performance recorded so far was 91 seconds wall-clock time, thus corresponding to the response of 4.96 Y-MP processors. As DISCO, depending on the example, delivers between 150 and 225 MFLOPS, this corresponds to a sustained 1 GFLOPS performance. At the time of the measurement, 5:30 a.m. CDT on Monday, August 19, the machines had nearly worked off their user-load and DISCO got 2.81 CPUs from sc (70% of machine), 3.33 CPUs from sf (83%), 2.12 CPUs from u2 (53%), and 2.14 CPUs from uy (54%).

The SDSC Y-MP/8 (yl) is very often the machine with the heaviest user-load, and its inclusion in our networks rarely showed significant speedups. The SERI Cray-2 (sc) is the machine with the narrowest communications bandwidth, and it will be possible to operate it as a server once the client/server communication has been fully optimized.

Outlook

Reducing the turnaround time for large calculations coupled with more optimal machine utilisation (in DISCO/2, using all CPUs in parallel is more cost-effective than executing in single-processor mode due to better memory utilisation) is an undisputable advantage. Speeds of computation in the neighbourhood of 0.5 GFLOPS during regular service hours (non-dedicated mode) on the «Swiss cluster», for example, would allow computations which would be unreasonable, if not impossible, to run in single-processor mode.

Many centres operate more than one high-performance computer (e.g., see Table 1). This makes it very attractive to network these machines, locally or between centres. The main difficulty in networking machines based at different centres is not a technical issue, but the fact that these machines may be operated and administrated in incompatible ways. For large jobs the servers will have to execute in batch mode. This, however, requires that all servers execute within a relatively small bracket of time to avoid long wait-for-connection times.

In the DISCO program development, a number of issues, such as fault-tolerance, load-balancing, and communication will have to be addressed to make the program more efficient and easier to use. The program is being ported to networks of (multiprocessor-) workstations as well as to massively parallel machines of comparable architecture (distributed memory, MIMD).

Acknowledgements

This work has involved the following computer centres: SIC EPF Lausanne, RZ ETH Zürich, MSC in Minneapolis (Minnesota), NCSA in Urbana-Champaign (Illinois), SDSC in San Diego (California), SERI in Seoul (Korea), and the Cray Research computer facility (CCN) in Eagan (Minnesota). Generous grants of computer time, as well as the excellent technical support from personnel at these centres have been instrumental for this project. This project has been supported by the Swiss Science Foundation (SNF) and by Cray Research (Switzerland).

References

COMPUTATIONAL FLUID DYNAMICS ON MASSIVELY-PARALLEL SIMD COMPUTERS

BY M.L. SAWLEY — IMHEF, ECOLE POLYTECHNIQUE FEDERALE DE LAUSANNE
& C.M. BERGMAN — C2M2, ROYAL INSTITUTE OF TECHNOLOGY (KTH), STOCKHOLM

INTRODUCTION

The major advance in supercomputing technology during the last decade has made possible the solving of very large computational problems that previously could not have been considered. Traditional supercomputers are characterized today by very short clock cycles (down to 2.9 ns), very large main memory (up to 4 GBytes) and adapted memory bus bandwidth. Efficient compilers that run codes simultaneously on several vector processors provide the possibility of solving real problems at GFlop speeds. Despite these achievements, it is recognized that traditional supercomputers are reaching a physical performance barrier in communication time, set by the speed of light. It is for this reason that there has recently been a large interest in the development of massively-parallel computers, which rely on a large number (> 1000) of processors of modest power, each with its own local memory, to obtain a large overall performance [1].

Parallel computing, however, requires not only adapted computer architecture, but also the development of software to enable the efficient simultaneous use of all the processors. Traditional supercomputers with a low level of parallelism (4 processors in the case of the Cray-2) employ a control-parallelism approach. In this approach, the computational problem at hand is subdivided into a number of more-or-less independent tasks, with different processors performing the different tasks in parallel. A second approach, which relies on each processor performing the same instruction on different data, is called data parallelism. The data-parallelism approach is employed by massively-parallel computers of SIMD (Single Instruction, Multiple Datastream) type.

Computational Fluid Dynamics (CFD), which deals with the numerical simulation of fluid flows, has become increasingly important over recent years due to its immediate application to a large number of engineering problems. Most applications of CFD are based on the resolution of a set of partial differential equations (e.g., the Navier-Stokes equations) that describe the continuum behaviour of the fluid. Large-scale computations require the treatment of the sizeable quantity of data necessary to describe the flowfield. Such computations are computer-resource intensive, and therefore are strongly dependent on ongoing progress in supercomputing technology. While traditional supercomputers are presently used to solve numerous simplified problems, there exists a wide range of complex engineering problems that cannot be resolved.

Since the field of Computational Fluid Dynamics is computer-resource intensive, it is not surprising that there exists a growing interest in the use of massively-parallel computers to solve fluid flow problems. We present here the results of an investigation into the use of two different massively-parallel computers of SIMD type: the Thinking Machines CM-2 and the MasPar MP-1. A code to calculate inviscid flow, originally written in Fortran 77 for a traditional vector supercomputer, has been adapted to take advantage of the data-parallelism concept employed by SIMD computers. The code has been rewritten entirely in standard Fortran 90, for which suitable compilers are available for both the computers considered. It is shown that the discretization of the governing equations on a regular mesh is well adapted to data parallelism. For a typical test problem of supersonic flow through a ramped duct, computational speeds similar to those obtained using a traditional supercomputer have been achieved using these massively-parallel computers.

La simulation numérique des écoulements (CFD) est consommative de beaucoup de ressources de calcul, ce qui explique l'intérêt croissant pour les ordinateurs massivement parallèles dans la résolution de problèmes de Mécanique des Fluides. Nous présentons ici les résultats de deux ordinateurs massivement parallèles de type SIMD : le CM-2 de Thinking Machines et le MP-1 de MasPar. Nous avons pris un code initialement écrit en Fortran 77 pour un ordinateur vectoriel traditionnel, et l'avons adapté de façon à tirer profit du concept de parallélisation des données des machines SIMD. Le code a été réécrit en Fortran 90, langage pour lequel des compilateurs convenables existent sur ces deux machines. Il est démontré que la discrétisation des équations qui gouvernent le comportement du fluide sur un maillage régulier est bien adaptée à la parallélisation des données. Dans le cas-test typique d'un écoulement supersonique dans un conduit, des vitesses d'exécution comparables à celles obtenues sur un superordinateur traditionnel ont été atteintes sur ces machines massivement parallèles.
THE MASSIVELY-PARALLEL COMPUTERS EMPLOYED

In this article, we report on an investigation into the resolution of a CFD problem using two different massively-parallel computers of SIMD type: the Thinking Machines CM-2 computer at the Royal Institute of Technology (KTH), Stockholm; and the MasPar MP-1 computer presently on loan at the EPFL from MasPar Distributor, Zürich. Although the basic approach employed for the use of these two computers is similar, it is of interest to compare some of the technical details of the two machines. These can be a governing factor in the computational efficiency of the machines. The following table shows the values for the particular models employed, together with those of the fully configured machines:

<table>
<thead>
<tr>
<th></th>
<th>CM-2 (at KTH)</th>
<th>MP-1 (at EPFL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of processors</td>
<td>8,192 (max. 65,536)</td>
<td>16,384 (max. 4,096)</td>
</tr>
<tr>
<td>Bit size</td>
<td>32</td>
<td>4</td>
</tr>
<tr>
<td>Local memory per proc.</td>
<td>32 KBytes (max. 128)</td>
<td>16 KBytes (max. 64)</td>
</tr>
<tr>
<td>Total memory</td>
<td>256 MBytes (max. 8 GBytes)</td>
<td>64 MBytes (max. 1 GBytes)</td>
</tr>
<tr>
<td>Neighbor communication</td>
<td>hypercube - NEPS</td>
<td>2D eight-way X-Net</td>
</tr>
<tr>
<td>Global communication</td>
<td>hypercube - router</td>
<td>multi-stage crossbar router</td>
</tr>
<tr>
<td>Floating point operations</td>
<td>on proc. chip</td>
<td>inherent on proc. chip</td>
</tr>
<tr>
<td>MIPS (peak)</td>
<td>500 (max. 2,500)</td>
<td>4,000 (max. 26,000)</td>
</tr>
<tr>
<td>MHzp (32-bit peak)</td>
<td>1,750 (max. 6,000)</td>
<td>325 (max. 1,800)</td>
</tr>
<tr>
<td>Low level language</td>
<td>Paris</td>
<td>MPI</td>
</tr>
<tr>
<td>High level language</td>
<td>Fortran, C*, 'Lisp</td>
<td>Fortran, (MPI)</td>
</tr>
</tbody>
</table>

Both of these computers make use of a UNIX front end (SUN or VAX), which provides a very convenient interface between the user and the parallel processor array. Standard networking protocols, such as TCP/IP, NFS, and X Window, are thus available. In addition, the MP-1 computer offers a programming environment, based on the X Window standard, within which users can execute, debug, analyze and optimize programs. This feature proved to be invaluable in the present study. (While a similar programming environment is not available on the CM-2 at KTH, it is provided on the more recent CM-200 series.)

It is important to note that the vast majority of research and industrial CFD codes have been written in Fortran 77, and there is a strong reluctance throughout the CFD community to invest the necessary effort to learn other programming languages. Both the CM-2 and MP-1 computers, however, make available Fortran compilers with array extensions based on the recently-accepted Fortran 90 standard. While not all of the Fortran 90 standard has been incorporated into the present compilers, these compilers are compatible in most respects: the identical Fortran code can therefore run on both machines.

The goal of the present study was to investigate the utilization and efficiency of massively-parallel computers to solve a typical CFD problem coded in Fortran. All of the results presented here were obtained using codes written entirely in Fortran.

THE flow PROBLEM AND ITS NUMERICAL TREATMENT

A particular area of fluid dynamics that utilizes the full extent of computational resources currently available is the field of aeronautics. For a number of years, the Institut de machines hydrauliques et de mécanique des fluides (IMHEF) within the EPFL has been collaborating with similar institutes throughout Europe on the numerical investigation of very high velocity (hypersonic) flows. This work, undertaken within the framework of the HERMES research programme, has enabled the development of computer codes to model hypersonic flow around re-entry bodies (see, for example, [2] - [4]).

For the work reported here, a relatively simple code developed by one of the authors [5] has been used as a testbed to study the implementation of the data-parallel approach suitable for use on the machines described in the previous section. This code solves the well-known Euler equations for two-dimensional, inviscid, compressible flows. The required steady-state solution is obtained as the limit of the time-dependent problem. The Euler equations are discretized in space on a regular (structured) mesh using a finite volume method with a central difference scheme. Numerically-induced oscillations near discontinuities (shock waves) are damped by the addition of a second-order artificial dissipation term. A fourth-order dissipation term is included to damp odd/even oscillations permitted by the numerical scheme. The resulting system of discretized equations is integrated in time using an explicit fifth-order Runge-Kutta scheme.

The original code was written in Fortran 77 for use on a traditional supercomputer, and thus has been optimized for the vector processors employed. This provides, in principle, the possibility of comparing the performance of vector and parallel computers to resolve the chosen problem. An increased performance can be obtained on a multi-processor vector computer, such as the Cray-2, by making use of the multi-block structure (based on domain decomposition) inherent in the original code. All of the computations performed for the present study, however, employed meshes consisting of only a single block.

The complete code has been re-written in Fortran 90 for use on the CM-2 and MP-1 computers. Fortran 90 regards arrays as first-class language elements, and as a result many elemental operators can be used on entire arrays. In addition, this language contains a number of array functions and intrinsics. As an example, a comparison of Fortran 77 and Fortran 90 versions of a simple code fragment is given below.
In most cases, the use of Fortran 90 results in simpler, more readable code compared to the equivalent Fortran 77 code.

To take advantage of the data-parallel approach offered by the CM-2 and MP-1 computers, the inherent parallelism of the problem must be exploited. This is achieved very simply for the present problem by assigning one node point of the computational mesh to one processor. Since the number of mesh points is generally greater than the number of processors, the concept of virtual processors is employed. Each processor in this case undertakes the required operations for more than one mesh point, with the VP ratio defined as the ratio of the number of virtual to physical processors. In fact, the mapping of the computational mesh onto the (virtual) processors is completely transparent to the Fortran programmer, as the necessary manipulations are undertaken by the compiler. The same code can therefore be run on machines with different numbers of processors, and means that no re-coding is required in order to use different machines, or in the event of an upgrade of the number of processors.

**Results**

As an example of the utilization of the above-described methods, we present here results obtained for Mach 2 flow of air through a two-dimensional ramped duct. Such a problem can be seen as a simplified model for flow through the inlet duct of a supersonic aircraft. It also provides a

---

**Figure 1:** Contour plots of (a) pressure, (b) Mach number, and (c) Schlieren image, computed for Mach 2 flow through a ramped duct. The flow is from left to right, with high values represented in red and low values in blue.
useful test problem for the validation of a numerical code, as the physical situation involves a complex series of shock wave reflections from the duct walls.

Calculations have been performed using various meshes with different mesh spacing (and hence total number of mesh points). As the number of mesh points is increased, the resolution of the shock waves improves and the influence of the added numerical dissipation decreases. Contour plots of the computed local Mach number and pressure, as well as a computed Schlieren image (with values depending on the local density gradient) are presented in Figure 1. These results were obtained using the largest computational mesh employed, having 512 points in the streamwise direction and 128 points in the crosswise direction. The VP ratio for this mesh is 8 for the CM-2 and 16 for the MP-1. From Figure 1, it can be observed that a good shock wave definition is obtained.

These computations have been performed both on the CM-2 and MP-1 computers (using the Fortran 90 code), as well as on the Cray-2 computer at the EPFL (using the original Fortran 77 code). A comparison of the performance of these computers must be treated with a deal of caution, as a number of different measures of performance can be employed. In addition, there exists certain subtleties in the calculation of the number of floating point operations and the computational time used. However, in general, it has been noticed that an increase in the problem size (due to a more refined computational mesh) results in an increase in the performance of the massively-parallel computers. This is in part due to the fact that such a size increase causes a greater load increase of the parallel-intensive sections of the problem compared to the imposing of the boundary conditions at the edges of the computational domain. In addition, an increase in the problem size reduces the relative importance of inter-processor communications. Such a dependence on problem size was not observed for computations performed on the Cray-2 since, for the mesh sizes employed in the present study, the vector lengths are sufficiently long. To solve the problem using the largest mesh (512x128), the CM-2 and MP-1 required a similar computational time (approximately 30 CPU minutes to obtain the converged solution), this being roughly twice that for the Cray-2. Work is continuing in order to both improve and quantify the performances of the code on the different machines.

CONCLUSIONS

The present study has demonstrated that using two currently-available massively-parallel SIMD computers, typical CFD problems can be solved in an efficient manner. The availability of suitable compilers for Fortran 90 code has been found to make possible the simple conversion of an existing Fortran 77 code. The use of the debugging and analysis tools offered by the programming environment has been found to enhance substantially the utilization of the parallel computers considered.

A comparison of the computational efficiencies of the machines used in the present study can be regarded as a test, not only of the massively-parallel computers, but also of their compilers. In this respect, it should be noted that the present Fortran compilers of the parallel machines employed should be considered as initial (at least compared to the refined Cray CF77 compiler). Further improvements in compiler efficiency are thus to be expected, leading to associated improvements in the efficiency of the computations performed.

Finally, it should be noted that the flow problem examined in the present study was rather simple, being of two-dimensional nature and solved by means of an explicit scheme using a structured mesh. The extension to a three-dimensional problem should be straightforward, and should in fact result in an enhancement of the efficiency of the parallel computations. The modification of the numerical scheme to use an implicit scheme and/or an unstructured mesh poses certain difficulties in the application of the data-parallel approach. The application of such schemes to the resolution of CFD problems using massively-parallel SIMD computers is currently being investigated at both C2M2 and IMHEF.

ACKNOWLEDGEMENTS

The authors wish to thank Per Vahlund for his contribution to the work on the CM-2, and to MasPar Distributor A.G., Zürich for the loan of the MP-1 that made the present study possible.

REFERENCES

A worldwide effort is being made to develop the tokamak concept of magnetic confinement into the basis of a viable fusion reactor and this requires the resolution of various technical and physical problems the most fundamental of which is the question of stability. In particular the operational space of the tokamak is limited by various large scale instabilities which can allow the plasma to escape confinement. The most dramatic and least understood of these is the major disruption. Major disruptions can lead to a violent termination of the plasma discharge and in a large tokamak such as JET (Joint European Torus) mega-ampere currents are quenched in a matter of milliseconds producing forces up to 1000 tons on the wall and coils while intense heat fluxes can also produce severe erosion of the vessel wall. Since only a limited number of hard disruptions can be tolerated during the life-time of a large tokamak, it is important to understand the mechanisms by which disruptions occur. Despite detailed experimental studies, the theoretical understanding has remained poor because of the complex nature of the instability. At CRPP a study has been made in collaboration with JET to try to reproduce the complex sequence of events by a resistive, nonlinear, three dimensional, magnetohydrodynamic (MHD) simulation. The project has involved running cpu intensive computations on the Cray-2 and these have resulted in the first simulation study to reproduce the entire disruption sequence with detailed agreement with experiment.

**Background**

Tokamaks confine a hot plasma using a system of poloidal and toroidal magnetic fields that are generated by external current coils and an induced plasma current, see Figure 1. Of all the magnetic confinement configurations the tokamak is so far the most successful with a performance closest to achieving the high temperatures and densities needed to produce a net energy output from the thermonuclear fusion of light nuclei such as deuterium and tritium. Among the limiting instabilities, the most violent, arising on microsecond timescales, are those described. 

![Figure 1: Tokamak magnetic field configuration](image_url)
by the ideal magnetohydrodynamic (MHD) model which treats the plasma as a perfectly conducting fluid. Such instabilities cannot be tolerated during a plasma discharge and so ideal MHD provides the most important stability criterion for determining the operational space of a magnetically confined plasma.

Ideal MHD stability calculations form the major part of the theoretical effort at CRPP\[2\]. In particular, pioneering computations made with the 2-D stability code ERATO\[3\] have lead to the prediction of a fundamental $\beta$-limit for tokamak equilibria ($\beta = \text{plasma pressure/magnetic pressure}$) and has also provided essential information for the design of new tokamaks like TCV (Tokamak à Configuration Variable, under construction in Lausanne), NET (Next European Torus) and ITER (International Thermonuclear Experimental Reactor). A 3-D code called Terpsichore \[2\], has also been developed to investigate the ideal stability of configurations with 3-D equilibria such as the Wendelstein VII-AS stellarator at Garching.

Good confinement is achieved in a tokamak when the magnetic field lines lie on nested toroidal surfaces as shown Figure 2(a). Ideal MHD stability is favoured by arranging the field lines to wind helically around the torus with a pitch that varies from one surface to the next. The pitch is described by a parameter called the $q$-value which is defined as the number of times a magnetic field line winds the long way around the torus (i.e., in the toroidal direction) divided by the number of times it winds the short way around (i.e., in the poloidal direction), in the limit of an infinite number of times. An important feature of ideal MHD is that the topology of the magnetic field is conserved in time and this restricts the types of instabilities that can occur.

While favourable ideal MHD properties are necessary in a reactor they are not always sufficient. Non-ideal effects such as electrical resistivity relax the topological constraint of ideal MHD to allow the growth of a weaker class of instabilities called tearing modes which can reconnect the field lines into different topologies. The effect of the resistivity is mostly felt in a narrow region around a rational surface, where the field lines are closed and the $q$-value is a rational number, $m/n$. A tearing instability originating on this surface therefore has a poloidal mode number $m$ and toroidal mode number $n$ and reconnects the field lines into a new topology with a helical structure that follows the same

![Figure 2: Plasma cross-section of (a) simply nested magnetic surfaces and (b) with an m/n=2/1 magnetic island.](image)

pitch, $m/n$, as the field lines on the original rational surface. In poloidal cross-section this appears as a chain of magnetic islands. See Figure 2(b) for the case with $m/n=2/1$. At large amplitude magnetic islands can degrade confinement but by themselves do not necessarily lead to the loss of the discharge. The linear stability of resistive MHD modes in toroidal geometry is also studied at CRPP using a recently developed code called MARS in conjunction with a new cubic finite element equilibrium code called CHEASE \[2\].

**Mathematical Model for Disruptions**

Simulation of the nonlinear evolution of plasma instabilities is highly demanding on CPU time. Disruption, which involves strong mode activity in three dimensions on multiple time and space scales, is no exception and therefore requires optimisation at all levels of the calculation. The first step is to streamline the mathematical model so that no more than the essential information is included. This is guided by a combination of experimental observation and previous calculations.

Helical perturbations and in particular those corresponding to the growth of a large $m/n=2/1$ magnetic island are nearly always detected at the onset of disruptive activity and as such are considered to play a major role in their trigger. The disruptions which follow do not always lead immediately to the loss of the plasma. Often a long sequence of minor disruptions of the thermal energy lasting up to a second occurs before the plasma is finally terminated by a major disruption in which the plasma current is also quenched.

Such observations make it clear that being able to model the growth and saturation of resistive instabilities is not enough to model disruptions. The effects of microscopic transport processes must be included to describe the evolution between instabilities. In general there is an intimate coupling of transport and instability processes which cannot be ignored in selfconsistent plasma evolution. For example, the current and temperature profiles are determined mostly by the transport properties of the plasma but when large scale MHD activity occurs the magnetic field structure is modified which then changes the transport. To model disruptions we therefore evolve the magnetic and fluid variables according to the resistive MHD equations. This is then coupled to an equation for the electron temperature evolution that includes effects due to ohmic heating, radiated power losses and strongly anisotropic heat conduction where the heat conduction along the field lines is much greater than that perpendicular to them.

The code including these equations has previously been run on the old Cray-1 at Lausanne to model a less severe and periodic relaxation known as sawtooth oscillations\[4\]. During a sawtooth a slow rise in central temperature due to ohmic heating is interrupted by an $m=1/n=1$ instability which causes a small but rapid collapse of the central temperature. The ohmic heating then raises the temperature to repeat the cycle. Since sawteeth are often observed leading up to a...
disruption it is important that they be properly simulated by the disruption model.

Disruptions are principally triggered when either density or current limits are exceeded. We are concerned here only with the former although the basic disruption mechanism is considered to be the same in both cases. As the density is increased radiation from higher atomic weight elements is also observed to increase. At sufficiently high density the energy loss due to this impurity radiation cools the edge of the plasma causing it to contract inwards. This has the effect of steepening the current gradient sequentially around various rational surfaces which destabilizes MHD modes of different helicities. Therefore an important element to simulating density limit disruptions is in the inclusion of radiation losses responsible for the initial profile contraction. These are specified using a simple model that makes use of experimental measurements from an actual JET disruption. Similarly the transport coefficients are also chosen so as to be realistic for JET but more about this later.

Numerical Scheme

The helical nature of the modes in a circular cylindrical model of a tokamak plasma often means that they can be efficiently represented by a Fourier decomposition in the angular coordinates (i.e., the poloidal angle $\theta$ and the toroidal angle $\phi$). This corresponds to the $z$ coordinate along the length of the cylinder) while a finite difference grid is needed to resolve the usually more intricate radial dependence. In the code the magnetic, fluid and temperature variables are expanded in a complex Fourier series in $m\alpha-n\beta/R$ which is truncated to mostly include modes, $n/m$, that have rational surfaces within the plasma. For example, the $n=1$ value (defined earlier), in the JET disruption that we are trying to simulate, ranges from just below one in the centre up to five at the edge. Tests including modes with $m/n$ outside this range showed that they tend not to get excited and can be neglected without significantly affecting the overall evolution. In fact except for the initial profile contraction phase and the final stage of a major disruption the mode activity mainly occurs within the $q=2$ surface making it only necessary to include a few modes outside this smaller region for most of the simulation. This is with the exception of certain low order modes such as the $m/n = 1/0$ mode which was found to play a critical role in nonlinear mode coupling during disruptions. In most cases from 16 to 27 modes were used depending on the resolution needed while the radial grid was made uniform with 150 grid points. In principle more efficient use of grid points could be had by packing them more around a rational surface to resolve the narrow reconnection layer but disruptions involve activity simultaneously occurring on several rational surfaces which can move radially thus making mesh accumulation useless. No symmetry of the solution is assumed although because the configuration space variables are real, there is symmetry about the origin in Fourier space making it sufficient to represent the modes with $n>0,m$ arbitrary and $n=0,m \geq 0$.

Time Stepping

Optimal time evolution is obtained using a semi-implicit, block tridiagonal timesteping scheme in which the linear terms and the equilibrium components of the nonlinear terms are evolved implicitly to maintain numerical stability while the nonlinear terms are evolved explicitly to avoid CPU intensive matrix inversions. The algorithm allows very long timesteps to be taken when the level of perturbation is small and has made possible the integration of the equations over transport timescales.

Even so, the explicit evolution of the nonlinear terms means the equations are still subject to a modified form of the Courant, Friedrichs, Lewy (CFL) constraint on the timestep (the CFL condition on fully explicit timesteping schemes basically requires that the grid velocity $\Delta x/\Delta t$ be greater than the fastest physical velocity to avoid the exponential accumulation of errors). This time-step limitation is particularly restrictive in situations of violent, fine scale, mode activity during which a broad spectrum of high energy modes is excited.

In order to always be running at optimal performance the code dynamically adjusts the size of the timestep, up or down, to keep as close as possible to the marginal numerical stability boundary without stepping over it. This is based on periodic testing of the mode energies as well as setting limits on allowable spatial gradients. Exceeding these limits causes the code to restart with a smaller timestep from a previous checkpoint and thereby prevents a numerical instability from taking hold.

Numerical Limitation on Transport Coefficients

In practice, simulation using realistic values of the transport coefficients for a large tokamak such as JET is far too demanding, and the ratio $S$ (Lundquist number) between the resistive timescale and the Alfvén (Hydromagnetic) timescale has to be reduced. This is for reasons of resolution and numerical stability. Firstly the width of the reconnection layer, which varies as $S^{-2/5}$, is narrower at high $S$ in JET, requiring smaller grid spacings and hence more grid points to be resolved. Whenever mode energies are high, the smaller grid spacings in turn require a smaller timestep to satisfy the CFL condition. Secondly the greater separation of timescales makes the evolution of phenomena occurring on a resistive or intermediate timescale take longer while the timestep is still limited by the nonlinear CFL constraint to values around the Alfvén time. To reduce $S$ down to values that can be simulated with an affordable resolution and still keep the relationship between various dissipative timescales the same as in JET, all the small transport coefficients of resistivity, viscosity and perpendicular thermal conductivity are multiplied by a transport enhancement factor $E$ in the range of a few thousand. Runs made at several different values of $E$ indicate that the disruption mechanism is not sensitive to the value of $E$.

Vectorisation

A high degree of vectorisation is achieved, around a factor of 8 faster than scalar mode on the Cray-2, by...
representing the large radial grid on the inner loop. In cases where the radial grid loop is not vectorisable, the loops are inverted to allow vectorisation of the loop over the Fourier components instead.

Although most of the computation is performed on the Fourier space representation the $T^{-3/2}$ ($T$=electron temperature) dependence of the resistivity is more efficiently computed in real space. To perform the 2-D complex Fourier transformation the NAG library FFT routine C06FRF is used twice: first over the $n$ mode index after which the result is put into Hermitian form (since the physical space variables are real) and then over the $m$ index. The routine is particularly efficient because it allows vectorisation of the radial grid loop while at the same time using the efficient Fast Fourier Transform algorithm on the mode loops. By choosing mode dimensions without large prime factors this potentially expensive part of the computation is kept down to a negligible proportion of the cpu time.

**Simulation Results and Numerical Performance**

A description of the sequence of nonlinear events leading to disruption in the simulation is given here followed by a discussion of the numerical efficiency during various stages.

In the simulation as in the experiment, the radiative contraction destabilizes a large $m/n=2/1$ tearing mode which helically deforms the shrunken plasma column by forming large magnetic islands as shown in Figure 2(b). The resulting $q=2$ helical quasi-equilibrium turns out to be unstable to a rather complicated instability involving strong coupling between Fourier components of the form $m=2n-1$ especially the $m/n = 1/1, 3/2, -1/0$ components. As a result reconnection occurs simultaneously on several rational surfaces and in particular the growth of a large $m/n=1/1$ magnetic island causes an almost rigid displacement of the thermal energy which flattens the temperature in the central region. See the first three plots of Figure 3(a). This is similar to the previously mentioned instability that gives rise to sawtooth oscillations, although, because the shrunken plasma is already quite cold outside the $q=1$ surface, the thermal collapse is much more severe here. Towards the end of this phase the $2/1, 3/2$ and $1/1$ magnetic islands are so large they overlap to cause field line stochasticity which breaks up the magnetic surfaces in the manner shown in Figures 5(a) to (c). In theory the growth of just a single chain of magnetic islands changes the topology but leaves the magnetic surfaces intact but when two or more magnetic island chains of different helicities overlap the field lines can no longer be constrained to a single surface and instead wander stochastically through a volume. By tracing field lines around the torus and marking the points where they cross a particular poloidal cross-section (i.e., a Poincaré plot), as in Figure 5, the stochastic or chaotic breakup of the magnetic surfaces due to this overlap can be seen very clearly in the $q<2$ region. The resulting loss of confinement in this region also causes MHD turbulence to develop with filamentation and broadening of the current profile as can be seen in Figure 3(b).

The stochasticity of the field lines in the region of high current gradient violates the equilibrium condition that the current be a constant on a field line, so that from this point onwards, either the magnetic surfaces must re-heal or the current gradients must be eliminated by violent MHD activity. The former case corresponds to a minor disruption in which the discharge recovers. However if the initial instability is sufficiently strong (this depends on the size of the $m/n=2/1$ magnetic island) the stochasticity gets worse so that a major disruption, corresponding to the latter case, occurs. The continued filamentation and broadening of

![Figure 3](image.png)

**Figure 3**: Plasma cross-section of (a) the electron temperature and (b) the current density, at different times during a simulated major disruption. The colours, in order of increasing numerical values are: blue, green, yellow and red.
the current profile (see last 2 plots of Figure 3(b)) then strongly destabilizes MHD modes with \( m \geq 2/n = 1 \) near the edge of the plasma, which overlap to breakup the last remaining magnetic surfaces. The final complete loss of confinement is thus demonstrated in Figure 5(d) by the field line stochasticisation having spread across the entire plasma cross-section. The simulated time dependent diagnostics shown in Figure 4 show that as in the experiment, the major disruption is preceded by a sequence of minor disruptions identified by sharp drops in the temperature and increased mode activity. The major disruption, in contrast shows much stronger mode activity and is particularly distinguished by finally making the current profile almost flat giving rise to a large drop in internal inductance as well as a negative spike in the toroidal loop voltage, in excellent agreement with experimental observations.

The numerical efficiency of the code is very much a function of the number and type of modes that are excited during the simulation and this of course depends on what is happening. When it is clear from the physics which modes are going to be excited the roughly \( M^2 \) dependence of the cpu time, where \( M \) is the number of modes in the simulation, means that large savings can be made by truncating the spectrum accordingly. This is the case for the initial profile contraction during which individual transient modes are sequentially destabilized and then restabilized by the inwardly moving radiation front. After the onset of the large \( 2/1 \) mode a much larger selection of Fourier components must be included to allow for nonlinear mode coupling. The most difficult phase, however, is during the short periods of a disruption when MHD turbulence develops. This excites a broad spectrum of high energy modes and since the CFL condition is more severe on higher order modes it is particularly difficult to properly converge the spectrum without incurring prohibitive costs in cpu time. The spectrum is however viscously damped by running with a large enhancement factor which means that mode convergence can be played off against realistic transport coefficients. Runs made with different numbers of high order modes together with runs made with different enhancement factors showed that small scale turbulence is not a critical factor for the disruption behaviour.

Conclusions

The simulations have shown that with the computing power of the Cray-2 one can perform realistic numerical experiments on transport timescales which are able to reproduce some of the most complex mode activity observed in real plasma experiments. The simulations in particular have clarified the mechanism of tokamak disruption.

References

TERPSICHORE: A THREE-DIMENSIONAL IDEAL MAGNETOHYDRODYNAMIC STABILITY PROGRAM


The 3D ideal magnetohydrodynamic stability programme TERPSICHORE has been developed to investigate and optimize magnetic plasma confinement configurations. The principal goal is to identify attractive fusion reactor devices. The vast parameter space that must be explored requires very efficient programming on vector-parallel computers, a task that TERPSICHORE has demonstrated with an almost 2 Gigaflop performance on an 8 processor Cray YMP machine.

On the path towards a thermonuclear fusion reactor there are several technological and physical uncertainties to be understood and solved. One of the most fundamental problems is the appearance in the machines of many sorts of instabilities which can either enhance the energy outflow or even destroy the magnetic confinement of the fusion plasma. The knowledge of such instabilities is a prerequisite to a good understanding of the behaviour of existing experiments, and to the design of new devices. Most of our effort has been devoted to the study of axisymmetric toroidal configurations such as tokamaks or spheromaks and to helically twisted toroidal devices such as stellarators. An example of such a configuration is the WVEIIX stellarator to be built at Max Planck Institut für Plasmafysik and shown in Figure 2.

The most violent global instabilities arising in timescales of microseconds are those described by the linear, ideal, magnetohydrodynamics (MHD) equations. These are obtained when the linearized motion of a magnetically confined plasma around its equilibrium state is studied. In this model, we neglect non-ideal effects such as the influence of finite resistivity, viscosity or kinetic effects. It is well suited to model reactor relevant plasmas having very high temperatures of the order of \(10^8\) K. The efficiency of a fusion experiment is measured by the parameter \(\beta\) which is the ratio between the plasma pressure and the confining magnetic field energy density. Reactor relevant \(\beta\) values should exceed 5%. We compute equilibrium solutions by means of the VMEC code [1] (developed at the Oak Ridge National Laboratory, USA).

The mathematical model of the stability problem can be expressed in a variational form

\[
\delta (W_p + W_v - \omega^2 K) = 0
\]

Here, \(W_p\), \(W_v\), and \(\omega^2\) are the plasma potential energy, the energy of the vacuum magnetic field surrounding the plasma column and the plasma kinetic energy, respectively. The variational form constitutes an eigenvalue problem with \(\omega^2\) as the eigenvalue. The eigenfunction \(\xi = \xi (r, t) = e^{i\omega t}\) is a measure of the displacement of the plasma column from the equilibrium position. A configuration is unstable if \(\omega^2 < 0\).

This model enables studying the ideal MHD stability properties of three-dimensional plasma configurations. Based on our experience from the intense computational effort necessary to study axisymmetric geometries with our 2D stability code ERAITO [2], treating 3D problems needs very careful choices of the numerical methods and of their implementation on parallel vector computers. In order to minimize the matrix size, we perform a Fourier expansion in the two angular directions (poloidal and toroidal) and a finite hybrid element approach in the radial direction. The coordinate system has been chosen on the basis of analytical considerations of unstable modes which interest us. Specifically, the coordinates follow the instability structure along the magnetic field lines in such a way that the coupling between Fourier modes is minimized.

The stability code TERPSICHORE consists now of 6 basic modules, categorised as: Interface to the MHD equilibrium, Reconstruction of the MHD equilibrium, Mapping the MHD equilibrium, Construction of the stability matrix elements, Eigenvalue solver, Analysis and diagnostics of the results. We intend to add other modules concerning an adaptive radial mesh requiring an interpolation of the equilibrium quantities, a mode selection algorithm to minimize the number of selected Fourier.
mode pairs and a penalty function to be added to the constructed matrix in order to eliminate destabilized modes belonging to the stable continuous spectrum.

The overall modular structure of TERPSICHORE is shown in Figure 1. The program is embedded in the ASTRID programming platform which is described elsewhere [3].

Our stability code has been applied to study existing and prospective 3D configurations shown in Figures 2 to 6. Stability studies with TERPSICHORE and the CAS 3D code [4] on the proposed W VII-X stellarator (Figure 2) have confirmed the prediction that the machine will be stable up to \( \beta = 4.5\% \), a value close to that necessary for a reactor relevant plasma. On the other hand, stability studies on the ATF stellarator (Figure 3) installed at the Oak Ridge

**Figure 1**: Diagram of TERPSICHORE

**Figure 2**: The proposed W VII-X stellarator configuration with the coils that confine the plasma shown in grey. The plasma surface is shown in red on which a magnetic field line is traced in green. This configuration has 5 field periods. An internal flux surface is shown in yellow in one section of the device. The shape of the flux surfaces at two different toroidal planes are depicted in different shades of orange.

**Figure 3 (a)**: The perturbed pressure distribution at the plasma-vacuum interface for the standard ATF configuration with a finite toroidal plasma current in the same direction as the magnetic field. The perturbed pressure attains maximum values in the regions depicted in red and minimum values in the regions depicted in blue, respectively. The configuration is weakly unstable with a dominant \( m = 3 \) (poloidal mode number), \( n = 2 \) (toroidal mode number) structure at \( \beta = 4.5\% \). The coordinate system in which the calculations are performed is superimposed.

**Figure 3 (b)**: Perturbed pressure distribution for an \( n = 1 \) instability on a toroidal cut.
known radio frequency current drive techniques. It shows that taking into account new parameters can alter the stability properties of magnetic fusion devices. A 4 period Helias stellarator, a figure 8 stellarator as well as a racetrack tokamak are shown in Figures 4, 5 and 6, respectively. All of these devices are being studied intensively with TERPSICHORE to understand regimes of stable operation which could lead to designs of the next generation of experiments and reactors. One of the final goals of ideal 3D MHD stability computations is to find stable configurations in which the plasma energy in a relatively dense gas at a temperature of $10^8 \text{K}$ can be magnetically confined during at least 1 second.

**Figure 5 (a):** The equilibrium pressure distribution on 84 different toroidal planes. The pressure is a maximum at the centre of the plasma depicted with the colour red. It vanishes at the plasma surface where the colour changes to dark blue.

**Figure 5 (b):** The perturbed pressure distribution at the plasma-vacuum interface.

National Laboratory in the USA have revealed a new type of a stable regime. Stellarators have always been designed and built in such a way that no externally induced plasma currents flow in the device except for those driven internally by the plasma pressure gradient. However, in ATF we were able to show that by adding a small net toroidal current in the direction of the magnetic field (in order to enhance the twist of the field lines), the $\beta$ value could be increased from 3% (no net toroidal current) to 4% (with current). This is a promising result because this effect can be realized by driving a small current near the plasma surface with well

**Figure 3 (c):** Displacement arrow plot of an $n = 2$ instability on the same toroidal cut.
TERPSICHORE has been designed to run efficiently on vector-parallel computers either of shared or distributed memory types. Evidently we have succeeded based on speeds achieved with the code. The fixed boundary version achieved 1.74 Gflops on a Cray Y-MP eight processor machine just by using the autotasking system CF77 (which had a parallelism of 7.35 of a possible 8). The free boundary version that was constructed in 1990 is running at approximately 2.0 Gflops but has not yet been fully optimized. We believe these speeds are maximal in the Cray "world" because they have let us win a Cray Gigaflop Performance Award both in 1989 and 1990.

More recently, TERPSICHORE has been transported to the INTEL iPSC2 hypercube which has 8 processors. For a small test case of only 16 radial intervals, a speed-up factor of 7.1 with respect to the execution on a single processor has been measured. Here, slight improvements are still possible. We expect an almost linearly increasing speed-up factor for a case with 128 radial intervals on a 64 processor hypercube.

Acknowledgments

We would like to express our gratitude to Stephen Behling of Cray Research Inc., to Arno Liegmann and to Jean-Pierre Therre of Cray Switzerland for the time and efforts devoted to the successful parallelization of the TERPSICHORE code.

This work was supported by the Swiss National Science Foundation.

References:


