Progress Report
1993

PDC
Center for Parallel Computers

Progress Report
1993

KTH

Front Cover Picture: A rendering of a geometric model of the computers installed at the Center for Parallel Computers at KTH. In the picture we see the Connection Machine CM200 with its framebuffer and DataVault, the MasPar MP-1, the Silicon Graphics Onyx VTX, and also the newly installed FDDI ring at PDC connected to SHPCNet and the world. The rendering model was developed by Johan Ihrén, using the ray tracing program *rayshade.*

Back Cover Text: This is David H. Bailey's list of different ways to 'improve' the performance of ones parallel program. It is a 'tongue in cheek' summary of questionable practices found in a field that is not yet quite mature. David H. Bailey works at the NASA Ames Research Center and can be reached at `dbailey@nas.nasa.gov`.

PDC Parallelldatorcentrum
Royal Institute of Technology
S–100 44 Stockholm, SWEDEN
Telephone: +46–8–790 78 84, +46–8–790 78 91
Telefax: +46–8–24 77 84
Email: info@pdc.kth.se
WWW: http://www.pdc.kth.se/

Publisher: Parallelldatorcentrum
Editors: *Fredrik Hedman*, *Per Hammarlund*, *Jesper Oppelstrup*
Typesetting & layout: *Jan Michael Rynning*, *Fredrik Hedman*
Printed by: *Ekblads, Västervik, November 1994*

ISBN 91-7170-805-7

Foreword

The Center for Parallel Computers provides open and convenient access to parallel computing for Swedish academic and industrially related research. The Center promotes the use of parallel computing through courses, workshops, a research program and provides access to applications engineers for the users.

During 1993 a number of new research groups have started to use the PDC facilities. The earlier users in the different fields of physics, chemistry, mechanics, biology, geophysics and computer science have continued, many of them with new exciting projects.

In December 1993, PDC organized an international conference on Applications in Parallel Computing "Parallel High-Performance Applications", with around 90 participants. The conference should be seen as part of PDC's effort in promoting the knowledge of parallel computing and increasing international cooperation.

During a period, PDC has rented a MasPar MP-1 computer from Digital Equipment in Sweden. The computer was successfully and rather heavily used in application areas like Computational Electromagnetics and also in weather forecasting.

The international trend of increased importance of parallel architectures within high performance computing became very pronounced in 1993. A clear sign was the entry of CRAY and IBM as serious manufacturers of high performance parallel computers. In addition, the number of vendors of scientific and commercial parallel software increased substantially during the year.

When this foreword is being written we are nine months into 1994. PDC has just doubled the capacity of the CM200 to 16K processors and 2 GByte of memory. The users have immediately taken advantage of the increased capacity. An IBM SP-2 system with 29 nodes and 4 GByte of memory will soon be installed. With this system, PDC can offer high-level, state-of-the-art MIMD capability to both industrial and academic users. The computers will be housed in the recently modernized part of KTH. The PDC local environment now includes integrated machine rooms, a computer graphics laboratory, support staff offices and conference rooms, all located centrally at KTH.

With these new facilities PDC can serve large groups of scientists, engineers and students with easy access to a diversified parallel computing environment. However, scientific development is pressing on and there is a continuous need for very large scale parallel platforms for grand challenge problems. These large systems are needed for breakthroughs in an increasing number of fields of science. We are confident that the new National High Performance Computer Council will make every effort to support continuous upgrading of the national supercomputer facilities.

The industrial users will benefit from the creation of the Parallel and Scientific Computing Institute (PSI). The proposal for this Institute was chosen by NUTEK as one of the centers of excellence to be established in the near future. Naturally, PSI will work closely with PDC in the increased contact between PDC and industry. The Institute will work with parallelization of algorithms and existing codes as well as development of new software.

PDC is looking forward to the coming year with new facilities and an active academic and industrial research program. Our priorities will be to reach new groups of users from both industry and academia and to increase the knowledge of high level parallel computing in Sweden.

The Board of the Center for Parallel Computers, September 1994

Contents

1 PDC – Parallelldatorcentrum

This third progress report of the Center for Parallel Computers (PDC) at the Royal Institute of Technology (KTH) covers the activities of the center during 1993. A brief overview of this year's activities at the center follows in this section. User projects are described in Sections 3 through 10. The last three sections contain a glossary, a bibliography and an index.

1.1 Background

To stimulate research and spread information on the use of parallel computers

The Center was formed in January 1990 to act as a focal point and national forum for research on and use of parallel computers. Our goal is to stimulate research and spread information on the use of parallel computers.

This is achieved by providing high-performance parallel computers and expertise on their use to the technical and scientific computing community in Sweden. Parallel computing is an important and necessary technique because parallel computers seem to provide cost-effective increase in capacity to levels not previously anticipated.

Parallel computing however requires new ways of approaching a problem and in many instances new algorithms and techniques – in short one has to learn to think in parallel. This means that practical experience of parallel computing in many different areas of science is important. The computer resources at PDC are used both in projects which have as their main goal to develop new methods and in projects which are more oriented to solve a large problem in some application area.

To achieve our general goal it is of increasing importance to take an active part in the development of new methods, algorithms and tools and also in application projects. This is necessary in order to develop and maintain the competence at the center.

The Connection Machine CM200 is the most powerful computer in Sweden for a number of problems. The machine with its large primary memory is especially useful in problems which require large amounts of data.

PDC is an interdisciplinary organization and our intention is to disseminate knowledge of how to apply parallel computers in science and engineering to experts in both computer science and computing intensive application areas. We feel that the organization of PDC as a small independent unit in close cooperation with different research groups has worked well.

The procurement of the next generation parallel supercomputer at PDC is currently going on with a funding of 15 MSEK from FRN. This system will be more powerful than the Connection Machine CM200 and have an architecture which is applicable to a larger class of problems. This new computer will further increase the number of application projects at PDC and will require additional user support.

Most of the application software used at PDC has been developed locally. As a consequence the need for user support is larger compared to what is needed at supercomputer centers with more conventional architectures and a larger base of commercial third party software.

The application engineer has proved to be an important and efficient way of providing user support. The application engineer takes part in a number of research projects in the same way as the other researchers of the project, and through his in depth knowledge of parallel computers he can contribute with new views on how to approach the problem at hand as well as advise on how the implementation should be done.

PDC has recently received support from NUTEK for an application engineer especially directed to industrial applications. This will make it possible to introduce the technique of parallel computing in Swedish industry on a much broader scale.

1.2 Organization of PDC

The center has a staff of four persons, or about three full-time equivalents: Fredrik Hedman, application engineer; Johan Ihrén, UNIX system manager and graphics specialist; Britta Svensson, administrative assistant; and Gert Svensson, project coordinator. Associated part-time staff are Per Hammarlund and Lars Malinowsky.

As a part of a recent reorganization of KTH, PDC was in April 93 transferred from the Department for Teleinformatics (previously

TDS) to the Department of Numerical Analysis and Computer Science (NADA). Professor Lars-Erik Thorelli has been the chairman of the board from the formation of PDC until May 93, when Professor Björn Engquist became chairman. Members of the board:

Björn Engquist	Professor of Numerical Analysis, NADA
Fredrik Hedman	Application Engineer, PDC
Anders Lansner	Director of Research SANS, NADA
Jesper Oppelstrup	Lecturer, NADA
Yngve Sundblad	Chairman of the KTH Computer Council
Gert Svensson	Coordinator, PDC
Lars-Erik Thorelli	Professor of Computer Systems, IT

1.3 Funding

The original grant of 10 MSEK for the CM2 was given by Skandinaviska Enskilda Bankens Stiftelse för Ekonomisk och Teknisk Forskning and the Swedish Council for Planning and Coordination of Research, FRN. The operational cost, including staff, has been covered by the Royal Institute of Technology, the Swedish National Board for Industrial and Technical Development, NUTEK, the Swedish Research Council for Engineering Sciences, TFR, and the Swedish Natural Science Research Council, NFR. Our past and present funding is detailed in the table below.

A grant of 8 MSEK for the budget year 93/94 and 7 MSEK for 94/95 has been given by FRN to purchase a scalable MIMD-system. This procurement is underway.

Cost of Operation and Staff

Grants kSEK	89/90	90/91	91/92	92/93	93/94
NUTEK	1500	1500	1785	473	920
KTH	100	300	380	880	880
TFR	0	0	0	300	300
NFR	0	0	0	0	300
Total	1600	1800	2165	1653	2400

The relatively low cost 92/93 was because the upgrade of the Connection Machine included a one year warranty.

1.4 Hardware Resources

Current Hardware

- Connection Machine CM200 with 8 192 processors, 1 GByte memory and a 10 GByte DataVault. Scalable massively parallel SIMD-computer with data-parallel programming environment.
- MasPar MP-1 with 16 384 processors and 1 GByte of memory.
- High-end equipment for visualization. One SGI Onyx VTX and three SGI Indigo R4000.
- Five workstations for personal use and visualization.
- Equipment for access to SHPCNet (Swedish High Performance Computing Network). Currently being installed.

The Maspar Computer

As part of an engineering research cooperation project PDC has rented a MP-1 computer from Digital Equipment in Sweden, starting in August 1993. Initially the computer had 8K processors but it was later upgraded to 16K processors and a total of 1 GByte of memory. The frontend computer also went through a series of upgrades to match the needs of PDC's users.

The architecture of the MP-1 computer is similar to that of the CM200 with a couple of interesting differences in network topology and memory organization. The MP-1 processors are organized in a 2D grid for nearest neighbor communication and are also connected with a general purpose communication network. The MP-1 can be seen as a register based machine as compared to the CM2 that is more of a memory based machine. The MP-1 has a large number of registers on-chip and also faster support for indirect addressing.

The MP-1 was primarily used for computational electromagnetics (CEM) and weather forecasting. Examples of projects that have used the MP-1 can be found in Sections 4.3, 4.6.

Our experiences with the MasPar were positive and it was interesting to make comparative studies of its performance with the CM200. It is also our impression that many users appreciated being able to test both systems and then to use the one best suited for their needs.

Graphics Workstations

During 1993 several new workstations from Silicon Grahics, with a 2-CPU Onyx VTX as the most powerful one, were installed at NADA. They are intended specifically for scientific visualization and are located close to PDC. Currently experiments with a parallel interface between the SGI Onyx and the Connection Machine are underway with the intention that it should be possible to use the Onyx for realtime visualization of computational results from the Connection Machine.

Planned Hardware

Scalable MIMD-computer with programming environment for integrated parallelism. Currently being purchased by a grant of 15 MSEK from FRN.

1.5 Educational Activities

Throughout the year several seminars on the use of parallel computers have been arranged. The Connection Machine has been used for courses in the M.Sc. program at KTH and CTH. Two complete courses (3 KTH-credits each) on programming of massively parallel computers have also been arranged:

Course	Date	Students
Parallel Computer Systems	Spring 93	60
Programming of MPP Computers	Mar. 93	23
Programming of MPP Computers	Oct. 93	18
New Comp. Arch. for Numerical Calculations[a]	Jan. 93	16
Numerical Solution of Large Sparse Systems[a]	Nov. 93	11

[a] At CTH.

1.6 The Conference and Tutorial

During December 15–17, 1993 PDC organized a conference and tutorial on Parallel High-Performance Applications. The event took place at KTH, with tutorials during the first day and sessions during the last two days. The conference had about 90 participants, many from other Nordic countries and continental Europe. Sponsors of the conference were: Digital Equipment Corp., MasPar Computer Corp., and Thinking Machines Corp.

The tutorials were given by Björn Lisper, KTH, Robert Schreiber, RIACS, and Barry Smith, UCLA. The topics that were covered were: introduction to parallel computing, overview of parallel hardware architectures, data parallel programming, High Performance Fortran (HPF), message passing programming, and examples of parallel algorithms.

For the conference sessions there were 13 contributed papers and 9 invited speakers:

- Günther Bachler, AVL, *Parallel Applications in Automotive Industry.*
- Petter Bjørstad, University of Bergen, *Domain decomposition and parallel computing with applications to oil reservoir simulation.*
- Ian Duff, RAL & CERFACS, *Exploiting vector and parallel computers in sparse calculations.*
- Lennart Johnsson, Harvard University & Thinking Machines Corporation, *Scientific Libraries on Scalable Architectures.*
- Diane Lynch, Thinking Machines Corporation, *Molecular Dynamics Simulations on the CM-5.*
- John R. Nickolls, MasPar Computer Corporation, *Data-Intensive Applications on Massively Parallel Clusters.*
- Rob Schreiber, RIACS, *Subway, a communication compiler for the Maspar MP-x computer.*
- Barry Smith, UCLA, *Large PETSc: a Portable Extensible Large Toolkit for Scientific Computation.*
- Brian Wylie, Centro Svizzero di Calcolo Scientifico, Switzerland, *PARAMICS: Parallel Microscopic Traffic Simulator.*

The last topic of the conference was a panel discussion on *Industrial use of Parallel Computing.* The panel laboured with questions like: "How can industry benefit from high-performance computing?" and "Is parallel computing mature enough to be used industrially?".

2 Color Plates

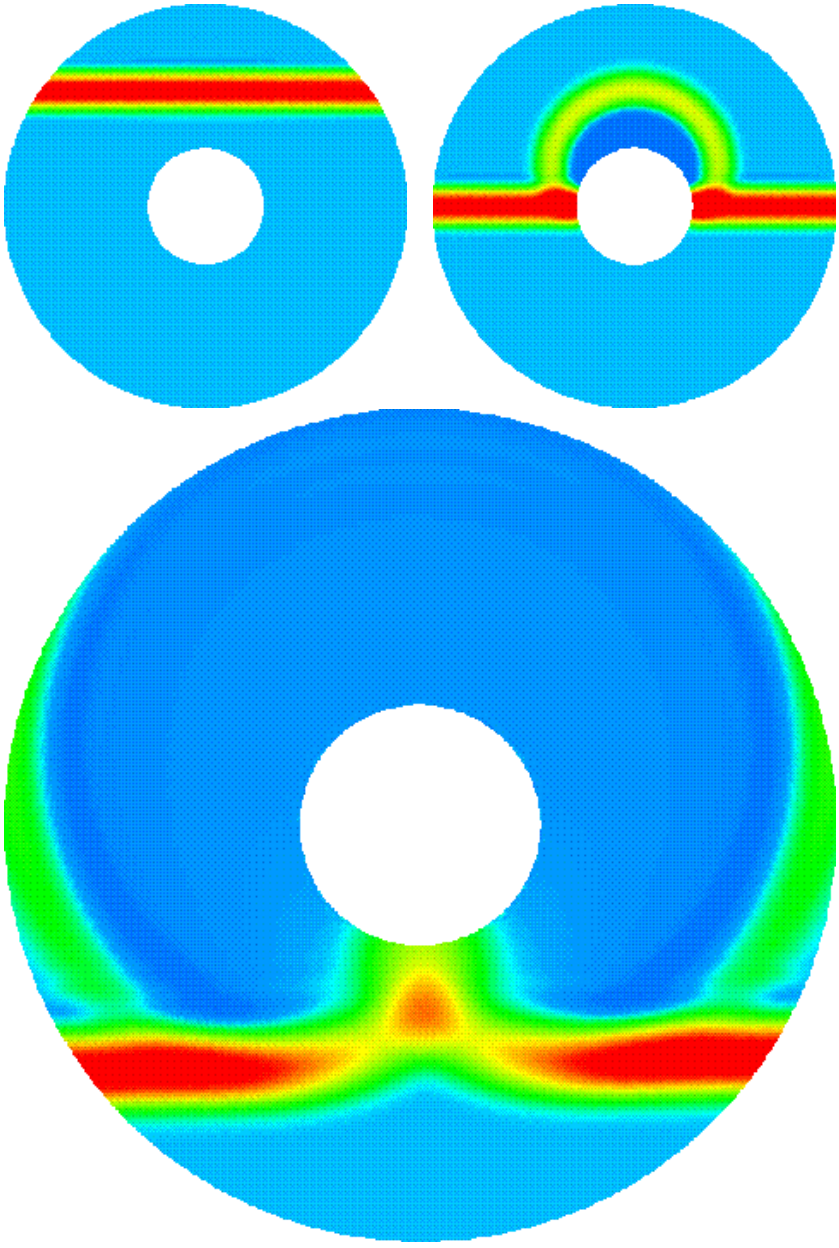


Figure 2.1. A plane TE pulse enters the computational domain from above, top left picture. The pulse strikes a metallic cylinder and is scattered by the cylinder. The scattered pulse propagates outwards from the metallic surface, top right picture. In the bottom picture the incoming pulse has just passed the cylinder. The scattered pulse leaves part of the computational domain and we can see creeping waves on the metal surface. The pulses leaving the computational domain must not be reflected back by the artificial outer boundary. Therefore great care must be taken in creating an absorbing outer boundary condition. The calculations have been made on the MasPar using a O-grid with 1024×128 cells. It is clear that the absorbing boundary condition works well. (See Section 4.3 on page 34.)

Figure 2.2. Pressure contours for inviscid flow in a supersonic air intake computed using an eight-block mesh. High pressure is indicated in pink, low pressure in blue. The white lines indicate the block boundaries. (See Section 4.5 on page 38.)

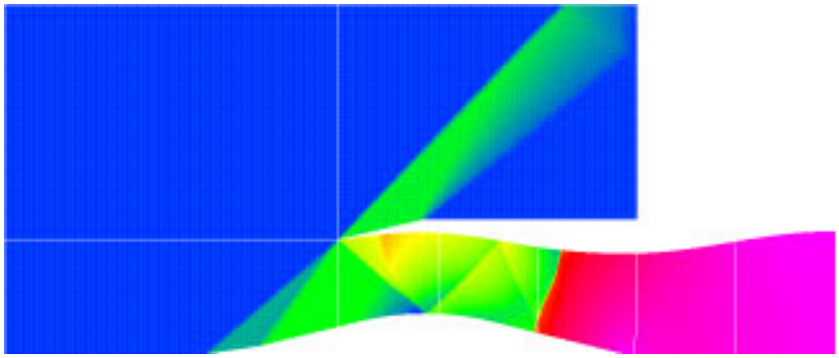
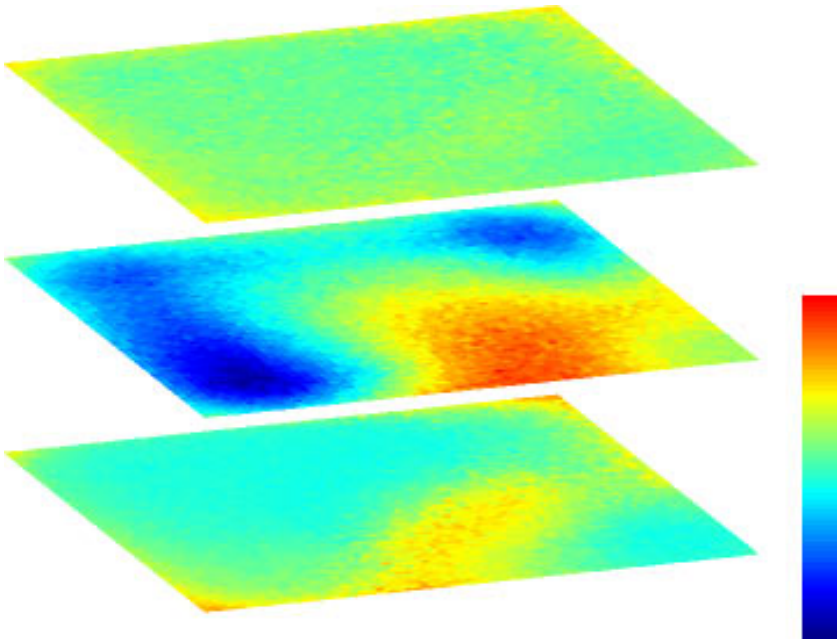


Figure 2.3. Simulation results of the dynamics of the 3-layered olfactory cortex model following a weak input pulse. The activity at time $t = 78$ ms in the three layers, each consisting of 64×64 network units. (See Section 3.4 on page 24.)



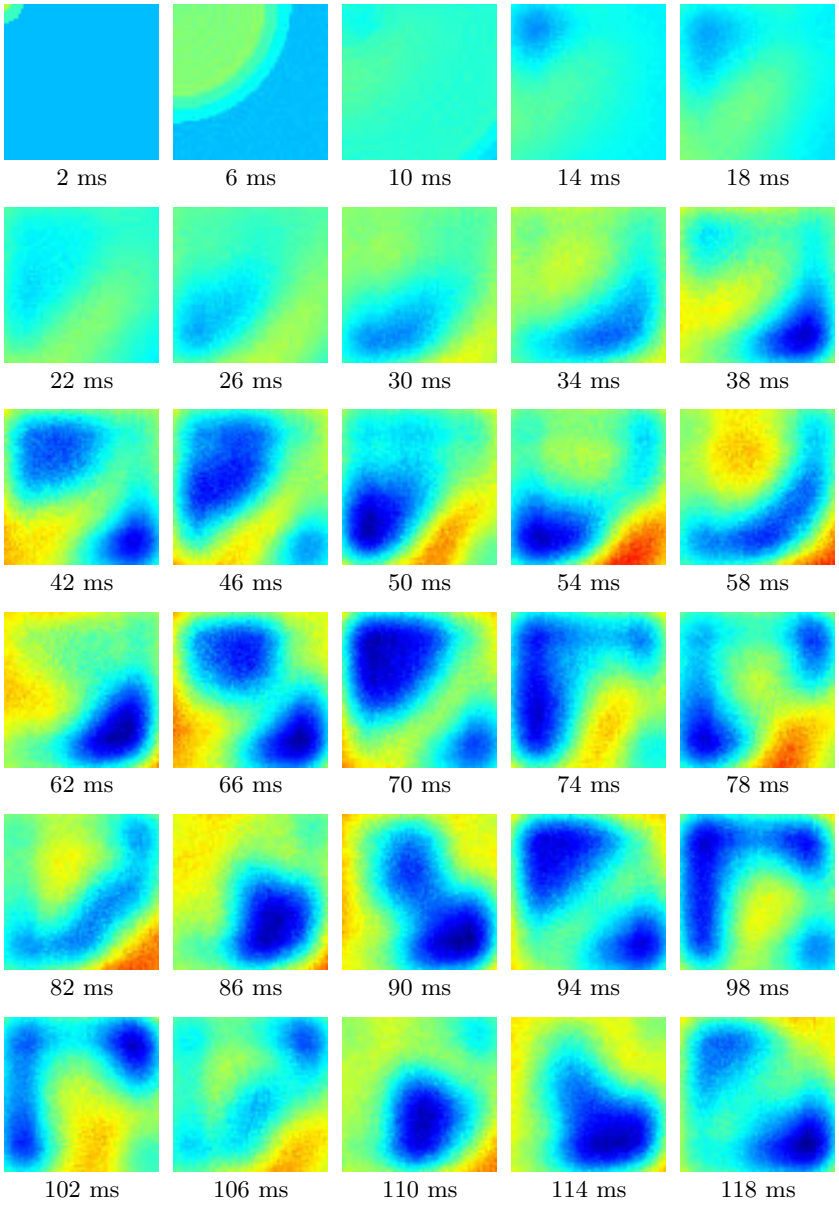


Figure 2.4. Each graph displays the activity of the 64×64 network units in the excitatory layer at time intervals of 4 ms, going from top left to bottom right. The corresponding area of the real cortex is 10×10 mm. The network units represent populations of (pyramidal) cells, with the activity given by the "mean cell membrane potential". The short pulse, which originates at the upper left corner of the network, initiates waves of activity that move across the network surface. The unit activity values range from -4.0 in deep blue to 6.7 in red. Units with positive values are sending to other units, those with negative values are silent. Each 0.5 ms time step of this simulation required 0.8 seconds on the 8K CM200. (See Section 3.4 on page 24.)

Figure 2.5. Velocity model showing the salt dome (4400 m/s material) from which the synthetic seismic section was generated. (See Section 8.2 on page 72.)

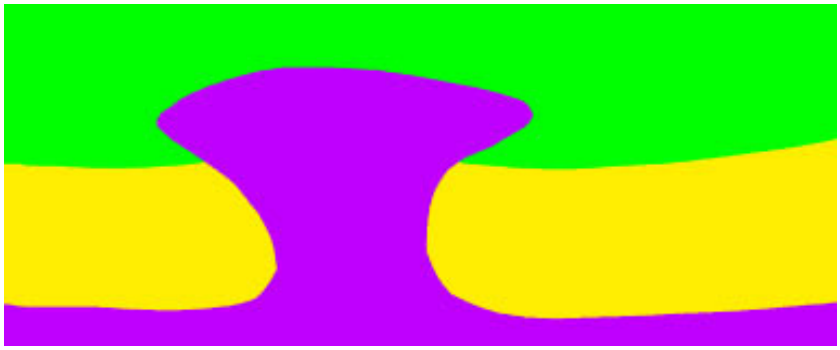


Figure 2.6. Snapshot of the wavefield shortly after the reflectors have exploded.

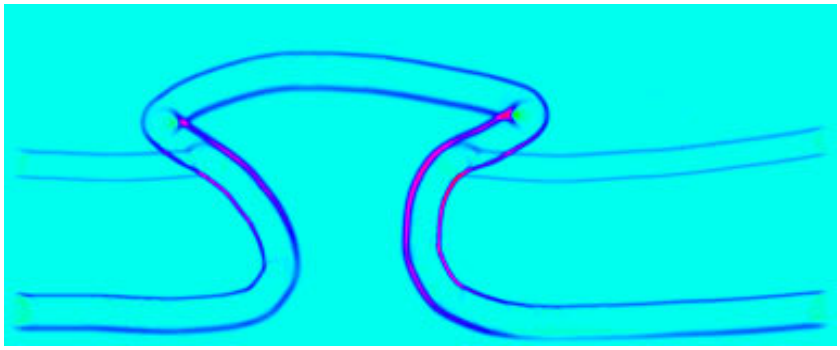


Figure 2.7. The synthetic seismic section that is recorded over the velocity model.

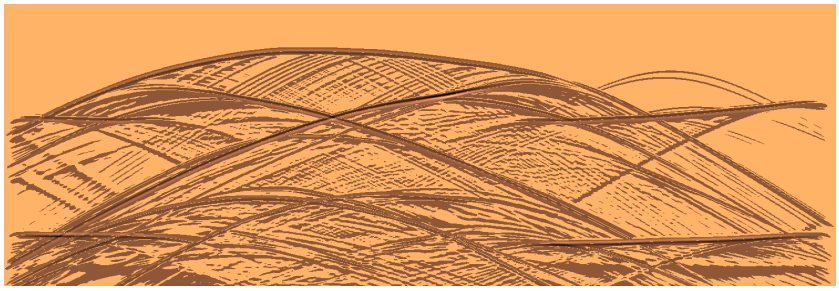
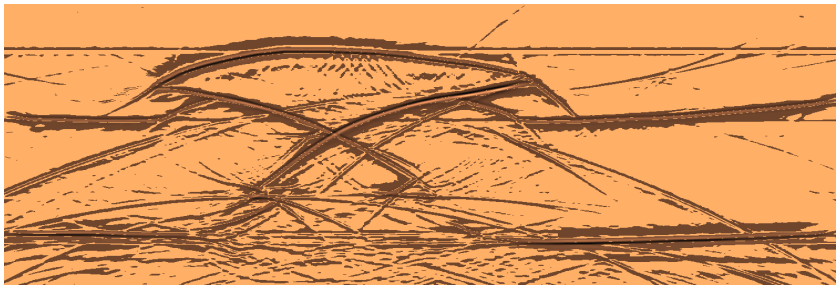


Figure 2.8. The seismic section after migration using a 1D velocity function.



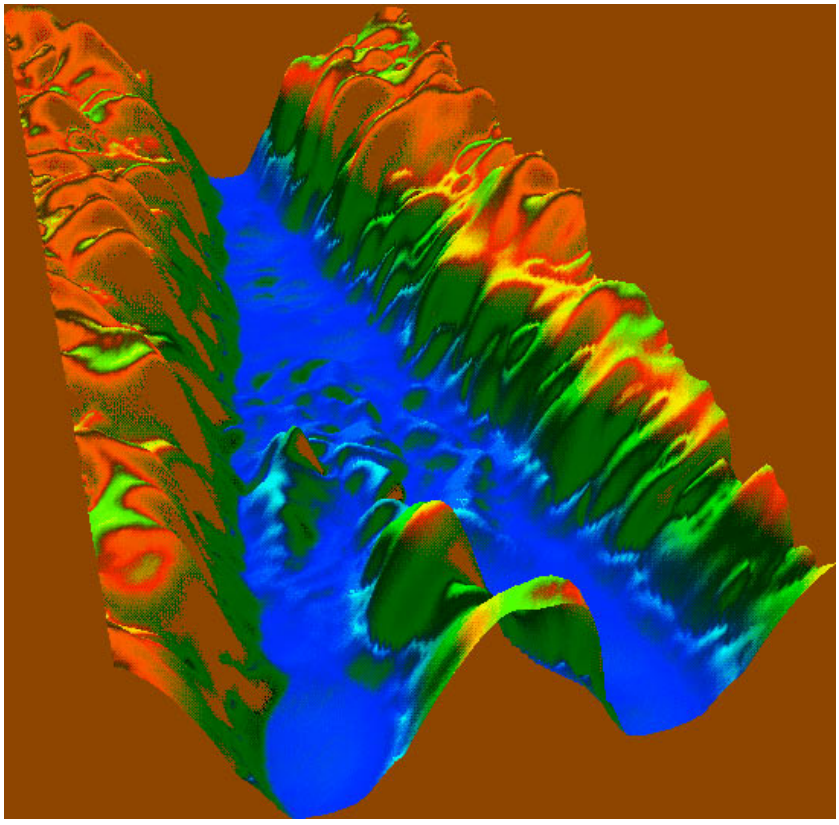


Figure 2.9. An electron wave-packet entering from the top propagating through a Y-branch switch. Ideally the electron would be equally split between the two lower waveguides. Random fluctuations in the potential causes it to instead mainly enter the left branch. (See Section 5.3 on page 49.)

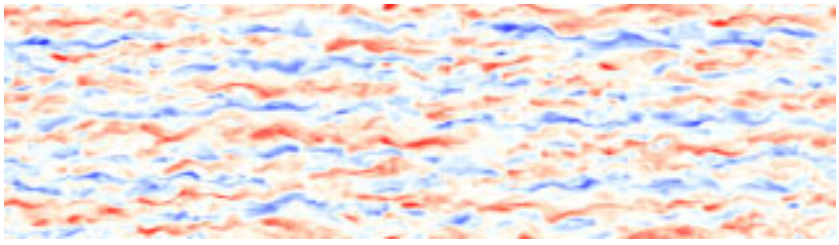
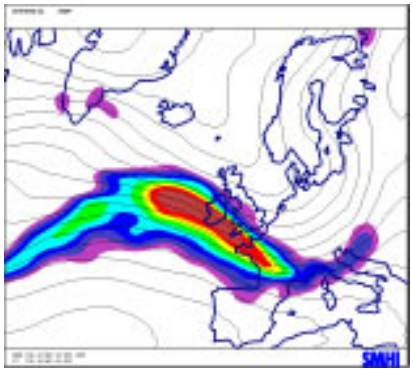
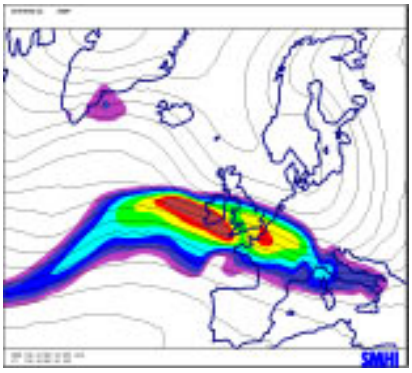


Figure 2.10. The streamwise velocity field in the x-z plane at the center of the channel, at Reynolds number 750. In the laminar flow the instantaneous velocity is zero at the center line, but in the turbulent case the instantaneous velocity is non-zero. The deviation from zero are color coded in read and blue, for positive and negative deviations respectively. (See Section 4.4 on page 36.)

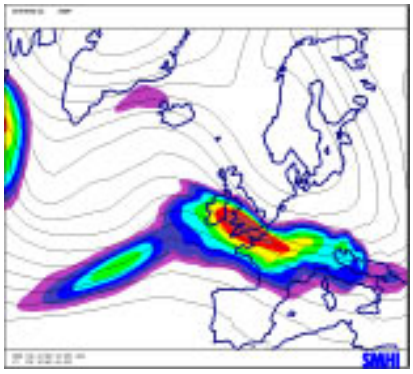
Figure 2.11. Jet winds at about 10 km altitude from the same forecast. The wind speed is color coded with 5 m/s intervals starting at 40 m/s in violet up to more than 75 m/s in dark red. (See Section 4.6 on page 41.)



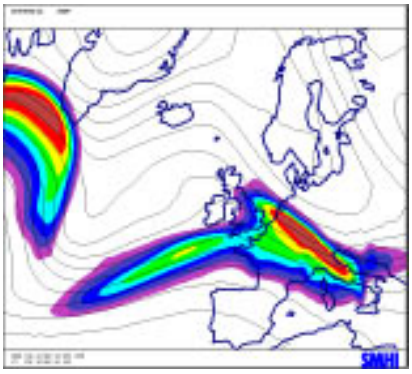
0 h



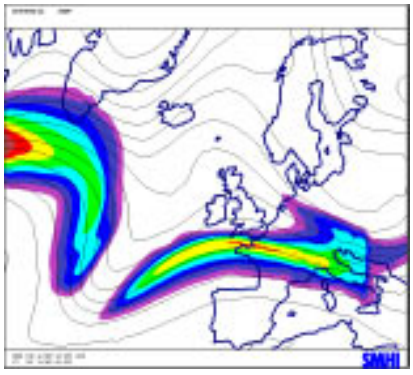
12 h



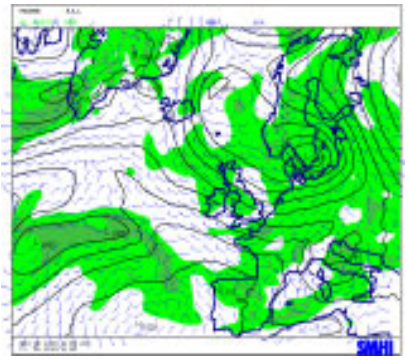
24 h



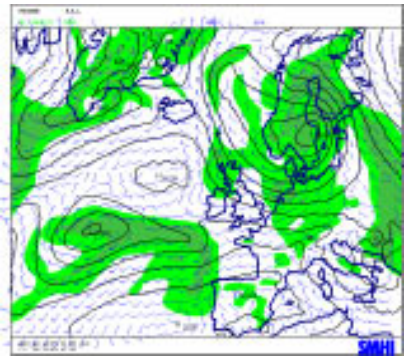
36 h



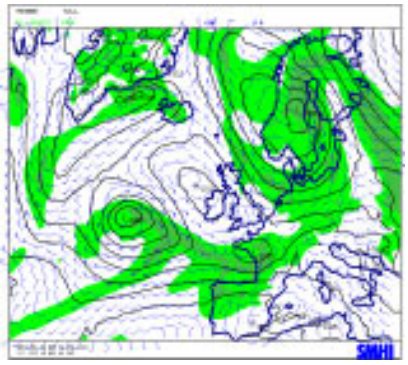
48 h



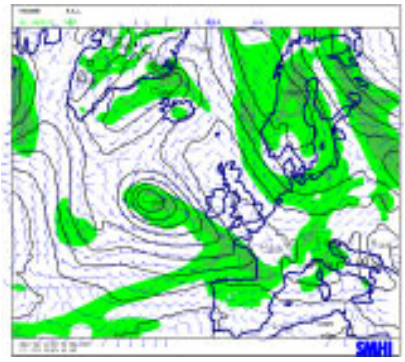
0 h



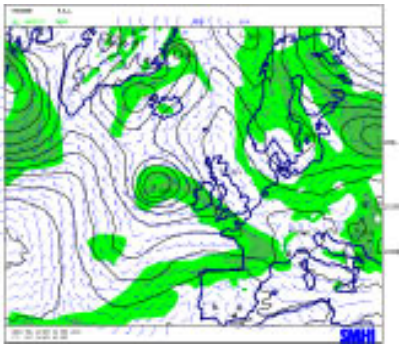
12 h



24 h



36 h



48 h

Figure 2.12. A 48 hour forecast at 12 hour intervals. This forecast was done on MasPar MP-1 with 110×100 horizontal gridpoints (55 km grid distance) and with 16 vertical levels. The complete forecast runs in 20 minutes on the MP-1 and in 10 minutes on the MP-2. One low pressure system over Southern Scandinavia moves north, while decreasing in intensity, and another low pressure system is approaching western Europe from the Eastern Atlantic. The black contour lines show the mean sea level air pressure. The blue arrows indicate the wind direction with more "feathers" for higher wind speeds. The green colors show relative humidity. The bright green areas are more or less overcast with 80% to 95% relative humidity and the darker green indicate rain areas with more than 95% relative humidity. (See Section 4.6 on page 41.)

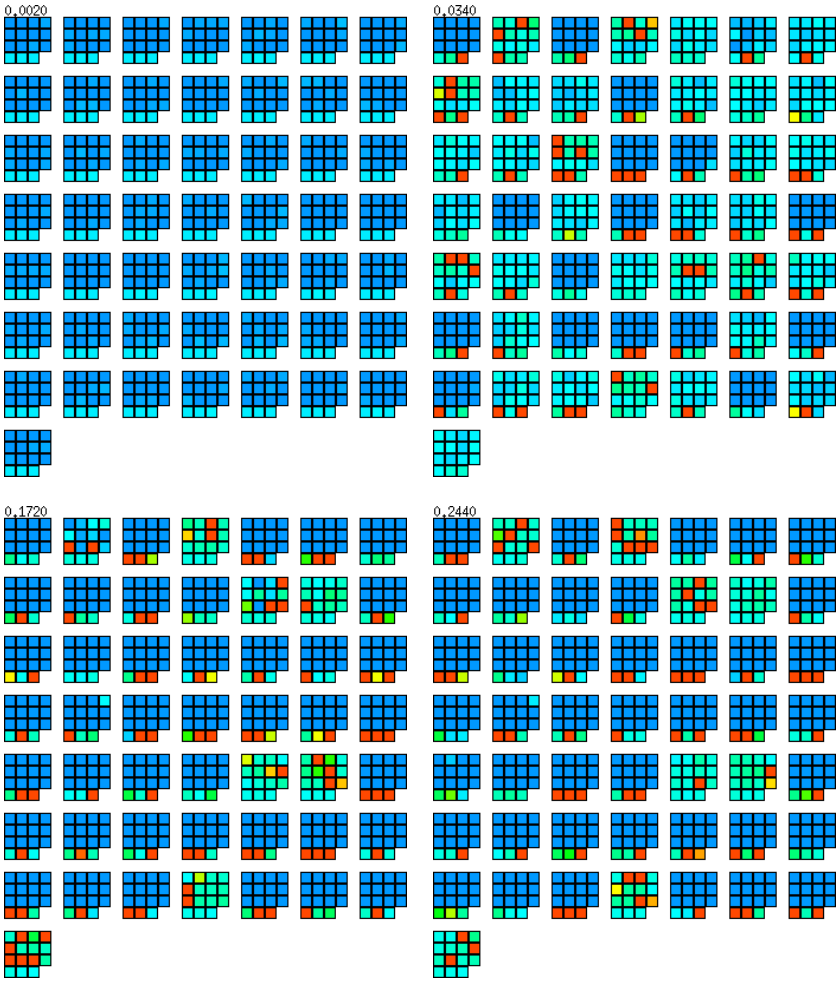


Figure 2.13. Output from a simulation of a network with 750 cells and 18,000 synapses. Each small square shows the soma potential of a cell with spiking activity red-yellow and resting blue-purple (maximum over 2 ms from -65 mV (purple) to +0 mV (red)). The total time simulated was 300 ms. Four frames are shown representing activity in 2 ms bins. Each column is shown as a 3×4 block of RS-cells and 3 FS-cells and the 50 columns are displayed as a $7 \times 7 + 1$ block. The stored pattern is the one shown in the margin of this section. (See Section 3.3 on page 21.)

3 Neural Modeling and Computation

Most work in the field of Neural Networks (NNs) is today done on standard sequential von Neuman machines. It has been pointed out many times that such sequential computers are quite inadequate for execution of NNs, as the networks themselves are inherently parallel. For this and other reasons it is important to implement the NN-algorithms and architectures on parallel hardware. Above all, this enables high computational capacity and therefore also the simulation of more realistically sized and structured NNs. For neurocomputing applications, this allows us to investigate the scaling properties of our algorithms as problem size increases. It also opens up the possibility of escaping from "toy problems" to some real-world applications. A further, perhaps less obvious, reason for this implementation research is that the use of parallel hardware puts relevant constraints on algorithm development and discourages inadequate solutions with embedded sequential sections. For our biological simulations, the use of efficient parallelized simulators means that we can get closer to the actual numbers of neurons in the systems under study and that we can simulate neuronal structures comprised of several interacting sub-networks.

Enables the simulation of more realistically sized Neural Networks

3.1 Document Retrieval Protein Sequence Matching

Björn Levin, Anders Lansner
SANS, KTH

The task examined is how to select relevant documents from a data base given a text description of the area of interest. A neural network is used to create a distance measure between the description and the documents of the database. Since the documents are treated as mere strings of characters the same procedure can be used to assign distances between the documents of the data base and can thus be used for clustering purposes. As the conventional coding for protein sequences are strings of letters, clustering and search for similarities among these sequences can be handled by

the same mechanisms. The reason for our interest in this specific field is the existence of widely accepted automatic methods for determining distances, methods that we then use to compare our achievements against [Wallin, 1992].

Naturally the essential part is how the distances are assigned. Our approach is to let the neural network determine these by building detectors sensitive to different features in the documents. Such a feature could for instance be the presence of a certain word or part of a sentence or protein sequence. Our system generates such feature detectors randomly and then evaluates them according to several investigated quality measures. These created units are then connected using the Baysian learning rule of the SANS I model [Lansner and Ekeberg, 1989a] to units representing the documents. In the retrieval phase the activity of these units constitute the representation of the assigned distances. It should be noted that the procedure of creating feature detectors has a far wider aspect than the document comparisons it is used for here. It is an example of the general creation of necessary complex units, needed in any neural network. The specific task of document retrieval serves here both as a platform for general research as well as an important real problem to be attacked [Levin and Lansner, 1992].

To be more specific about the continuation of the research that has been carried out during the year, effort has been directed towards adding another level hierarchically as well as including certain recurrent connections. One of the underlying ideas for this work has been to create sets of mutually exclusive units, i.e. sets where for any given input only one of the members is activated [Holst and Lansner, 1993]. At the same time the system has been extended to handle graded units; we have moved on from the binary unit, capable of signalling only the presence or absence of a feature, to a unit with a real valued output, capable of signalling also its estimate of the probability of the features presence in the data. This posed many problems algorithmically, but is of great value in many instances where a simple yes-no signal is too coarse.

During learning the times needed have increased. A non-graded learning can still be completed in around 1.5 hours using the new code but a complete treatment, including the disjunct sets, requires about 8 times more time. In that time all possible complex units built from up to 20 primary units have been evaluated, 2048

selected, all pair wise correlations examined as well as all correlations of 2048 selected second level units; in all a considerable amount of statistics gathered from the protein data base mentioned earlier.

The modifications have not changed the retrieval times significantly. The almost constant overhead has increased by a few seconds; the system is still able to rank 20,000 protein sequences in less than 10 seconds.

3.2 Developments of the SWIM Simulator Environment

Per Hammarlund
SANS, KTH

One goal of the SWIM simulator project is to create a simulator environment that can run on a wide variety of hardware platforms. Our simulations are quite large and we anticipate that they will become even larger. Still, there is the need of the researcher to be able to simulate parts of or a down-scaled version of the network. The large simulations must go on whatever supercomputer the researcher can get time on and the small ones are most conveniently performed on the workstation on the researcher's desk.

The SWIM program has been updated to run efficiently on single- and multi-CPU workstations, using threading, and on single- or multi-CPU CRAY vector computers, using tasking. To be able to incorporate even more hardware architectures like parallel SIMD computers (CM2) and loosely connected parallel cluster-like computers (IBM SP-2), the simulator engine of SWIM has been pulled out and is being rewritten in C++. This new code is called SPLIT and will be available in the form a of a library. The functionality of the BIOSIM program [Levin *et al.*, 1990, Hammarlund *et al.*, 1991, Hammarlund *et al.*, 1992b, Hammarlund *et al.*, 1992a] for the CM2 is now being integrated into the SPLIT library. The SWIM simulator will use the SPLIT library at a low level.

3.3 Modeling Cortical Associative Memory

Erik Fransén, Anders Lansner
SANS, KTH

Here we describe a simplified model of the associative memory function of the cortex. Our interest is to see how the connectiv-

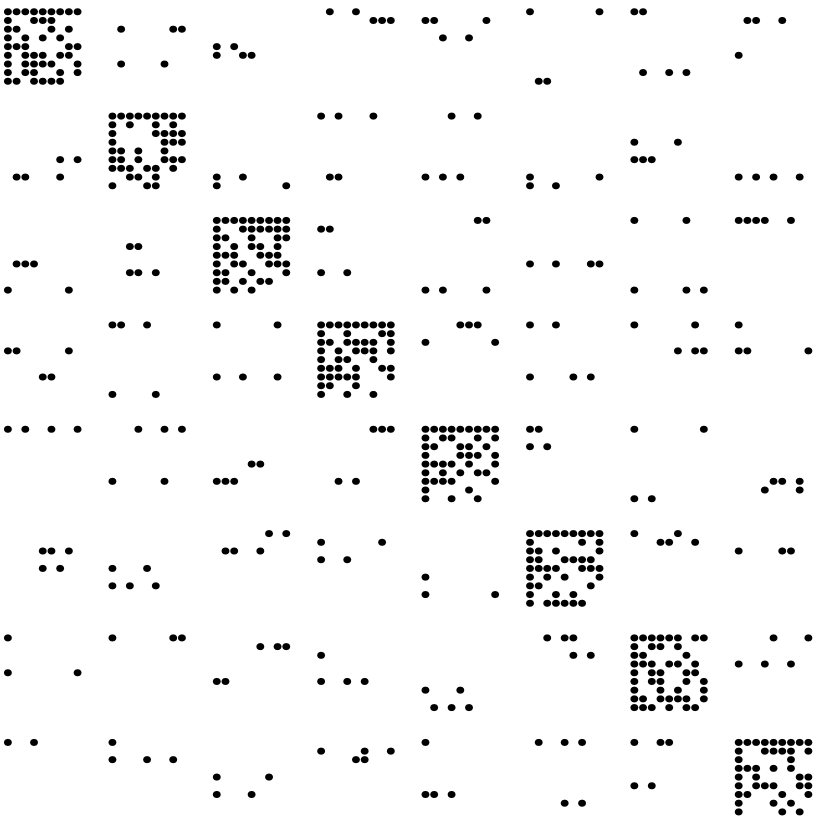


Figure 3.1. This figure shows an example of the connectivity used in the cortical associative memory simulations. In this figure there is a dot in a position ij if there is a connection between two neurons i and j. Here there are 8 mini-columns with 12 strongly connected neurons, hence the dense squares along the diagonal. The connectivity between the mini-columns is more sparse, as can be seen in the less dense off-diagonal areas.

ity structure of a recurrent artificial neural network (ANN) of the Hopfield type can be arranged in order to get a more cortical-like structure. The functional unit in our model, corresponding to an ANN unit, is a cortical mini-column. Inside the mini-column connectivity is dense and both excitatory and inhibitory, whereas the long-range inter-columnar connectivity is sparse and only excitatory. By this arrangement the unbiological constraint of full and symmetric connectivity is avoided, since, at the cell-to-cell level connectivity becomes asymmetric and sparse on the average, see Figure 3.1.

In this study we have used the general purpose simulator, SWIM, intended for numerical simulation of networks of biologically realistic model neurons [Ekeberg *et al.*, 1993]. The model neurons may be composed of an arbitrary number of iso-potential compartments. Voltage dependent ion channels are modeled using Hodgkin-Huxley-like equations. Parameter values used here are

given in [Fransén and Lansner, 1994, Lansner and Fransén, 1994]. The model is currently extended and implemented using the SPLIT-library, see Section 3.2.

The entire network consists of 750 cells arranged in 50 mini-columns. The connection matrix was created using an ANN and then incorporated into our network model. Eight different patterns like the one in the margin, each composed of 8 out of 50 columns active, were stored in the network by a simple correlation based (Hebbian) learning rule [Lansner and Ekeberg, 1989b]. Patterns/assemblies are overlapping, i.e. sharing some common columns. Connections are excitatory between columns that occur together in one or more of the patterns stored, and inhibitory between those columns that never occur together in any of the patterns. Each column is composed of 12 pyramidal cells and 3 inhibitory interneurons. Within a column the pyramidal cells connect to each other by 70 randomly chosen synapses. The inhibitory interneurons each make a contact to 8 pyramidal cells. The excitatory long range (inter-columnar) connectivity is from 2 pyramidal cells in the sending column to 6 pyramidal cells in the receiving column. Inhibitory connectivity is from 6 pyramidal cells in the sending column to 3 inhibitory interneurons in the receiving column. The total number of synapses in the network was about 18,000. (See Figure 2.13 on page 18.)

```
o*o*ooo
oooo**o
ooooooo
ooooooo
oooo**o
ooooooo
ooo*ooo
*
```

The aim here was first to demonstrate that a mini-column was capable to act as a functional unit. As an example, if eight pyramidal cells in the same column were stimulated they could activate their companions in some 10–30 ms. There was little tendency for after-activity to occur. Next, when one assembly was stimulated and the application of a neuromodulator like serotonin is simulated [Fransén *et al.*, 1993], the activity in an active assembly will persist. The after-activity is a result of a reduced adaptation of the cells. The firing rate in the after-activity period was relatively low, about 70 Hz due to the saturating conductance of the synaptic model [Fransén and Lansner, 1994]. With the summing model commonly used the frequency would have been 150–200 Hz. Further, when 6/8 of the columns in one assembly (pattern) are stimulated the remaining 2 will get activated (pattern completion) by the inter-columnar excitatory connections. A few randomly activated columns will be suppressed (noise tolerance). This process is relatively fast, a complete pattern is normally obtained in 40–

100 ms, comparable to that found in psychophysical reaction time experiments. An activated assembly will not produce any spurious secondary activation of columns in other assemblies despite the overlap. The spread of activation is prohibited by the lateral inhibition between assemblies. This overlap is essential for having an acceptable storage capacity. Finally, when parts of two assemblies are stimulated (ambiguous input) they will compete. When one of the assemblies wins it will complete its pattern and shut down the activity in the other one.

3.4 Simulating the Complex Dynamics of a Brain Structure

Tomas Wilhelmsson, Hans Liljenström
SANS, KTH

A model of the olfactory cortex, the "odor brain", has been implemented on the CM200 [Wilhelmsson, 1994]. The intention is primarily to investigate the biological significance of the complex dynamics found in certain brain structures. In particular, its role in perception, learning, and memory. A second goal is to study how such a neural network model can be efficiently implemented on SIMD computers for more realistic simulations. Much larger and more detailed systems can be simulated on the CM200 than on conventional workstations. Also several sub-systems can be modeled and linked to form a functionally more complete system. This will allow for a more realistic processing of (sensory) input, associating it with any stored memory, and giving an appropriate output that eventually could be used also for technical applications. The much higher processing speed that this parallel implementation provides, make it possible to run large simulations in minutes instead of hours, which is crucial when tuning model parameters.

The olfactory cortex is used as a model system because of its structural and dynamical properties. In comparison with many other brain structures, like the visual cortex, the olfactory cortex is relatively simple. Its three-layered structure with extensive interconnections distributed within the main layer resembles to a high degree the architecture of artificial neural networks used for associative memory. It has also a well characterized dynamics, including chaos-like behavior and oscillations in the 5 and 40 Hz

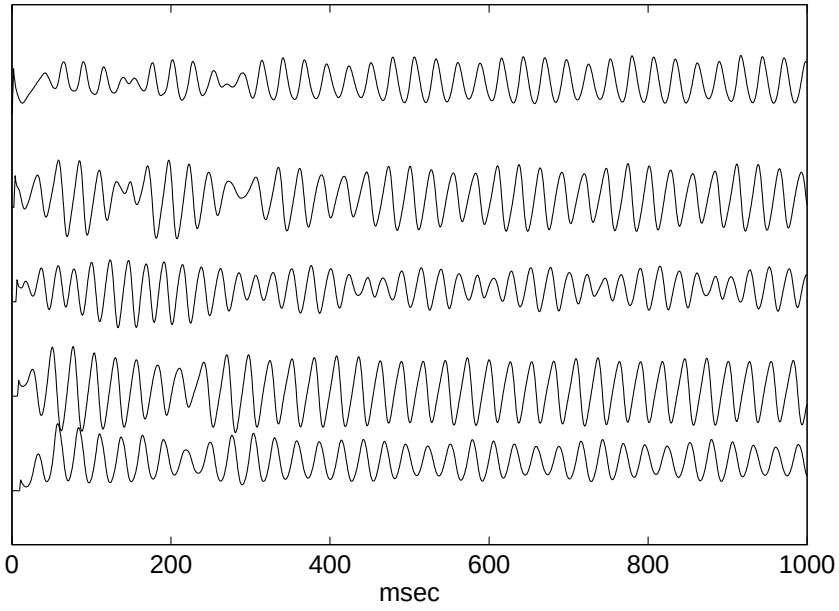


Figure 3.2. An example of how five excitatory network units may respond to a weak input pulse given to the system. The normalized activity of the five units is shown for one simulated second, with the 2 ms input pulse applied at time $t = 0$. There are two dominant frequency components in these oscillations, corresponding to the theta rhythm (around 5 Hz) and gamma rhythm (around 40 Hz) of the olfactory cortex. The total number of network units used in this simulation is $3 \times 64 \times 64$, which required 30 minutes on the 8K CM200.

range. The odor information is thought to be encoded in the rich and varying spatio-temporal activity patterns that are recorded from this area in behaving animals.

Model description

An artificial neural network consists of network units communicating via synaptic connections. Each unit sums the weighted activity levels it receives from connected units, using the sum to update its own activity level. The complexity of the current model network lies between that of abstract Hopfield nets and more biologically detailed simulations with spiking, compartmentized network units. The network units used here represent populations of cells (neurons), with an activity level given by the "mean cell membrane potential". The input-output relation is a continuous (sigmoid) function, and the network connectivity closely resembles that of the real cortex.

The model has a three layered rectangular structure, where one layer of excitatory units (pyramidal cells) is connected to two layers of inhibitory units (interneurons) on either side. Connections within a layer only exist for the excitatory units. There are both short and long range (asymmetric) connections, which are modeled with distance-dependent delays.

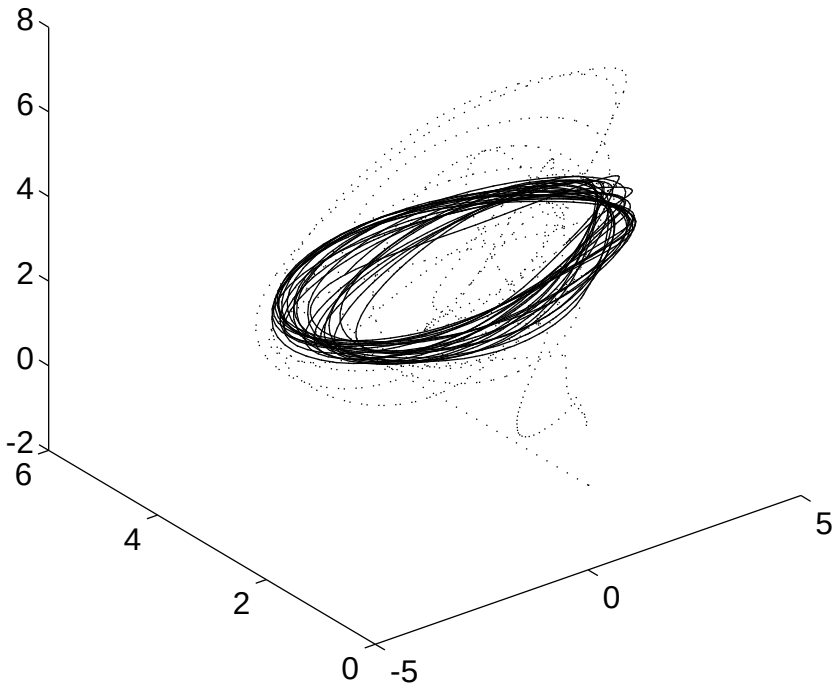


Figure 3.3. Phase space plot showing the activity of three excitatory units plotted against each other for one simulated second. The trajectory for the initial 300 ms is dotted to highlight the convergence to a limit cycle attractor for the remaining 700 ms.

External input is projected to the upper two layers (feedforward inhibitory and excitatory units) and simulated as incoming fibers from the olfactory bulb spreading out radially from one corner of the network. Network parameters, like time and space constants, as well as signal propagation speeds, are chosen within their physiological ranges. With the parameter values used in most cases, we simulate a 10×10 mm square area of the olfactory cortex of a rat. Network connection weights are adaptive to allow for learning and associative memory to be studied. We are currently using a modified Hebbian learning rule for weight updating.

Simulation results

More than 70 million interconnections

Earlier simulations, using workstations, have shown that this model is capable of reproducing the essential characteristics of experimentally observed and previously simulated dynamic behavior of the olfactory cortex [Liljenström, 1991]. These earlier simulations were made with 10×10 network units in each layer. On the 8K CM200, with 1 GByte main memory, we are able to simulate systems with up to 96×96 units in each of the three layers, with

more than 70 million interconnections. The dynamics of the network is exemplified in Figures 3.4 and 3.4 and also in Figures 2 and 2 on pages 12 and 13 showing different aspects of the same one-second simulation with 64×64 units.

Performance

The table below show simulation time per iteration for some layer sizes for the 8K CM200 as well as a modern workstation, the SPARC-10/51.

Layer size	SPARC-10/51	8K CM200	
	Time (ms)	Time (ms)	CM Busy (%)
10×10	62	362	55
16×16	326	326	52
32×32	4742	350	59
64×64	76000[a]	1285	97
96×64	171000[a]	2306	99

[a]For the workstation, the times for layers larger than 32×32 are extrapolated from the 32×32 case since there was not enough memory in the workstation for larger systems.

The CM200 outperforms the workstation by a factor of 60 or more

Note that the elapsed time per iteration is approximately the same for all of the three smaller sizes, where overhead dominates due to array padding. At a layer size of 64×64 units we achieve a sustained performance of 83 MFlop/s, i.e. 41.5 million connections per second. For the larger layer sizes, the CM200 outperforms the workstation by a factor of 60 or more.

About 80% of the elapsed time is spent summing up the input from all connections arriving at each network unit. The table below shows relative time and floating point performance for the different steps when doing the summation. The timings are for the 64×64 case.

Summation step	Relative time	MFlop/s
Get the delayed activity	37	
Multiply by weight and add to derivative	17	490
Shift the derivative one step	36	
Remainder, not timed	10	
Total	100	83

To get the delayed activity involves indirect addressing, which is an expensive operation on the CM200. In our case it takes as much time as the shifting. The calculation time is only 17% of the total time.

Connection Machine implementation

The workstation implementation was rewritten in CM Fortran to take advantage of the slicewise programming model of the CM200. For efficiency, we have used the CMSSL subroutine PSHIFT for doing four CSHIFTs at a time, and the subroutine CMF_aref_1d for the indirect addressing. The CM graphical debugger Prism has been very useful when analyzing and debugging the program.

The connection probability within the excitatory layer is specified to 10–50% in the simulations. We have implemented both sparse and full matrix storage of these connections. In the sparse storage case, grid or general communication is selected at runtime. Grid communications usually turns out to be faster than general communication, even when the connection probability is below 20%. This is due to the high efficiency of the PSHIFT subroutine. The sparse matrix storage also allows the 96×96 layer size to fit within the available memory.

Future work

We plan to utilize the gain in speed and storage provided by the CM200 for increasing the biological realism in our simulations. This includes modeling several interconnected subsystems simultaneously, such as the olfactory bulb and cortex represented by separate neural networks, with both forward and back projections. We can also include more structural details in network units and connections. The spatial resolution we obtain with these denser networks allow for more realistic dynamics with local variability. We can further simulate much longer time periods, which allows for a better comparison with experimental EEG and PET data.

4 Computational Fluid Dynamics

Computational Fluid Dynamics (CFD) is the art and science of computational modeling of fluid-flow phenomena under various physical conditions. The mathematical models are variants of the non-linear Navier–Stokes equations which can have very complicated solutions. Numerical simulations of these problems are therefore a necessary complement to the theoretical and experimental analyses. CFD is also pursued as an engineering discipline because it leads to better products and to savings in the design process.

Make progress by constructing computationally efficient methods and by exploring the possibilities offered by massively parallel computers

Most CFD simulations today are made with discretizations which are not sufficient to completely resolve all interesting physical scales. What is lacking is both computational power and physical memory size. However, it is still possible to make progress by constructing more efficient numerical methods and by exploring the new possibilities offered by massively parallel computers.

The CM is routinely used for both pilot studies and production runs

The CM is now routinely used for both pilot studies of a problem and for production runs with programs already developed. The projects under way in the CFD area explore several different methodological directions: high-resolution finite difference methods (Section 4.1), adaptive finite-element methods (Section 4.2), spectral methods (Section 4.4), (Section 4.6), and finite-volume methods (Section 4.5).

The applications include weather forecasting (Section 4.6), detonation wave simulation (Section 4.1), and aerodynamics (Section 4.5). Since the numerical techniques are very similar, we include the application on electromagnetic fields (Section 4.3) in this section.

The CFD projects have also inspired international contacts with groups working at EPFL, Switzerland and CERFACS, France.

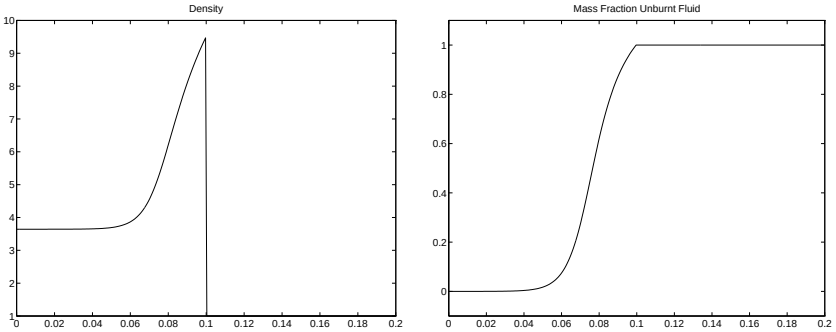


Figure 4.1. Mass fraction (right) and density of fuel (left).

4.1 Numerical Computation of Detonation Waves

Björn Sjögreen
TDB, Uppsala University

Consider a shock wave traveling into a flammable fluid. The shock wave increases the temperature, which triggers a chemical reaction. Thus such a *detonation wave* consists of a shock wave followed by a thin reaction zone immediately behind the shock. In Figure 4.1 (left) we show the density of such a wave profile in one space dimension. The wave travels into the unburnt state to the right. Figure 4.1 (right) shows the mass fraction of unburnt fluid.

We solve the equations of reactive compressible fluid flow, using a shock capturing finite difference method. The thin reaction zone requires very high resolution in the numerical method for solving the problem. This means that the number of grid points becomes very large, especially in two and three space dimensions. It is then necessary to use a powerful computer, such as the CM200.

In Figure 4.2 we show iso density contour lines of a detonation wave in two space dimensions at six different times. The wave propagates to the right, and transverse waves develop along the front. Two triple points which bounce up and down against the upper and lower walls are clearly seen.

The computation in Figure 4.2 was made using a high-resolution second order accurate difference method in space, and a second order Runge-Kutta integration in time. We used 256×128 grid points, the entire computation took about 20 minutes on 4K processors and the overall computational speed was 86 MFlop/s. All computations were made using double precision. If we do the same computation as in Figure 4.2, but instead use a less accurate first order difference method in space, we obtain the result in

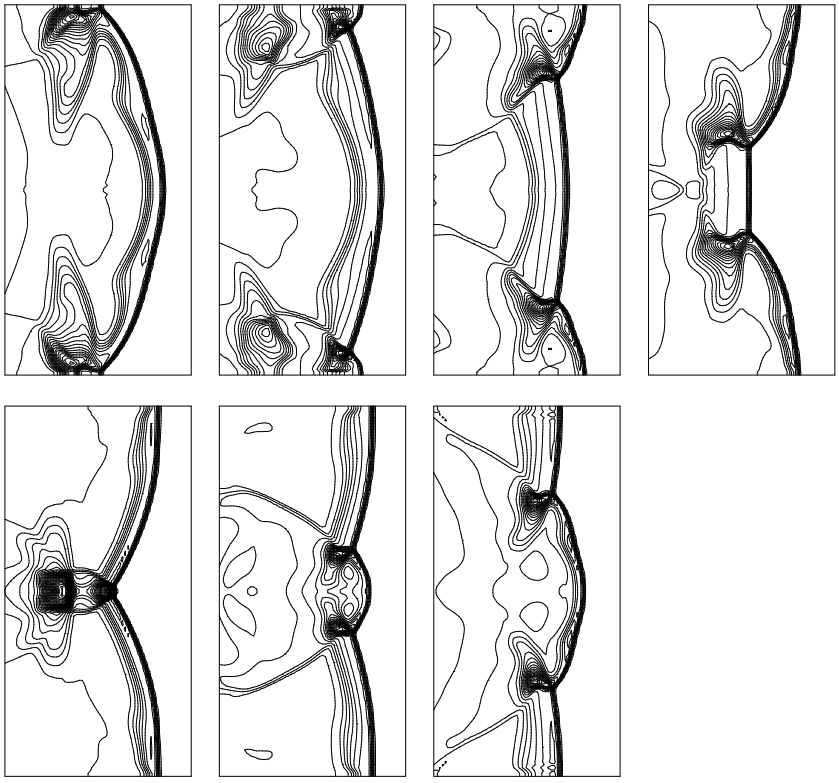


Figure 4.2. Iso density contours, second order accuracy, show the detonation front moving left to right with correct reflections at walls top and bottom. The sequence starts at the upper left figure.

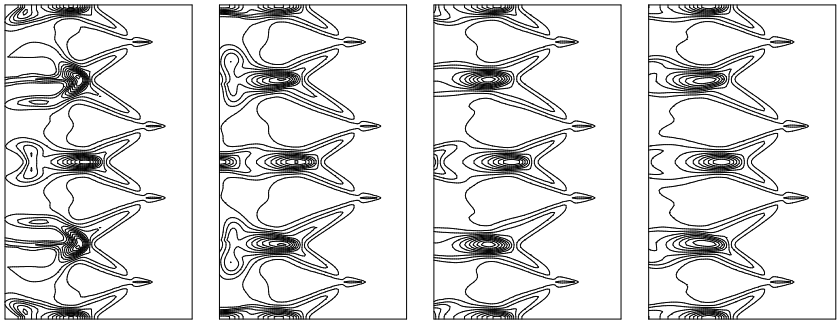


Figure 4.3. Iso density contours, first order accuracy, show erroneous solution due to excessive numerical dissipation.

Figure 4.3. This result is not correct, due to insufficient accuracy in the reaction zone. The triple points becomes stationary, and a steady wave structure develops. The four different solutions in Figure 4.3 correspond to the first four times in Figure 4.2. One of the main interests in this project is to study the influence of such numerical effects.

The performance of the CM200 on the present code was measured with the result shown in Figure 4.4. All results were computed in double precision on 4K processors. The language was CM Fortran and the compiler option -O was used. The code contains some shift operations, but is dominated by arithmetic work.

We see that a maximum rate of about 100 MFlop/s is reached if the problem is large enough. The number of grid points on the x axis is normalized by the number of physical processors. It seems that it is necessary to have this VP ratio above 16 in order to obtain maximum performance.

This project will develop along two lines in the future. We will implement a realistic reaction mechanism, with several chemical species, and we will extend the code to three space dimensions.

4.2 Adaptive Finite-Element Methods

Claes Johnson, Peter Hansbo, Kenneth Eriksson
Department of Mathematics, CTH

The main objective of our work is to investigate principles for the implementation of adaptive finite element methods on massively parallel architectures such as the Connection Machine (CM). Specifically, we have considered the implementation of the streamline diffusion (SD) method, which is a general finite element method for hyperbolic type problems such as convection-diffusion problems

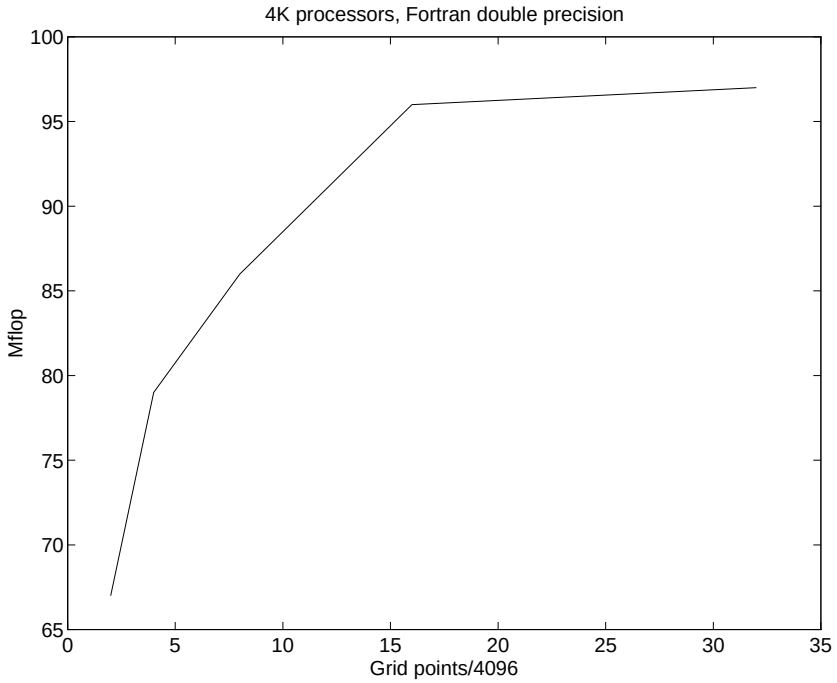


Figure 4.4. MFlop/s plotted against problem size.

and the Euler and Navier–Stokes equations of incompressible and compressible flow.

We are currently focusing on three-dimensional incompressible flow computations with or without free boundaries. For this purpose, we have implemented a 3D mesh generator, a 3D h-refinement algorithm, and a multi-grid solution algorithm taking advantage of the nested sequence of grids occurring in standard nested h-refinement. See Figure 4.5 for an example.

The SD-method we use is a modified Galerkin method based on space-time finite element discretization with piecewise polynomial basis functions, discontinuous in time and continuous in space. By orienting the mesh in space-time locally according to particle trajectories, we obtain methods with the essential features of so called 'particle methods' or Lagrangean methods. This yields a way of generalizing particle methods to include diffusive effects, impose boundary conditions and handle mesh distortion problems.

We are currently implementing our codes on the CM-system. The new CMSSL library supports new powerful tools for mesh partitions and communication routes. These new tools significantly strengthen the prospects of unstructured methods on parallel pro-

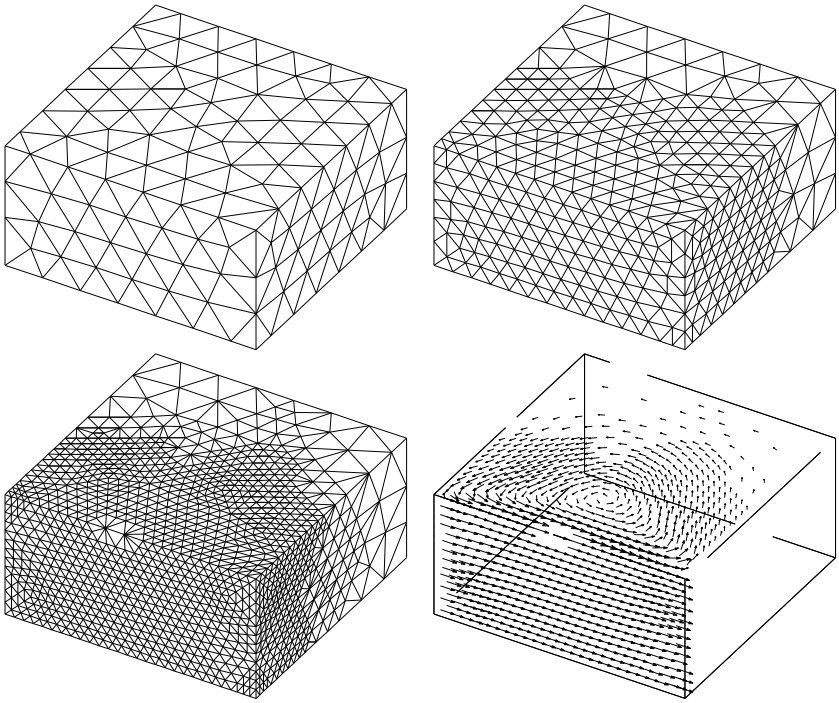


Figure 4.5. A sequence of adapted 3D grids for the solution of Stokesian closed cavity flow, with velocity solution on the finest grid (lower right).

cessors, and we aim to investigate this possibility. In order to be able to use these tools in our particular framework of adaptive methods, we have initiated a cooperation with one of the prime architects of the communications part of CMSSL, Zdenek Johan of Thinking Machines Corporation.

4.3 Computational Electromagnetics in 2D

Ulf Andersson, Gunnar Ledfelt
C2M2, KTH

The project develops techniques for computing electromagnetic fields by time integration of the Maxwell equations. The standard tools of today are frequency-domain based methods, in particular the Method of Moments (MOM). Frequency-domain based methods give the response for all angles of incidence at a single frequency. Time-domain based methods, on the other hand, provide solutions for many frequencies from one single transient calculation. Using spectral methods, the time-domain transient solutions can be processed to provide the frequency-domain response.

Time integration with explicit methods is well suited to massively data-parallel computers, and indeed one of the pilot codes runs at almost 50% peak speed of the CM.

Another benefit of time-domain techniques in Computational Electromagnetics (CEM) is that they can take advantage of the software technology developed in Computational Fluid Dynamics (CFD). For a transverse electric field in an isotropic, resistive medium the Maxwell equations can be written as

$$\begin{aligned}
\frac{\partial}{\partial t}E_x &= \frac{1}{\epsilon}\frac{\partial}{\partial y}H_z \quad - \frac{\sigma}{\epsilon}E_x \\
\frac{\partial}{\partial t}E_y &= \frac{-1}{\epsilon}\frac{\partial}{\partial x}H_z - \frac{\sigma}{\epsilon}E_y \\
\frac{\partial}{\partial t}H_z &= \frac{-1}{\mu}\frac{\partial}{\partial x}E_y + \frac{1}{\mu}\frac{\partial}{\partial y}E_x,
\end{aligned} \tag{4.1}$$

where $\bar{E} = (E_x, E_y, 0)^T$ is the electric field, $\bar{H} = (0, 0, H_z)^T$ is the magnetic field, σ is the conductivity, ϵ is the permittivity and μ is the permeability. This is a hyperbolic system just like the Euler equations of gas dynamics used in CFD. It is therefore possible to take algorithms developed for the Euler equations and use them to solve (4.1).

We have implemented a time domain based method for solving (4.1) on both the CM200 and the MP-1 by porting a existing scalar program [Gradin and Ledfelt, 1993]. We use a Finite Volume method based on central differences and the time discretization is made with a three stage Runge-Kutta method. This is second order accurate in both space and time for regular grids. Both code versions support periodic boundary conditions and boundaries at perfectly conducting surfaces. The MasPar version also supports absorbing boundary conditions.

A substantial effort was made to optimize the CMversion of the code and in single precision on an 8K CM200 it runs at 1.3 GFlop/s. This has been achieved by using the stencil compiler which imposes some rules on the code syntax meaning that around 25% redundant FLOPs must be introduced. FLOP-rates are taken from calculations with 2048×2048 cells which is the maximum problem size that fits into the 1Gbyte primary memory of the CM200. For every test case so far single precision has proven to be sufficient.

In the MasPar version of the code an artificial far-field non-reflecting boundary is included. This makes it possible to treat open problems, i.e. where the physical domain is unlimited. The performance of the absorbing boundary condition is critical. Otherwise reflections may occur which can destroy the solution. The absorbing boundary condition used is described in [Gradin and Ledfelt, 1993]. This was implemented in such a way that the main calculations are done the same way throughout the whole computational domain with a minimum of special treatment at the boundary. For a problem-size of 1024×128 the MasPar version runs at 370 MFlop/s in single precision.

Figure 2.1 on page 2.1 shows an electromagnetic wave that passes a infinitely long, perfectly conducting cylinder. The calculations have been made on the MasPar using a O-grid with 1024×128 cells. It is clear that the absorbing boundary condition works well.

4.4 Simulation of Turbulent Couette Flow on the CM200

Jukka Helin
C2M2, KTH
Anders Lundbladh
FFA, Bromma
Arne Johansson
Department of Mechanics, KTH

We have simulated turbulent plane Couette flow at low Reynolds number, using the CM200. The code used in this simulation was originally developed to run on vector machines [Lundbladh *et al.*, 1992]. It was ported to the CM200 during the autumn of 1993.

The code uses a spectral integration method, with FFTs in x and z directions, and a Chebyshev transform in the inhomogenous y direction. For the FFTs we use the CMSSL library while the Chebyshev transform is coded in CM Fortran. At each iteration of the main loop of the program there are a number of FFTs and Chebyshev transforms, making it a very communication intensive code. Despite this we have managed to get a performance of about 240 MFlop/s in double precision arithmetic on an 8K CM200. Compared with the vector version of the program run on one head of a CRAY X-MP, the parallel version is about 3 times as fast, and compared to one head of a CRAY Y-MP it is 30% faster.

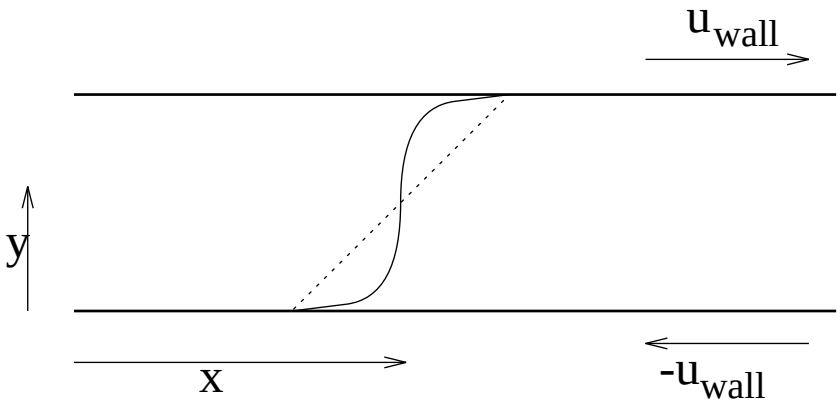


Figure 4.6. The mean velocity profile in plane Couette flow for laminar (dashed line) and turbulent (solid line) flow.

In Figure 4.6 the dashed line is the laminar velocity profile, and the solid line is the turbulent velocity profile. In laminar Couette flow, both the instantaneous and the time averaged velocity at the center line is zero. In the turbulent case the time averaged velocity is zero, but the instantaneous velocity is not zero. A picture of the streamwise velocity field in the x-z plane, can be seen in Figure 2.10 on page 2.10. It shows the instantaneous velocity field at the center line with the deviation from zero velocity color coded in read and blue, for positive and negative deviations respectively. Note the large-scale structures in the flow, which have long length-scales in the streamwise direction.

The main objective for this project is to study these large-scale structures, and we have therefore used a large computational domain, about $88 \times 2 \times 25$ measured in half channel heights. One important conclusion drawn from this study is that the large scales do not have infinitely large length-scales in the streamwise direction as some previous investigations indicated [Lee and Kim, 1991].

In these simulations it is an advantage to have a large primary memory because of the large number of grid points necessary to resolve the flow. The present low Reynolds number simulation has over 7 million grid points making use of almost all the available memory of 1 GByte. Simulations of flows at higher Reynolds number, or larger computational domain, would require even more grid points, and thus more memory.

4.5 Data-parallel Multi-block Flow Computations

Mark Sawley, Jon Tegnér
IMHEF, EPFL, Switzerland

To compute flows in complex geometries, block structured meshes provide an alternative to the use of unstructured meshes. Parallel computations using block structured meshes can exploit either a coarse-grain parallelism at the block level or a fine-grain parallelism at the mesh point level. For efficient parallel computation, the granularity of the problem needs to be matched to that of the parallel computer employed.

Coarse-grain parallelism has been exploited in CFD calculations by a number of researchers, using either shared-memory multi-processor computers or distributed-memory massively parallel MIMD computers. These implementations have employed a control parallel methodology, for which different blocks are computed in an independent manner on different processors with information transferred between blocks whenever appropriate [Sawley, 1993]. While control parallel multi-block methods are straightforward to employ if the mesh is comprised of blocks having equal computational work, in general the problem of load balancing between processors can become a major concern.

The fine-grain parallelism exhibited at the mesh point level of a structured mesh is well suited to the data-parallel programming model, for which different processors undertake the necessary computations of different mesh points in a synchronous manner. To date, the application of the data-parallel programming model to CFD appears to have been limited to computations based on single-block structured meshes and unstructured meshes [Sawley, 1993, Sawley and Bergman, 1994].

The present study investigates a serial data-parallel multi-block method that retains much of the simplicity of the data-parallel approach. Individual blocks are treated in a sequentially, the solution in each block being computed using a data-parallel approach. Such a method can be directly applied, for example, to both the CM200 and MP-1 computer systems.

The serial data-parallel multi-block method has been implemented in a code, originally developed by Magnus Bergman, that solves the time-dependent Euler equations for inviscid, compressible flow in 2D geometries. The equations are discretized in space

using a finite volume formulation, using central differencing with added artificial dissipation. An explicit five-stage Runge-Kutta scheme is used to perform the time integration. To enhance convergence to the required steady state, a local time stepping technique is employed. The values of the flow quantities throughout the flow domain are stored in global arrays. For the serial data-parallel multi-block method, the global values for one block are copied to corresponding local arrays, which contain an additional exterior layer of ghost cells to provide data locality and facilitate the application of boundary conditions. The computation is performed using the local arrays in a manner similar to that for a single block mesh [Sawley and Bergman, 1994]. The updated local array values are then returned to the global arrays. The transfer of data between different blocks (block connectivity) is achieved via the global arrays in an implicit fashion using the globally addressable memory. After the global array elements for one block are updated, the other blocks are treated in a sequential manner. See [Sawley *et al.*, 1993a] for more details.

As an application, we consider here the flow in a supersonic air intake. A freestream Mach number of 1.865 and a back pressure (static pressure at diffuser exit / freestream total pressure) equal to 0.83 has been imposed. The resulting flow, as shown in Figure 2.2 on page 2.2, is characterized by an oblique shock that impinges on the cowl lip, followed by a series of reflected shocks in the diffuser section. This flow case has been studied by the French aircraft engine manufacturer Snecma in the development of a power-plant for the replacement of the Concorde supersonic aircraft.

To compute the flow field in the air intake, a block structured mesh of eight blocks has been employed. (See Figure 2.2 on page 2.2.) Note that this mesh has been constructed from geometric considerations alone. Smaller blocks are allocated to regions of greater interest, the total number being chosen as suitable to cover the flow domain. The number of mesh cells in each block is the same (125×125). This choice has not been based on parallelization considerations, but solely on convenience, since the algorithm employed in the code is restricted to directly-connected blocks. While a control-parallel method could be efficiently used for such a mesh on a computer system comprised of eight processors of equal performance, additional artificial sub-division of the mesh would be required for more general parallel systems.

Since the blocks are treated in a sequential manner, a rather simple dynamic block management strategy has been employed. Every tenth iteration, the values of the residuals in each block (based on the local density) are calculated. The residual values are used to determine which of the blocks are to be considered (i.e. "switched on") for the next ten iterations. The criterion for a block to be switched on is that the maximum residual in the block, or at the appropriate ghost cells in neighboring blocks, is greater than the pre-determined convergence value. In this manner, the computation is performed only in blocks where there is useful work to be done. Not only are blocks switched off to avoid unnecessary computations, but they are also switched back on should the development of the solution in a neighboring block produce an influence. After the maximum residual in all blocks has fallen below the desired value, further iterations are performed on all blocks to avoid accumulative effects. For the present problem and for the majority of iterations, it has been necessary to compute on at most four of the eight blocks. The use of the above-described dynamic block management, has therefore been found to diminish the work undertaken – and thus the CPU time – by a factor of two.

Code portability and performance are also being investigated in the present study. Adhering to the Fortran 90 standard has enabled the code to be run on serial, vector/parallel and massively parallel systems [Sawley *et al.*, 1993b]. However, it does not guarantee that the performance obtained on each of these platforms will be acceptable. Indeed, detailed information regarding the target computer hardware and compiler implementation is necessary to optimize code performance. On distributed memory parallel systems, the communication time required for imposing the boundary conditions and the transfer of data between different blocks is critical in determining code performance. Such communication must be performed using the fastest possible means. The MasPar Fortran compiler interprets the array statements associated with these communication tasks to be performed using the fast X-Net of the MP-1. Since the CM Fortran compiler does not recognize that these statements can be performed using NEWS communications, recoding with the explicit use of CSHIFT intrinsic functions was necessary in order to optimize performance using the CM200. The above described flow problem has been computed, using 64-bit precision, on both the MP-1 and CM200 sys-

tems at PDC. A total CPU time of 5600 seconds was required on the 16K MP-1 (171.7 MFlop/s) and 14,170 seconds on the 8K CM200 (67.8 MFlop/s). Since a substantial amount of communication is required for the flow computation, these performance figures reflect the significantly higher ratio of communication to computational speed of the MP-1 system. Higher performance could be expected from both computer systems by using library routines, but at the expense of portability.

4.6 High Resolution Numerical Weather Prediction

Nils Gustafsson
SMHI, Norrköping
Tomas Wilhelmsson
NADA, KTH

More than 70 years ago, Lewis Fry Richardson calculated the first numerical weather forecast. It took him six years of hand calculations to do a three hour forecast. The forecast predicted winds with speeds of more than one hundred meters per second blowing in the wrong direction! In spite of this initial failure, due to inaccuracy of his initial data, Richardson was correct in his basic approach. He envisioned a massively parallel human computer, with 64,000 people doing the calculations of operational weather forecasting. Today massively parallel computers with 16,000 processors are available, bringing Richardson's vision to reality in a way he probably never imagined. The metereological community should try to use these new machines.

The HIRLAM weather forecasting system has been developed within a common research project among the the Nordic countries, Ireland and the Netherlands. Present operational versions can describe scales of motion of the order of 100 km while there is a need to move toward scales of the order of a few kilometers. The computer power for such resolutions, besides the general parallel structure of the problem, makes it interesting to try HIRLAM on massively parallel architectures. Two versions of HIRLAM are available, a gridpoint version and a spectral version [Gustafsson, 1991] based on the spectral transform technique [Orzag, 1970]. Both versions have been implemented and tested on the MasPar MP-1 at PDC as well as on the MasPar MP-2 in Bergen.

The MasPar version of the spectral HIRLAM

For most of the code it was possible to use the VAST-2 source code translator for an automatic conversion.

The spectral HIRLAM was originally coded in Fortran 77 and had to be converted to High Performance Fortran in order to run on the MasPar. For most of the code it was possible to use the VAST-2 source code translator for an automatic conversion. With regard to the core of the spectral dynamics, code conversion was carried out in a semi-automatic way by simple editing commands.

An efficient FFT package is a prerequisite for the application of the spectral transform technique. Two different FFT packages were investigated – the FFT package in the MasPar Math Library and an FFT package developed by Hans Munthe-Kaas [Munthe-Kaas, 1993] at the University of Bergen. The first trials with the Munthe-Kaas package resulted in a two-fold speedup of the complete model as compared to the use of the single transform MasPar Math Library routine.

Benchmark Results

Operational data sets from the SMHI were used for the benchmarking. The integration domain consists of 110×100 horizontal gridpoints and 16 vertical levels. In order to have periodic variations in both horizontal directions, this domain was extended in the horizontal to 128×128 gridpoints. The total elapsed computing times for HIRLAM spectral model runs are given in the table below.

Computer	Configuration		Time (s)
MasPar MP-1	16K processors	(32 bits)	106.9
MasPar MP-2	16K processors	(32 bits)	48.7
MasPar MP-2	4K processors	(32 bits)	200.6
Cray YMP C90	1 processor	(64 bits)	61.9
Cray YMP C90	8 processors	(64 bits)	8.5
CONVEX C3840	4 processors	(64 bits)	197.0
CONVEX C3840	4 processors	(32 bits)	123.0

Only the elapsed computing time for the pure forecast model integration was measured since the initial data handling and post-processing was not converted to HPF at the time of the measurement. Elapsed times for the MasPar are all for 32 bits arithmetic. Benchmark results for the HIRLAM spectral model on CRAY C90

and Convex C3840 have also been included in the table. The following of more general interest can be noted about the results: (1) The speedup factor to run the same forecast on 4 times as many MasPar processors is slightly greater than 4; (2) The MasPar MP-2 with 16K processors is about 2.5 times faster than the Convex C3840 in 32 bits arithmetic; (3) The MasPar MP-2 with 16K processors in 32 bits arithmetic runs 1.3 times faster than one CRAY C90 processor in 64 bits arithmetic. The measured average MFlop/s for the spectral HIRLAM on one CRAY C90 processor is 425. Thus, as a rough estimate, the MasPar MP-2 with 16K processors runs the spectral HIRLAM in 550 MFlop/s. This corresponds to 9% utlization of the peak performance. The corresponding figures for the MasPar MP-1 with 16K processors are 250 MFlop/s and 21% utilization of the peak performance.

The MasPar MP-2 in 32 bits arithmetic runs 1.3 times faster than one CRAY C90 processor in 64 bits arithmetic.

Recently also the pre- and post-processing parts of the HIRLAM forecast model have been converted into HPF. For a full 48 hour operational forecast run, pre- and post-processing accounts for 9% of the elapsed run time on the MP-1 and 17% on the MP-2. While the MP-2 runs the model integration about twice as fast as the MP-1, both machines have the same front-end. The front-end packs and unpacks the description sections of the metereological GRIB file format which causes the difference between the two MasPar models.

The Figure 2.12 on page 17 show a 48 hour forecast at 12 hour intervals. This forecast was done on MasPar MP-1 with 110×100 horizontal gridpoints (55 km grid distance) and with 16 vertical levels. The complete forecast runs in 20 minutes on the MP-1 and in 10 minutes on the MP-2. A low pressure system over Southern Scandinavia moves north, while decreasing in intensity, and another low pressure system is approaching western Europe from the Eastern Atlantic. The Figure 2.11 on page 16 show color coded jet winds from the same forecast.

Concluding Remarks

The HIRLAM spectral model including the complete physical parameterization package has been implemented successfully on the MasPar computer system. The model runs 2.5 faster on the 16K MasPar MP-2 system than on the 4 processor Convex C3840 currently in operation at SMHI (32 bits arithmetic). Considering the costs of the two computer systems, the MasPar system certainly

has proven to have a very competitive price-performance relation for this particular but very complex software package.

Contrary to what is generally believed and contrary to the present main stream of developments, this successful application of the complete and spectral HIRLAM model on the MasPar system has also proved that Massively Parallel Processing systems based on the SIMD concept are still very interesting alternatives to systems based on the MIMD concept. Also contrary to what is generally believed, the experiments indicate that spectral methods are competitive also on MPP platforms. Finally, the MasPar system has also proven to be capable of handling the I/O needed for a full operational forecast run.

5 Applications in Physics

The number of projects in this area is large; both those doing production runs and those that are in a development phase. Some of the projects study field problems by numerical solution of conservation laws such as the Poisson, Schrödinger or wave equations, for which the CM200 is ideal. It is interesting to note the many different types of studies, such as a model for 2D-turbulent flow made up of particle-like vortices in a quad-tree structure, and Monte Carlo (MC) simulations of spin glasses.

The ongoing projects come from a variety of fields: hierarchical models of turbulence (Section 5.2), quantum phenomena (Section 5.5, 5.3), disordered magnetic systems (Section 5.7), galactic and stellar evolution (Section 5.1, 5.4), Cellular Automata (Section 5.6), and Ising models (Section 5.8, 5.9).

5.1 Colliding Galaxies on the Connection Machine

Stefan Engström
Department of Astronomy, CTH

One popular explanation invoked to explain the spiral structure of disk galaxies is the gravitational tide incurred by a passing companion galaxy. Numerical simulations have substantiated this idea since the early sixties. Serious effort has been put into modeling actual observed systems with numerical models in which 10k–1M particles sample the collisionless system of approximately 10^{12} stars that constitute a disk galaxy. One particular famous example is that of M51 which was one of three models by the Toomre brothers [Toomre and Toomre, 1972]. This is one of the crucial papers on this subject which convinced many of the tidal perturbation scenario.

Toomre & Toomre's model was based on the optical image of M51 (Figure 5.1). It reproduces several independent features of the observations and has been the starting point for most subsequent attempts at refining the model. Prompted by later observational evidence in the form of observations of neutral hydrogen [Rots, 1990], in collaboration with Professor Athanassoula of

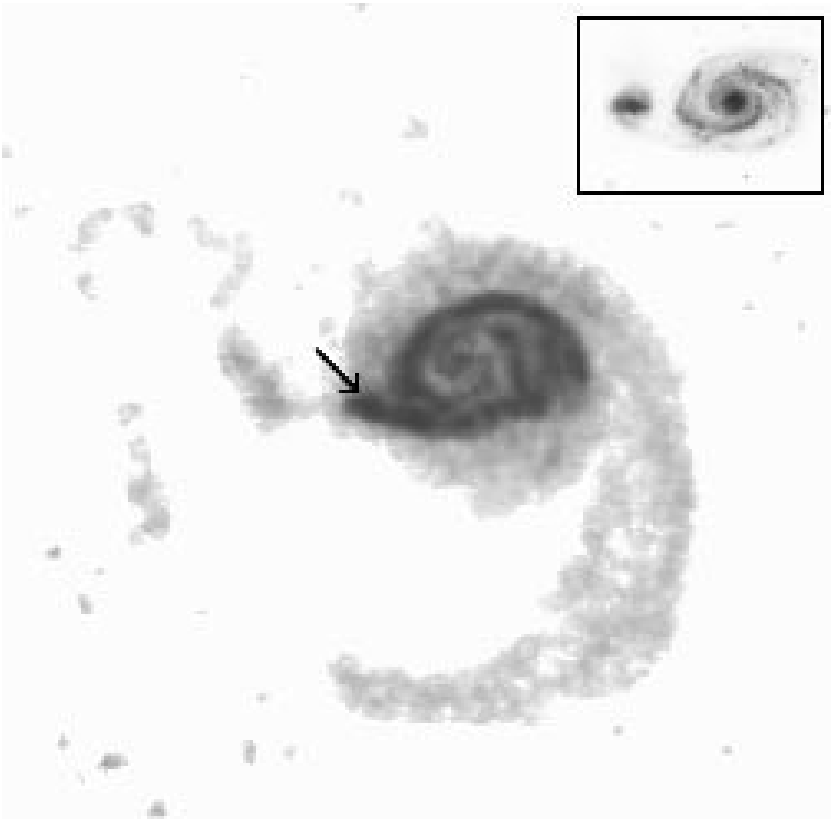


Figure 5.1. M51 column density in neutral hydrogen (HI) and in the optical waveband (inset). Note that the images have the same scale. The HI traces an entirely different set of scales.

the Observatoire de Marseille, I have opted for a different approach to the modeling procedure [Engström, 1993] .

The HI-data trace much larger scales than the optical frequency bands and constitute in a sense an independent observation of the same system, see Figure 5.1. Now we can see that the Toomre & Toomre model needs to be modified because the proposed orbital parameters do not reproduce the structures in the outer parts of the disk.

Making the minimal assumptions that: (a) the structure is mainly due to the passage of the companion galaxy, and (b) the disk was initially axisymmetric, we have investigated a reasonably complete orbital space in a model where the disk material is non-self gravitating. This last assumption is very reasonable for the HI-disk and streamlines the problem for attack by massively parallel computing. This project became possible due to a generous amount of time at the Connection Machines located at

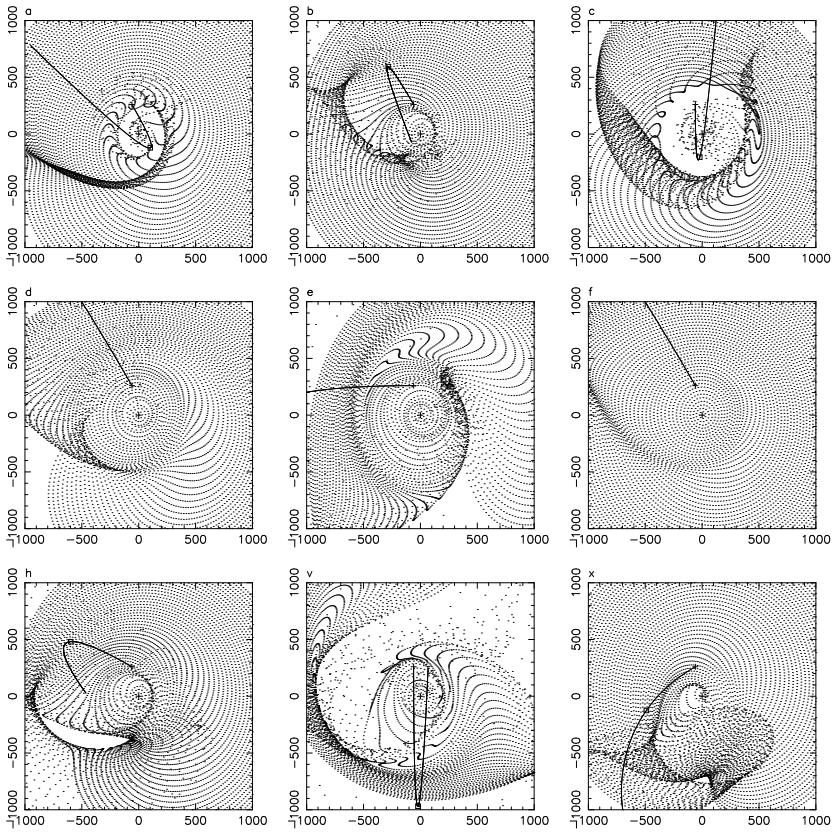


Figure 5.2. Examples of density morphology seen in the numerical simulations. The density is traced by particles which originally move on circular orbits which are perturbed by the passing companion.

INRIA (Sophia-Antipolis, France) and at PDC. On the order of 10,000 simulations with 8K particles integrated for several thousand timesteps have been calculated and the results have been compared to observations with respect to density and velocity fields.

It is possible to find families of solutions (Figures 5.2 and 5.3) that reproduce the gross structure seen in observations, however – none of these reproduce the velocity field. In fact, selecting solutions based solely on the velocity field of the large outer arm in Figure 5.1 produces no convincing solutions to the modeling problem. To fully specify a two-body interaction demands knowledge about three relative positions and three velocities in addition to the relative mass of the objects. Of these quantities, we can determine observationally only four (two positions in the plane of the sky, one velocity component along the line of sight and the relative mass of the systems). Three entirely free parame-

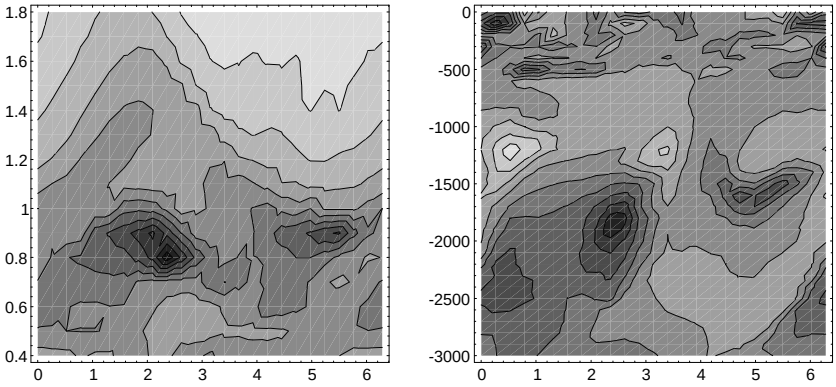


Figure 5.3. Extensive parameter surveys and automatic analysis of the data enable us to identify families of orbits which produce densities and velocity fields which have a certain level of correlation to the observed quantities.

ters remain. By physical arguments we have bounded these into intervals which we have carefully searched for matching density and velocity responses. Had we not been able to cover the entire three-dimensional parameter space, we would not be in the position to seriously cast doubt on the fundamental assumptions which is where we have to turn next to unravel the secrets of this particular galaxy.

5.2 Hierarchical Model of 2D Turbulence

Peter Frick, Vladislav Shaidurov
ICMM, Perm, Russia

The number of excited degrees of freedom in fully developed turbulence is so large that numerical investigations demand, and will continue to demand, more and more powerful computers. This motivates the construction of special models to investigate particular properties.

A model for 2D-turbulence has been constructed as a hierarchical tree of vortices of different sizes. In contrast to shell models, the number of degrees of freedom grows with wave number, as in real turbulence: modeling comes in only in the connections and interactions of the vortices. The functional form of the vortices are taken from a discrete, hierarchical basis, well localized in both real-space and Fourier-space. The vortices are connected along the back-bone of the tree, but not horizontally on a given level. The inertial forces conserve both energy and enstrophy. Statistical properties of the cascade are investigated, both integral and local. For details of the model see [Aurell *et al.*, 1994].

We have studied in detail the direct cascade of enstrophy in forced 2D-turbulence. Most simulations were done for 11 levels of scales (from $N = 0$ to $N = 10$, the total number of vortices being 1398101). The integration time per step was then around 30 seconds using the 8K CM200 at PDC. The maximal size of tree, that was run on the CM after very strong optimization was 12 levels (5592405 vortices). To do this we used the embedding of a hierarchical tree into a hypercube described in [Johnsson and Ho, 1989].

We have checked that different resolutions of the dissipative range does not change the results in the inertial range. The tests were performed by keeping the Reynolds number based on viscosity constant, but adding or subtracting one more level in the dissipative range. The add of one level for the same Reynolds number does not give any essential changes in the spectral distributions.

5.3 Scattering in Electron Waveguides

Thomas Palm, Jan-Olof Wesström
Microwave Engineering, KTH

One-dimensional electron waveguides have great potential for making fast electronic devices. Here electrons travel ballistically, i.e. without scattering in narrow channels, which increases the speed of the device. The small size also means that quantum mechanics is important, which leads to new possibilities such as interference effects. Given idealized models we have earlier shown good performance in modulation-doped, split-gate structures [Palm, 1993]. In particular we have studied a Y-branch switch.

The small size of these devices implies that individual impurity atoms can affect the performance. It is therefore only during the last couple of years that samples with enough purity have been fabricated to allow ballistic transport.

All impurities cannot be removed; the conduction electrons must be compensated by positively charged donors to maintain charge neutrality. Although one keeps these donors separated from the electrons by using modulation-doping, the scattering can be expected to be significant.

Our earlier model has been enhanced to include the effects of the donor atoms. Using analytical methods this would have been a major challenge. With a numerical simulation this merely re-

quired adding one subroutine. A random distribution of atoms is created and, using Fourier transforms, their contribution to the electrostatic potential at the plane of the electrons is calculated.

It was shown that with this effect added, device performance was drastically reduced. (See Figure 2.9 on page 2.9.) The device shown here, although interesting for basic science, would not be practically useful. Work is currently in progress to improve the models and find better device parameters to solve this problem.

In a parallel effort the effects of phonon scattering is studied. This type of scattering is caused by the thermal vibrations of the semiconductor lattice. This effect is important to understand in order to predict the performance at temperatures higher than 4 K. These higher temperatures are of course preferred in practical applications.

Phonons interact with electrons by changing the potential experienced by the electrons. Phonon scattering is included in the model by adding a time varying perturbation to the electrostatic potential. To be practical this method requires that the potential is calculated in advance and stored in a long sequence of matrices. This is simple due to the large primary memory of the CM.

5.4 Smooth Particle Hydrodynamics

Magnus Selhammar
Uppsala Astronomical Observatory, Uppsala University

Smooth Particle Hydrodynamics (SPH), is a gridless Lagrangian hydrodynamic method. Instead of a grid one uses particles to represent the density distribution of the model. This representation is often efficient in astrophysical applications because it is easily implemented in three dimensions. It is also easy to include different physical phenomena such as gravitation, cooling or magnetic fields. Another advantage over the usual grid methods is the possibility to work with great density differences in the evolution of the model.

This work has mainly been focused on making an efficient general parallel SPH code for studies of accretion disks in star formation modeling. Gravitation using the Barnes-Hut method has been implemented to structure the calculation. The tree is also used to find the neighbour particles for the hydrodynamic interaction. Work has also been done on the use of artificial viscosity

in SPH, and its use in models with varying length scales.

5.5 Quantum Wavepacket Studies

Mats Persson
Department of Applied Physics, CTH

The detailed mechanisms behind a wide variety of important surface phenomena like catalysis, surface chemical reactions, plasma-wall interactions, and growth of materials, are most convincingly revealed by dynamical studies of elementary processes on an atomic scale. Examples of such processes are sticking, tunneling through and climbing over reaction barriers, and gas-surface energy transfer. The rapid advances and developments in experimental methods for the study of such processes like, for instance, state-to-state molecular beam scattering makes this field a timely subject for theoretical studies.

In many cases a quantum mechanical description is needed and the associated complexity often makes computer simulations of detailed models necessary. We study elementary dynamical processes at surfaces using pseudospectral methods for time-dependent propagation of multi-dimensional wavepackets on the CM. In these methods the multi-dimensional wavefunction is represented on a grid and in each timestep the action of the kinetic part of the Hamiltonian on the wavefunction is handled by an FFT of the wave-function [Per, 1991]. The limiting factors in the computations are the primary memory needed to represent the wave function and the speed of the FFT subroutines. A parallel machine like CM200 is found to be very well suited to handle these factors.

In collaboration with Bret Jackson at Department of Chemistry, University of Massachusetts, I have just completed a study of a prototype catalytic surface reaction; the formation of hydrogen molecules by the direct reaction of an incoming hydrogen atom with an adsorbed hydrogen atom on a copper surface [Persson and Jackson, 1993]. This study is motivated by recent dynamic measurements of such the Eley-Rideal mechanism for reactions involving hydrogen atoms on metal surfaces. The dynamics of this reaction mechanism is of interest in understanding the formation of hydrogen molecules in interstellar space and also for plasma-wall interactions in fusion reactors.

Earlier two-dimensional wavepacket studies using restricted col-

inear configurations show that this kind of a reaction can produce the observed high vibrational excitations of the formed molecules [Jackson and Persson, 1992]. However, this study was done on SPARC workstations and a CRAY vectorcomputer and cannot be extended to a more realistic model that can give full rovibrational distributions and reactive cross-sections due to the limited primary memory on these machines whereas such a study is feasible on a CM200 with its large primary memory and also its high speed. In this model all six degrees of freedom of the two atoms are included with a flat surface approximation for the model potential energy surface. This effectively reduces the six-dimensional problem to a tractable three-dimensional problem in curvilinear coordinates by introducing three conserved quantities. We have developed a new and efficient method to handle the kinetic part in these coordinates.

Our extensive computational study of the Eley-Rideal dynamics includes the dependence of the reactive cross-section, translational energy distribution and rovibrational distributions on the shape of the model potential energy surface and on the kinetic energy and angle of the incident H-atom and the vibrational state of the adsorbed H-atom. For instance, the result for the rovibrational distribution displayed in Figure 5.5 demonstrates that the product H_2 molecule ends up in highly vibrationally and rotationally excited states with a characteristic bimodality of the rotational distribution. In Figure 5.5, we show a snap-shot of the reduced two-dimensional probability density of the three-dimensional wave-packet in the reaction zone.

The code is written in CM Fortran and we use a versatile CMSSL subroutine for the time-consuming discrete fast fourier transform (FFT). The grid size is $128 \times 128 \times 256$ (single precision) and one time step takes about 6.0 seconds where 2.7 seconds is spent in the time propagation which includes two calls to the FFT subroutine. A full calculation involves about 4000 time steps and a run takes about 6 hours. So it has been possible to do rather extensive studies for different initial conditions and model potential energy surfaces.

At the moment we have extended our study to the Eley-Rideal reaction of the different isotopic combinations of hydrogen in order to investigate, as suggested by our earlier colinear studies and also by recent molecular beam experiments, the interesting dynamics

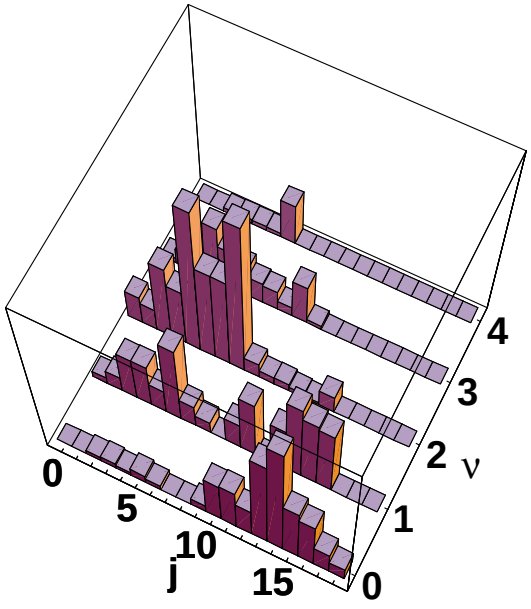


Figure 5.4. A three-dimensional barchart of the internal rovibrational state distributions of a hydrogen molecule formed in a Eley-Rideal reaction at a surface. The height of a bar gives the relative probability to find a molecule in a rotational state j and vibrational state ν.

associated with the large mass differences for the isotopes of hydrogen. The heavier mass of deuterium requires a grid of the size $256 \times 256 \times 256$ and cannot be run on the present configuration of the CM200. In collaboration with Diane Lynch of Thinking Machines Corporation, this calculation is presently performed on a 64-node CM-5 machine.

In summary, the computational study of dynamical processes at surfaces using multi-dimensional quantum wave packets is well-suited for a data parallel machine like the CM200 and is typically very computationally demanding. The availability of the CM200 and other CM machines have made it possible for us to go beyond restricted two-dimensional models for quantum dynamics to more realistic models.

5.6 Global Effects in Cellular Automata

Jan Hemmingsson
Physics and Measurement Technology, LiTH
Gongwen Peng
HLRZ, Forschungszentrum Jülich, Germany

A cellular automaton is a spatially extended system consisting of many connected cells or sites. Each cell contains a variable which

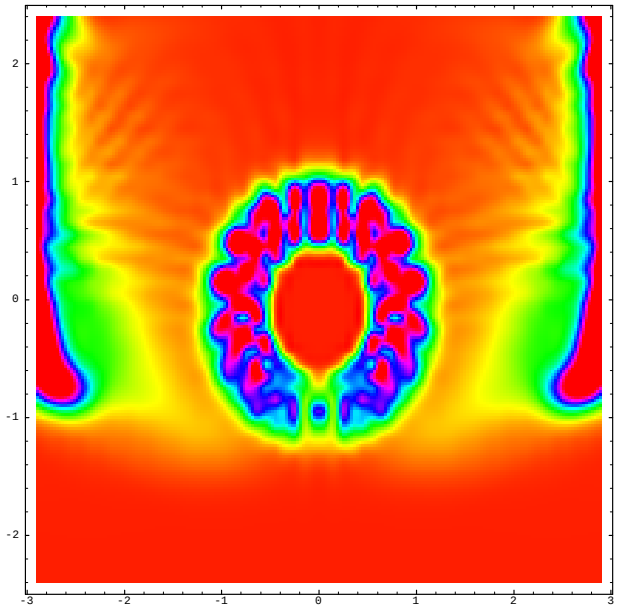


Figure 5.5. Contour plot of the reduced two-dimensional probability density of the 3D wavepacket describing the formation of a molecule by a Eley-Rideal reaction of two hydrogen atoms at a surface. It gives the probability to find the atoms in a relative position b perpendicular to the surface and with a relative distance a parallel to the surface. Note that the wavepacket has been defined for negative a by a reflection in the $a = 0$ axis.

takes discrete values. The time is discrete, and at each time step, the whole system is updated due to some local rule, i.e. the value $\sigma(t+1)$ of the site i at time $t+1$ depends on value at time t of the sites in some neighborhood around i. All cells are thus updated synchronously, normally according to one fixed rule.

In the very simple class of binary cellular automata, where each site can take the values zero or one, it has been found that for certain rules, the magnetization, defined as the average number of cells containing a one, may show a periodic or quasiperiodic behavior with time [Chaté and Manneville, 1991]. This behavior is very stable to external noise, and the time series contain intrinsic noise due to randomness in the local configurations.

One way to visualize such a behavior is to make a return plot. Along the x-axis is the magnetization at time t, and along the y-axis is the magnetization of time $t+1$.

Despite the effort put forth by several groups [Hemmingsson and Herrmann, 1993, Pomeau, 1993, Grinstein *et al.*, 1993, Binder and Privman, 1992] these systems have not been completely understood yet; so far, there has not been found a periodic behavior in three dimensions, and there are even arguments that such a system cannot be stable under generic conditions. We have investigated a mixture of two of the rules in four dimensions [Hemmingsson and

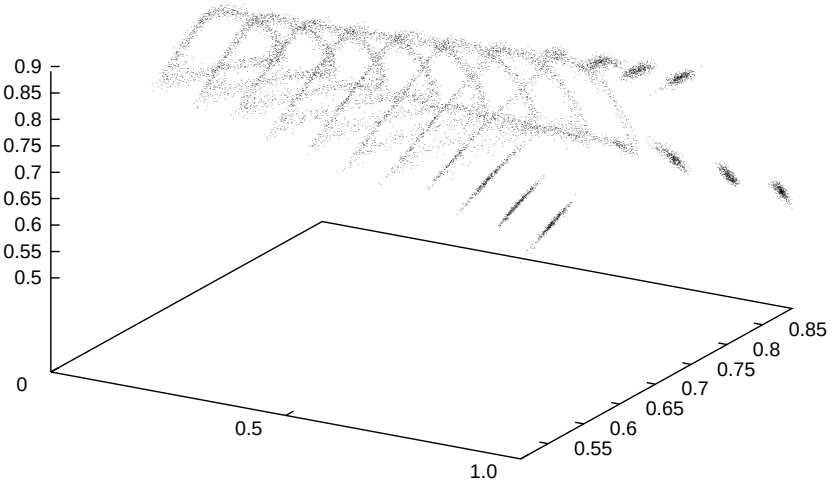


Figure 5.6. Return maps for different values of mixing parameter p. At $p \approx 0.79$ the behavior changes from quasiperiodical to periodical.

Peng, 1994]. One rule shows quasiperiodic behavior, the other periodic behavior. In our simulation, a fraction p of the sites were randomly chosen to be updated according to the periodic rule, and the remaining sites were updated according to the quasiperiodic one. In Figure 5.6, the return map for different values of p are shown. In order to investigate this change more qualitatively, we fourier-transformed the time series, and extracted the period three component Θ. By plotting Θ against p for different system sizes, we found a phase transition at $p_c \approx 0.79$, this means that at this critical mixing of rules, the behavior changes from being quasiperiodic to periodic. We also found the values of the critical exponents normally used for describing critical phenomena.

5.7 Monte Carlo Studies of the Dynamics of Random Anisotropy Dipolar Models

Jan-Olov Andersson, Tomas Jonsson
Solid State Department, Uppsala University

Experimental studies have shown frozen ferrofluids, i.e. solutions of magnetic particles suspended in a nonmagnetic solvent [Rosensweig, 1985], to exhibit glassy behaviour: hysteresis, irreversibility, a peak in the zero-field magnetisation and nonexponential relaxation [Luo *et al.*, 1991]. Similar behaviour has been found in both experimental and theoretical studies of spin glasses [Fischer and Hertz, 1991].

The glassy behaviour of the ferrofluids can be discussed in terms of dipolar interactions between the magnetic particles combined with a random anisotropy, the latter feature being due to the freezing. In a recent simulation study [Załuska-Kotur and Cieplak, 1993], some of the experimental behaviour could be reproduced in a local-mean-field study of the random anisotropy dipolar (RAD) model. Using this approach, the hysteresis, irreversibility and the peak in the zero-field magnetisation found in experiments can be reproduced.

Due to the intrinsically non-dynamical nature of the model, it is not possible to determine any dynamical properties. To get access to dynamical properties, it is necessary to use the MC method, which has previously been found to reproduce both the crossover from quasi-equilibrium to non-equilibrium, normally denoted aging, that is evident in spin-glasses [Andersson *et al.*, 1992], as well as reproducing the effects of temperature cyclings [Andersson *et al.*, 1994] giving evidence for the concept of overlap length [Fisher and Huse, 1988] in spin glasses.

We have been able to verify that the non-exponentional, nearly logarithmic, relaxation found in experiments is also evident in MC simulations.

5.8 Excitation Morphology of Ising Spin-glasses

Jan-Olov Andersson
Solid State Department, Uppsala University
Paolo Sibani
Fysisk Institut. Odense Universitet

Spin-glasses are magnetic materials with two essential features: disorder and frustration. One type of spin-glasses are metallic alloys of small concentrations of randomly distributed magnetic impurities, e.g. Mn, in a non-magnetic host such as Cu. Another type of spin-glasses are alloys of two different isolating magnetic materials, such as $FeTiO_3$ and $MnTiO_3$. The complex physics of spin glasses has been intensively studied in the past two decades by experimental, theoretical and numerical methods [Fischer and Hertz, 1991]

For the purpose of numerical simulations, the systems are often modeled by a set of Ising spin variables $S_i = \pm 1$, which are placed on a regular grid in two or three dimensions. Let x be any of the

2^N configurations of a system of N spins. Its energy is defined in the model as

$$E(x) = -\sum_{i,j} J_{ij} S_i(x) S_j(x). \tag{5.1}$$

For $i < j$ the couplings J_{ij} are independent gaussian variables, with zero average and variance 1; the remaining couplings are fixed by symmetry requirements. Only nearest-neighbour couplings are considered, i.e. $J_{i,j} \neq 0$ only for indices i and j representing neighboring grid points. Lattice sizes up to 32^3 in 3D and 256^2 in 2D have been considered in this study.

As a first step, several local minima Ψ of the energy are found by a careful annealing procedure and then stored for subsequent use. Their energy per spin is comparable to published estimates of the ground state energy per spin for the relevant models. The excitation morphology and phase-space of the system close to each local minimum is then sampled by first generating an excited state with a well defined energy b relative to the energy of the reference state Ψ. After this the system is quenched to a new local energy minimum Ψ'. Connected clusters of spins which differs in the two configurations are finally identified, and their number, size and energy dependence is studied as a function of b. The parallel algorithm used for identifying the spin-clusters can be found in [Hillis and Boghosian, 1993]

In a complementary approach we have replaced the excitation step above by a conventional MC simulation. After completing the simulation we quench the system and analyze the excitation as described above. For more details see [Sibani and Andersson, 1993, Andersson and Sibani, 1993].

5.9 Mapping the Spinodal Region

William Klein
Boston University, Boston, USA
Lawrence Thomas
Theoretical Physics, KTH

Our current work centers around completely mapping the spinodal curve of the modeled system. The spinodal curve represents the separation of the stable region from that of the metastable and unstable regions within a system consisting of two constituents.

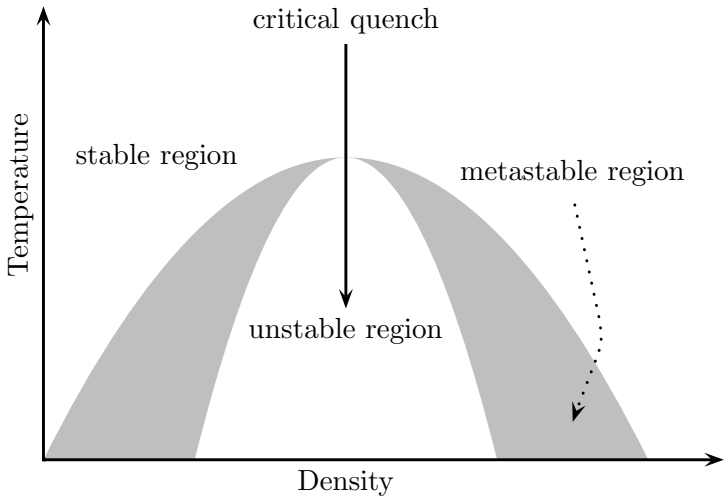


Figure 5.7. Diagram of a critical quench through the spinodal region in a Klein-Conniglio diagram.

Above this curve, the two components are miscible in all proportions, homogeneous and considered a stable system. Below this curve, there exists the meta-stable and unstable regions. The metastable region is similar to that of the stable region in most respects, i.e. the mixture is in a homogeneous state. They differ with respect to the Gibbs free energy (at constant pressure and temperature) which is higher for the homogeneous mixture than for a system formed by two co-existing phases. In the unstable region, there exists a separation of the material into its constituent phases. [Glansdorff and Prigogine, 1975]

In order to map this spinodal curve, we have used the Connection Machine to set up an Ising model system in a 128 x 128 square lattice configuration. The system is initially set up with a random distribution of constituents which are opposite in nature, i.e. half the sites contain up spins and half the sites contain down spins. The system is then subjected to a magnetic field. Because the Ising model is magnetic in nature, some of the spins will reverse their characteristics, causing an imbalance in the ratio of particles represented in the system.

The system is then allowed to evolve in a heat bath, which is simulated via a Metropolis algorithm [Gould and Tobochnik, 1988]. The overall temperature is then lowered and each site within the system has the probability of bonding with any neighboring 'like-

site' within its interaction range. The temperature continues to decrease until a transition from a paramagnetic state to a magnetic state occurs. This transition is marked by an infinite cluster (which is defined as a cluster of like sites that spans the entire lattice unbroken) appearing within the system. This infinite cluster is found by taking advantage of the CM's parallel architecture through the utilization of a multi-grid cluster finding algorithm on the system [Bower *et al.*, 1990].

The simulation is then repeated for varying strengths of the magnetic field with results plotted in order to reveal the spinodal curve of the simulated field.

6 Biocomputing

Scientific computing done by researchers in medicine and biology can be viewed as a new branch of science. This branch is often called biocomputing. It is transforming many parts of medicine and biology from an informational science into a computational and analytical science [Lander *et al.*, 1991]. For several years now PDC has had people working on protein sequence matching. A protein sequence matching server is currently available on the CM and researchers connect to this server with a client program running on their local workstation and match protein sequences against the database on the CM (Section 6.1). This project has now taken a step further by automating the process all the way from DNA samples to reports based on a data base scan of the sequences found in the laboratory samples (Section 6.2). A new application using recurrent Neural Networks to do DNA sequence analysis is reported Section 6.3. This work has close connections to the work in Section 3.1. In Section 6.4 a project where the size of the data sets are a computational challenge, namely the area of PET scan image analysis is described.

6.1 Gene Sequence Database Scanning

Erik Wallin
PDC, KTH
Gunnar von Heijne
Center for Structural Biochemistry, Karolinska Institutet

The client-server system that was developed during 1991 and 1992 has been further refined and the functionality increased during 1993. The engine of the system is still the parallel implementation of the Smith and Waterman sequence similarity measure algorithm. Thanks to the speed of the supercomputer this sensitive method can be used to scan the entire protein sequence database.

Several improvements have been made to the system.

Several improvements have been made to the system. The memory use on the CM200 has been minimized and instead it makes more intensive use of the DataVault. This means that larger databases can be handled without interfering with other users.

Another improvement is the ability to run searches using batch queues during nights and weekends. This is particularly important for large sequencing projects that involve tens of sequences every day.

In order to be able to search the DNA databases, which are much larger than the protein databases, it was necessary to develop new algorithms for the search. These are under evaluation and will be incorporated in the search program during the summer of 1994. With DNA databases it is also interesting to receive regular updates for new sequences every day. This is performed by running a script every night to fetch the sequence updates.

To facilitate for the user to search also the descriptive texts associated with each sequence we have loaded these on to the DataVault. It is now possible to do a free text search through all the records in less than ten seconds. This simplifies the use of the database as the users do not have to worry about how the data is stored and how to write a database query. They simply enter the keywords that they are interested in and the result is returned as a set of sequences that contain the specified keyword. This way the system can also be used as a fast retrieval system for sequence records.

The only real drawback with the system today is that the user interface is still text based. To alleviate this we are planning to release a client program for the Apple Macintosh during the spring of 1994. This should hopefully simplify for the infrequent user.

6.2 Large Sequencing Projects

Erik Wallin
PDC, KTH
Gunnar von Heijne, Rhiannon Sanders, Khalid Islam, Edvard Smith
Center for Structural Biochemistry, Karolinska Institutet

In large scale sequencing projects the number of sequences that have to be analyzed soon become too many to handle manually. Normally the user has to enter the sequence into the system manually. This takes a lot of time and is also a source of errors. We have tried to automate the process as much as possible. The computer takes care of all the steps from sample to report which details whether the sequence had any similarity or not. (See Figure 6.1.)

In the process of sequencing new genes, the material (a sample

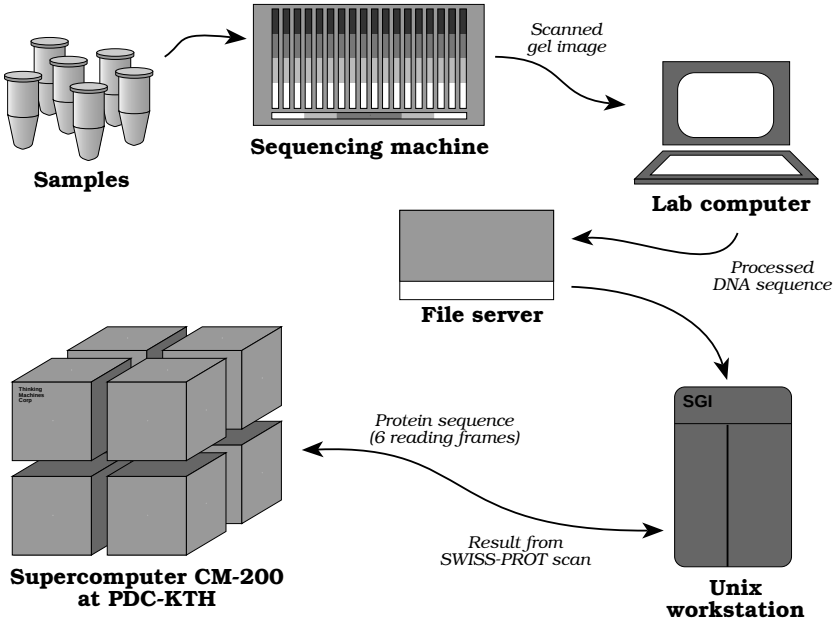


Figure 6.1. Simplifying massive DNA sequenceing. The samples containing the material to be sequenced are submitted to the lab. The lab personnel loads them on the sequencing machine which runs over night. After scanning the gel a lab computer is used to analyze the image and extract the sequences. These are saved on a common server area. The automated search system fetches the sequence and sends it off to the supercomputer for database search. The result is then sent back to the workstation and further processed to produce a summary of the results.

in a test tube) is first sent off to a sequencing lab. The sample is run on a sequencing gel and the gel is analyzed by lab personnel on their lab computer. After that the DNA sequence is stored on a common file server. The normal procedure is then for the scientist to pick up the sequence file and post-process it locally. With our automated setup the sequence is instead fetched and sent off to the search system described in Section 6.1. When the result from the search is finished it is processed into a report and a summary. By looking at them the person submitting the sequence can quickly determine if there is anything similar in the database.

Until today we have run over 300 sequences through this automated search system. This would have taken a substantial amount of time to do manually.

6.3 Recognition of Human mRNA using Recurrent ANN

Hans H.H. Hansen
Department of Physical Chemistry, DTH

Recurrent neural networks and in particular fully recurrent networks have not earlier been applied to identification in DNA sequence analysis. In this project the Real Time Recurrent Learning

(RTRL) algorithm [Williams and Zipser, 1989, Smith and Zipser, 1989] was applied to training of a fully recurrent neural network for splice site identification in human precursor messenger RNA (mRNA).

The massive parallel architecture of the CM200 was exploited by allocating an array of RTRL-networks and simultaneously train these on a rotating queue of input sequences. The CMSSL library and the built-in communication mechanisms of the CM simplified construction and control of the replica system.

Built-in communication mechanisms and the CMSSL library simplified construction and control of the replica system.

The results of training and testing the networks were compared to similar experiments involving conventional feed-forward neural networks [Brunak *et al.*, 1991]. It turned out that a fully recurrent network is a powerful tool for extracting the sequential structure of human DNA sequence defining the splice site locations. When generalizing the extracted structures to classification of "unknown" data, the method successfully identified the splice sites locations. More precisely, a peak prediction rate was reached at 87% (correlation 0.83) for donor sites and 83% (correlation 0.79) for acceptor sites.

Furthermore, it was indicated that the sequence patterns learnable by a fully recurrent network represent neither a subset nor a superset of the patterns learnable by a feed-forward network. This conclusion was reached by comparing the order in which RTRL network and the feed-forward networks of Brunak *et al.* learned to recognize the splice sites of the data set.

6.4 Analysis of 3D Brain Data

Per Roland
Karolinska Institutet, Stockholm
Björn Levin
SANS, KTH

New brain-imaging methods, such as measurements of the regional cerebral metabolism or the regional cerebral blood flow in combination with Positron Emission Tomography (PET), sample the activity in more than 50,000 parts of the brain into one 3D image. These images can be used as clues in solving the puzzle of which different parts of the human brain contribute to the thought process. We have performed a series of simulations to study the effects of noise on different detection schemes for analysis of these images.

Because of the 3D structure and the large amount of data, analysis using a traditional sequential computer would be extremely time consuming

By generating 100 or 1000 re-runs of the same experiment using distribution data from sampled images of subjects in the control state, i.e. a state in which the brain is not engaged in a particular task, it was possible to generate empirical distributions of false positive activation events when using the detection schemes.

The simulations were done in several steps. First the CM was used to investigate the spatial 3D autocorrelation function in a sample of regional cerebral blood-flow pictures. This function was then used to generate a 3D filter that imposes the found autocorrelation in subsequent simulations. Finally, to get the desired accuracy in the estimations, tens of thousands of randomly generated images having the correct statistical properties were analyzed using the mentioned methods, at the same time as false positive events were recorded. In each such generated image around 200,000 values mimicking measurements of biochemical and physiological variables in the brain are generated simultaneously, making the task very well suited for analysis by massively parallel computing. A detailed report of this work can be found in [Roland *et al.*, 1993].

7 Applications in Chemistry

Computer simulations of chemical systems is a complement to theoretical and experimental chemistry, made possible by the rapid growth of computer capabilities. These simulations can be regarded as computer experiments at a molecular level, and they require large amounts of computing power measured both as raw CPU power and as volume of stored and transferred data. By simulating a sufficiently large system one can compare experimental results with simulation results, thereby simplifying the fundamentally important interaction between experiment and theory.

In many systems studied, it is crucial to be able to perform as large simulations as possible to ensure that the results are physically relevant. These large simulations are often naturally massively parallel and require both very large memories and fast calculations. Typical examples are classical and quantum-mechanical many-particle calculations of structural features and dynamic behavior. The present projects running on the CM are examples from molecular dynamics (Section 7.1) and quantum chemistry (Section 7.2).

Large simulations are often naturally massively parallel.

7.1 Molecular Dynamics for Liquids with Coulombic Interactions

Fredrik Hedman
PDC, KTH
Aatto Laaksonen
Department of Physical Chemistry, Stockholm University

We have investigated an approach for large-scale data-parallel molecular dynamics of systems with Coulombic interactions. Short-range interactions are calculated with a method based on coarse-grained cells. In this method the simulation cell is decomposed into equally sized subcells, with the shortest side being larger than the cut-off radius of the short-range interaction. Electrostatic interactions are calculated using a data-parallel version of the Ewald summation method. Calculations of long- and short-range interactions are merged by a suitable choice of the size of the subcells and the Ewald sum convergence parameter.

The computational effort of the Ewald summation method scales with the number of particles N as $O(N^{3/2})$ when one chooses method parameters optimally. [Kolafa and Perram, 1992, Fincham, 1993] For appropriate choices of system size we find that a full CM200 would require between 10 and 15 seconds for a time-step with of 256 thousand particles. The conclusion is that for very large simulations the Ewald summation technique and the coarse-grained cell method do not match very well. However, if one could neglect the Fourier-space part of the calculation at only a small loss of accuracy then the method may still turn out to be an interesting alternative. [Hedman and Laaksonen, 1993]

7.2 A Direct Recursive Residue Generation Method

Hans O Karlsson, Osvaldo Goscinski
Department of Quantum Chemistry, Uppsala University

A major bottleneck in large scale matrix eigenvalues problems is the need to construct and save the Hamiltonian matrix in computer auxiliary memory. The idea of *direct methods* is to avoid the use of secondary storage by representing the Hamiltonian matrix in terms of integrals and coupling coefficients. The only way the matrix enters the eigenvalue calculation is via a matrix-vector product.

A proven and useful method for Rydberg atoms in strong external fields is the Recursive Residue Generation Method (RRGM). It is based on a partial tridiagonalization of the Hamiltonian matrix via the Lanczos algorithm (LA). In the LA a Krylov space is built, spanned by a starting vector and successive applications of the matrix. The CPU time of the process is solely determined by the matrix-vector product.

To study the simplest Rydberg atom, Hydrogen, a basis set built up by spherical harmonics for the angular part and Laguerre functions for the radial part was used. Within this basis, perturbations result in sparse matrices. The Hydrogenic Hamiltonian, on the other hand, is block diagonal in angular blocks leading to a memory need of $N^2/2$ where N is the number of radial functions. For large perturbations and/or high excitations large basis set are needed leading to storage problems. To bypass this problem a direct method was developed.

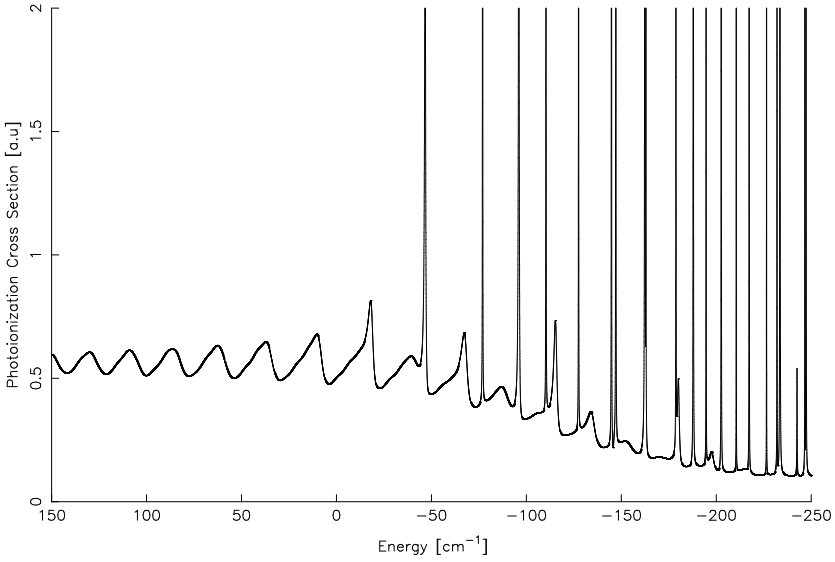


Figure 7.1. Photoionization cross section of the Hydrogen atom in an static electric field calculated with the RRGM method.

For the Laguerre basis used, we noted that via a factorization, the Hamiltonian matrix times a Lanczos vector, could be written as a sum of three vectors which can be constructed in a recursive way within each angular block. The direct method leads to a storage reduction from $N^2/2$ to $4N$ per angular block. Further, the recursive procedure can be done in *parallel* for all angular blocks. For details see [Karlsson and Goscinski, 1994].

To fully exploit the parallelism the direct version of the RRGM the program was implemented on the CM200. The high level of parallelism led to the result that the computational time for the Hydrogenic matrix-vector product is *independent* of the number of angular blocks and scales *linearly* with the number of radial functions. The CPU time is in fact determined by the CMSSL routine used for the sparse perturbation matrix-vector product.

The CPU speedup due to the parallelization of the code and the large memory of the CM200 has made it possible to study systems, such as the photoionization of Hydrogen in strong electric fields. (See Figure 7.1.)

8 Geophysics

Parallel computers have in recent years found many applications in the field of computational geophysics. These applications include 2D and 3D solutions of the elastic anisotropic wave equation by finite-differences, modeling fractured rocks, lattice-gas modeling of wave propagation and solution of the seismic travel-time inversion problem. In particular, access to GBytes of fast data storage, as well as calculation speeds on the order of GFlop/s, allows the simulation of 3D transient wavepropagation.

Numerous rocks and materials exhibit anisotropic behavior. The first step in understanding the effects of anisotropy in the seismic data we collect is to model seismic wave propagation in anisotropic media. (Section 8.2)

8.1 Simulation of Ground Vibration on the CM200

Marcus Berglund
C2M2, KTH

During the Bruce Springsteen concert held at the football stadium "Nya Ullevi" (Göteborg, Sweden) in 1985, the rhythmic motion of the crowd lead to violent structural vibrations of the stadium and surrounding buildings with resulting structural damage. This incident has motivated intensified research into elastic wave propagation in the kind of soil materials found in the foundations of the arena.

Wave propagation in soil can be approximated by a three dimensional linearly elastic model in which the modulus of elasticity increases with depth, the density and Poisson ratio are constant and the material damping is assumed to follow the common Rayleigh model.

Earlier studies have been restricted to 2D models by the computing resources. The 2D approximation is very crude since the decay of wave amplitude with distance is $1/r$ in 2D, rather then the correct $1/r^2$. Also the fact that the bedrock under the stadium is very irregular (See Figure 8.1.) makes the 3D effects important.

These kinds of problems are of general interest in soil dynam-

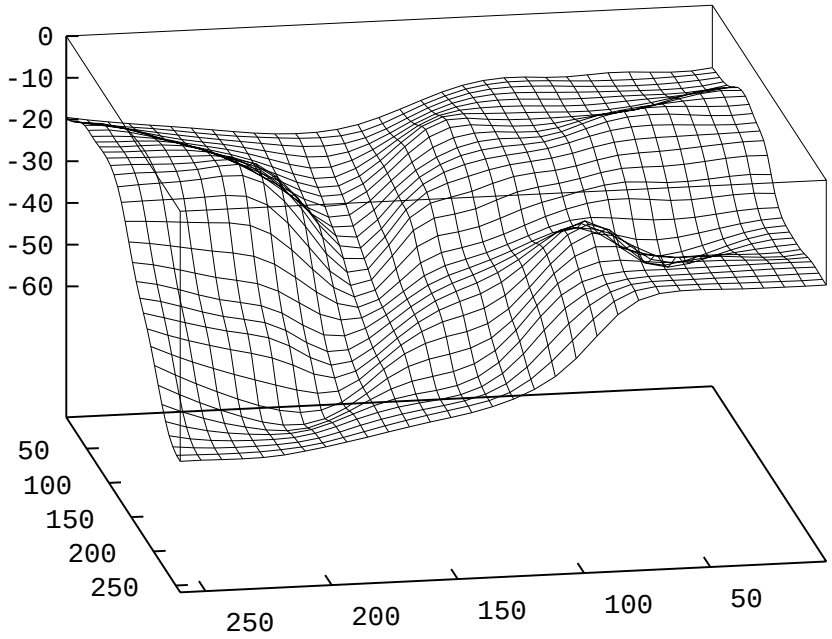


Figure 8.1. Geometry under sports arena "Nya Ullevi" where bedrock is approximated with splines. Length scale in meters.

ics with applications in e.g. analysis of effects of earthquakes and effects of moving loads, like trains and trucks, where the three dimensional geometry is of great importance.

The choice of algorithm is guided by the demand for high accuracy and a realistic computation time. This is achieved by a compromise between the computational stencil and the data structures. Both have to be simple enough to allow for a high FLOP rate on a parallel computer, but also general enough for sufficient numerical accuracy.

The computational domains we have in mind can be transformed by a smooth mapping to a box. This means we can use a structured regular grid for the unknown displacements and an explicit computational stencil, i.e. the new value of an unknown displacement in a time-step is given by a linear combination of the old value and old values from neighbor coordinates. (See Figure 8.2.) Interior and boundary points are treated in the same way by using appropriate stencil weights.

Second order finite differences are used on transformed equations in the interior of the domain. The absorbing boundary conditions are of first order numerical accuracy which is enough for overall second order accuracy [Gustafsson, 1981].

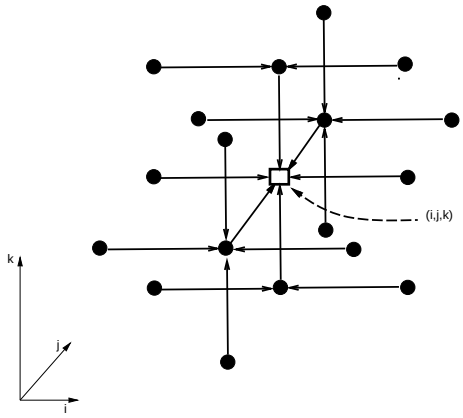


Figure 8.2. Communication pattern for the stencil operation at an inner node.

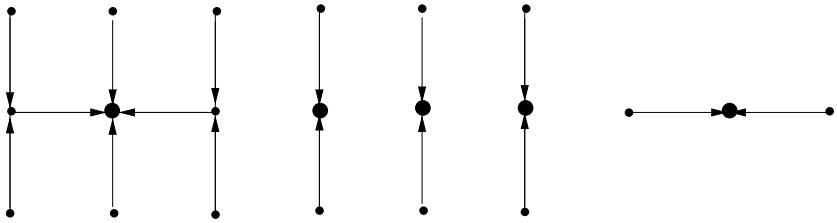


Figure 8.3. The nine-point stencil in the left figure is decomposed into two communication and computation steps. The cost of the naive way of calculating our stencil is 28 CSHIFT operations and 37 floating point operations, which should be compared with 8 CSHIFT operations and the same number of floating point operations for the sub-stencil decomposition.

Implementation on the 8K CM200

The program ELWA has been implemented using CM Fortran. The current performance is 872 MFlop/s in single precision using an 8K CM200 [Berglund, 1994].

The program has been validated by comparing results with an analytic solution of a one dimensional problem and with results obtained from the finite element package ABAQUS [Hib, 198] on a 2D problem. ELWA has also been used by Erlingsson in his thesis [Erlingsson, 1993].

The most important thing when programming a massively parallel computer is to distribute the data in an appropriate way onto the processors so as to minimize communications and maximize local computations. It is also important to avoid operating on a subset of the data, otherwise the operation is performed on the whole data set using a mask operation.

Since nearest neighbor communication is fast on the CM200, the default distribution is natural for our case. However, in our algorithm the stencil operation for the nodes belonging to the prescribed surface tractions boundary condition has to be evaluated separately. Fortunately we have this boundary condition on only

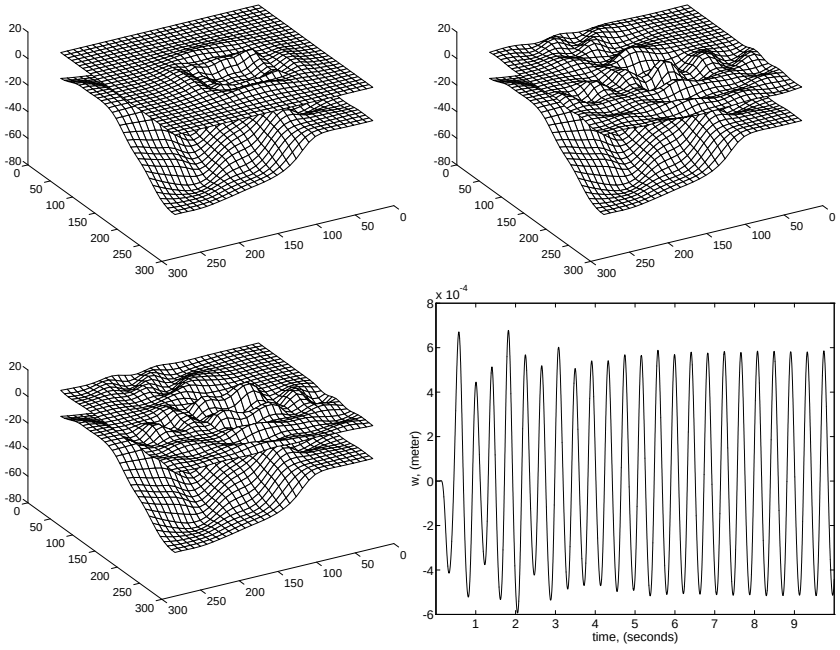


Figure 8.4. In the first three figures the total displacement field at the surface is plotted together with the bedrock at times t = 0.5 seconds, 5 seconds and 9.75 seconds. All displacements are multiplied by a factor 7000. In the last figure the vertical displacement component w is plotted versus time at an observation point at the surface. Every second point is plotted.

one face which suggest another data layout than the default. By keeping the data of grid points on lines orthogonal to the surface in the same node, the surface boundary conditions can be done about ten times faster compared to using the default data distribution. In CM Fortran parlance a layout of `(:serial,:news,:news)` is used instead of `(:news,:news,:news)`. We can do yet another optimization; complicated stencil operations can be decomposed into sub-stencil operations. This is illustrated in Figure 8.3 on a 2D stencil.

Ullevi simulation

A crowd consisting of thirty thousand people, jumping up and down in front of the stage, with approximately 4–5 persons/m^2 leads to a load amplitude of 3.0 kPa. Pernica has found that audiences can easily produce rhythmic forces with a frequency of 1–3 Hz [Pernica, 1988]. Sahlin has estimated the frequency range to between 2.2 and 2.5 Hz for the "Nya Ullevi" concerts [Sahlin, 1989]. Figure 8.4 shows the result from a simulation with $\nu = 0.47$, force frequency 2.0 Hz and damping constants $\gamma = 0.241$ and $\kappa = 0.0016$. To solve this transient problem 5 seconds forward

in time with 64^3 nodes and $3 \cdot 64^3 \approx 785000$ degrees of freedom takes about 6.5 hours on an 8K CM200.

8.2 Elastic Wave Propagation in 3D Heterogeneous Media

Christopher Juhlin
Solid Earth Physics, Uppsala University

In 1992 a program was implemented on the CM200 at PDC to solve elastic wave propagation problems in 2D hexagonally anisotropic media. The velocity-stress formulation of the problem [Levander, 1988] was used, where only first order equations are treated. A finite difference scheme on staggered grids was used with fourth order operators in space and second order operators in time.

The ultimate purpose of this research is to compare physical model data collected over 3D anisotropic solids with numerical modeling results using finite difference methods. Current limitations of memory do not allow a complete comparison at this time. A realistic model of elastic wave propagation in 3D anisotropic media will require about 16 GByte of core memory.

While looking into alternative methods for modeling and waiting for an upgrade of the capacity at PDC, other programs were implemented on the CM200 dealing with elastic wave propagation. Of these, one was a code for elastic wave propagation in 3D heterogeneous isotropic media, and another was a code for radially symmetric isotropic media using the same formulation as for the 2D anisotropic program. The former is being used to study the effects of surface topography on the waveforms recorded in seismic experiments. The latter is being used to model the seismic wavefield produced by a mechanical source in a borehole.

In addition, cooperation has been initiated with the Institute of Earth Sciences at Uppsala University to compare seismic sections generated by modeling on the CM200 with real data from petroleum exploration. An example is shown in Figures 2.5–2.8 on page 14 where synthetic seismic data are generated over a salt dome using the exploding reflector principle [Loewenthal *et al.*, 1991]. A zero-offset section, where the source and receiver are located at the same point on the surface, may be simulated by letting the reflectors explode (Figure 2.6). These synthetic data (Figure 2.7) are then migrated to provide an image (Figure 2.8)

which can be compared with real data. The goal of the modeling is to constrain the spatial location of the bottom of the salt dome in the vicinity of its neck. Ideally, Figures 2.5 and 2.8 should be equivalent, however, since the model is 2D and the synthetics were migrated using a 1D velocity function, the original model is not recovered.

8.3 Groundwater Transport Modeling

Roger Thunvik
Civil and Environmental Engineering, KTH

A numerical model for saturated-unsaturated water flow in porous media coupled with solute mass and heat transport is under development at the division of Land and Water Resources. As the model evolves it is also being implemented on the CM200 at PDC. The intention is to develop a tool for studying problems in e.g. groundwater flow, contamination, artificial recharge, saltwater intrusion in coastal aquifers, heat convection, analysis of field tests and unsaturated flow. The model code was tested on a problem concerned with the modeling of the average hydraulic gradient in a coastal aquifer [Soldal *et al.*, 1993], conceptualized as a 2D vertical profile, whose boundary conditions were fixed in time. The parallel version of the model could for example be used for a 3D analysis of the above problem with transient boundary conditions, such as the tidal effects in a fjord, and also account for time dependent changes in the elevation of the river through the flow domain.

The code is written in Fortran 90 and solves the governing equations for coupled water, heat and solute transport in a porous medium in 1D, 2D, or 3D. The solution method is based on a Galerkin finite element method for the spatial discretization of the flow domain. Time integration is performed using a backward Euler (or trapezoidal) time stepping scheme. The parallel implementation integrates over all the elements in parallel. This is achieved by keeping all the values and physical properties as well as basis functions and their derivatives, at the integration points for all elements at all times in core.

The original version of the model uses a frontal method (Harwell's MA32-solver) for solving the algebraic equations resulting from the integration. This algorithm does not mix well with the parallelization used, so the parallel version uses an iterative solver

based on the so-called TFQMR algorithm.

9 Numerical Analysis

Massively parallel machines present a large challenge to numerical analysts, since many old and established serial algorithms and methods turn out to perform very poorly on these machines. However, those algorithms that are well adapted to these machines are worth optimizing. This has been done in the FFT project (Section 9.1). It is also interesting to investigate how well some algorithms can be implemented directly in Fortran 90 (Section 9.6). New opportunities in using new parallel algorithms have been explored in the areas of: Legendre transforms (Section 9.3), parallelizable preconditioning methods (Section 9.5) and optimization (Section 9.7).

9.1 Mingle and Un-mingle for Real-to-Complex Transforms

Lars Malinowsky
PDC, KTH

The fact that Fourier Transforms of real data result in conjugate-symmetric sequences is traditionally used to store only half of the conjugate-symmetric sequence, with proportional savings in both memory usage and compute time. The elements of the conjugate-symmetric sequence are also known as wave-numbers. The relationship $X(k) = \overline{X(N-k)}$ can be used to obtain the data that are not stored, so that only $N/2+1$ wave-numbers are needed. In CMSSL, the two wave-numbers $X(0)$ and $X(N/2)$ are packed into one complex number. Thus, in fact, N real numbers are sufficient to store both the real sequence, and the conjugate-symmetric sequence. In principle, the real and the conjugate-symmetric arrays can be equivalenced into a single array of real length N. However, CM Fortran does currently not support such equivalencing. The real and complex data must be represented as separate arrays. Nevertheless, for the purpose of load balance on parallel machines, it is beneficial to store data in arrays of length N rather than length $N+1$.

CMSSL conserves storage and enhances load balance by storing each conjugate-symmetric sequence in $N/2$ complex numbers. In the case of multidimensional transforms, the fact that two real sequences are stored in one complex number after the transform along the first axis results in some interesting alternatives for the storage of data for subsequent transforms.

In short, each new dimension transformed generates two new mingled conjugate-symmetric sequences, mingled since they have been transformed with ordinary complex sequences. A method has to be found that, in a manageable way and in-place, unmingles and stores these two new sequences of wave-numbers in the same amount of space as any of the wave-numbers of their companion complex sequences require. The following method has been implemented: for each transform after the first, store elements of the two conjugate-symmetric sequences in a fashion such that the position in the array gives the wave-number for one or the other sequence. [Thi, 1993]

9.2 Parallelizing the Fast Wavelet Transform

Mats Holmström
TDB, Uppsala University

The interest for wavelets and wavelet techniques has grown enormously over the last few years, both in theoretical and applied areas. In image compression wavelets are used as an alternative to Fourier techniques. In numerical analysis wavelets are used for solving integral equations and partial differential equations. To understand the basic properties of wavelets it is of value to make a comparison of similarities and differences between wavelets and the more familiar Fourier basis.

If we have a time dependent signal and want to gain information about its frequency content, the standard solution is to use the Fourier transform. One drawback of the Fourier transform is that we do not get any information about *where* in time these frequencies are located. A short pulse cannot be located in time by examining the Fourier spectrum of the signal. In signal analysis one usually solves this problem by using a windowed Fourier transform.

Wavelet analysis provides another approach to this localization problem by using basic building blocks that are smaller for higher

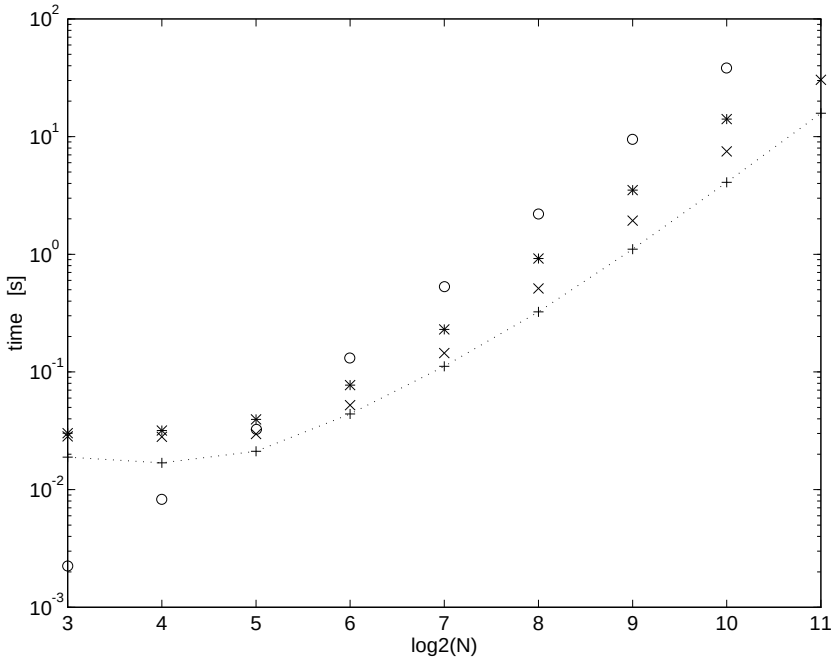


Figure 9.1. Comparison of execution times for some two-dimensional FWT algorithms. The FWT is done in three stages on a square with N^2 points. Legend: '$\circ$' a sequential algorithm on a Sun SPARC-10; '$*$' Algorithm 1; '$\times$' Algorithm 2; '$+$' Algorithm 3. All three algorithms were executed on a CM200, configured with 128 FPUs.

frequencies. These building blocks, or basis functions, obey a relation of the following type

$$\psi_{j,k}(x) = 2^{-j/2}\psi(2^{-j}x - k).$$

All basis functions are scaled and translated versions of a single *mother wavelet*, $\psi(x)$. The scaling corresponds to index j and the translations to index k. We can then represent a function as a linear combination of these basis functions, $f(x) \approx \sum_{j,k} b_{j,k}\psi_{j,k}(x)$.

Since we usually deal with sampled values of functions, i.e. we know a functions values $f(x)$ at certain points $x_i,\ i = 1, 2, \ldots, N$, we need an efficient way of calculating these wavelet coefficients $b_{j,k}$, given the function values $f(x_i)$. The algorithm for doing this is called the Fast Wavelet Transform (FWT). The time for executing the FWT on N points is proportional to N. This can be compared to the Fast Fourier Transform which has an execution time that is proportional to $N\log(N)$.

When using wavelet methods on large scale problems the time to execute the FWTs can be prohibitively long, although the FWT has a linear time complexity on sequential computers, as noted above. One solution is to use massively parallel computers, but

we are then faced with the problem of constructing an efficient FWT algorithm for such computers.

The reason for the need of different algorithms for parallel computers is that new considerations, such as communication time has to be taken into account when constructing parallel algorithms. In CM Fortran the elements of an array are distributed on the processors according to a virtual grid. It is desirable to reduce the amount of communication between the processors by trying to make the computations local. When we need communication we prefer NEWS-communication (shifting the whole array on the virtual grid) since it is fast on the CM.

Taking the above communication considerations into account, two new algorithms for doing the FWT on a CM were constructed, and compared in execution time with a previously published algorithm. The first new algorithm provided a speedup by a factor of two and is suited for parallel computers in general (actually it is also suited for sequential computation). The second new algorithm uses a feature of the CMs hypercube topology: the ability to quickly shift arrays by a distance that is an even power of two and achieved a speedup by a factor of four compared to the previously published algorithm. In Figure 9.1 the execution times for the three algorithms are presented. Note that the test problem is two-dimensional. This does not present a problem, since when we have an algorithm for the one-dimensional FWT it is easy to extend it to two or more dimensions.

A great help when programming on the CM200 is the debugger Prism. By using Prism one can, in addition to debugging facilities, get timing information on a per-line basis, thus allowing the programmer to evaluate the communication and computation cost of each individual CM Fortran statement as well as statistics for the whole program.

The two new algorithms for the FWT on the CM shows that it is possible to implement an efficient FWT on the CM200. There still remains a lot of work to be done in terms of testing and fine-tuning, and the ultimate goal should be to provide an efficient and robust "black-box" FWT algorithm on the CM200, much like the FFT algorithm that is provided in the CMSSL library.

9.3 Fast Parallel Legendre Transforms

Erik Aurell
Department of Mathematics, Stockholm University
Igor Wertgeim
ICMM, Perm, Russia

Legendre transforms and their generalizations are common-place in mathematics and in the Physical Sciences. They play a prominent place in Lax' construction of the maximum entropy (weak) solutions of hyperbolic conservation laws.

We consider here the inviscid Burgers' equation, for which the solution is essentially determined by a standard Legendre transform, and show that a Fast Parallel Legendre transform (in 1D) may be constructed. The leading term in the operations count of a *serial* Fast Legendre Transform is $N \log N$ in one dimension [She *et al.*, 1992], and $(N \log N)^d$ in d dimensions [Noullez, 1992, Noullez and Vergassola, 1993]. The Parallel Fast Legendre Transform is based on the Connection Machine "send", and "segmented scan" operations. The operation count on a hypothetical machine with N processors, each one holding one data element, arranged in a hypercube for send operations, and on a line for segmented scan operations, is $O((\log_2 N)^2)$. On a real Connection Machine, with a finite number N_{phys} of physical processors, the operation count is $O((N/N_{phys})(\log_2 N)^2)$.

The initial motivation for constructing a fast parallel Legendre transform was to extend the numerical investigations of Burgers' equation with random initial data reported in [She *et al.*, 1992]. A 1D version of the algorithm was implemented on a Connection Machine CM200 and used to study a model problem in [Aurell *et al.*, 1993].

9.4 Solvers for Systems of Equations Arising from PDE Problems

Sverker Holmgren, Kurt Otto, Lina Hemmingsson
TDB, Uppsala University

Solution methods for flow problems, e.g. the Euler, Navier–Stokes or Maxwell equations are studied. Both time-dependent and steady-state problems are of interest. Finite difference discretizations are used in space. For flow problems, explicit time-marching is normally used both for steady-state and time-dependent problems.

This means that the time-step must be of the same order of magnitude as the space-step. For some problems, this stability criterion is unrealistically strict. The aim here is to try to avoid these problems by using solution methods based on solving large, sparse systems of equations. For time-dependent problems, implicit time-marching is employed, and for steady-state problems the system of equations arising from the original problem is solved. In the solution process, large linear systems of equations are solved using conjugate gradient-like iterative methods combined with semicirculant (SC) and semitoeplitz (ST) preconditioners. It has previously been shown that preconditioners of these types yield good convergence properties for flow problems [Holmgren and Otto, 1992, Holmgren and Otto, 1994, Hemmingsson, 1993, Hemmingsson and Otto, 1994].

Explicit time-marching methods are highly parallelizable, while preconditioned iterative methods often are more difficult to implement on parallel computers. Normally, it is vital to perform the preconditioner solve efficiently. The work presented here is part of a project in progress, where we want to prove that exploiting iterative methods combined with SC and ST preconditioners yield solution methods that are competitive with standard explicit time-marching methods on different parallel architectures.

The SC and ST preconditioners are formed by discretizing a system of n_c partial differential equations (PDE) closely related to the original one. The boundary conditions in one space direction are modified, and variable coefficients in this direction are replaced by their averages. The modifications are chosen such that the resulting system of equations can be solved by utilizing fast trigonometric transforms.

On the CM200, only a preliminary version of the SC preconditioner solve has been implemented so far. The CMSSL-routines for FFT are exploited for the fast trigonometric transforms, and the block-tridiagonal systems of equations are solved using an algorithm of cyclic reduction-type. For a system of 4 PDEs, solved on a 256×256 grid, the performance using 4K processors is 90 MFlop/s in double precision. The conclusion is that it is possible to implement SC and ST preconditioners efficiently on massively parallel SIMD computers.

9.5 Implementation of an Approximate SSOR Preconditioner

Ivar Gustafsson, Gunhild Lindskog
Department of Computing Science, CTH

We consider completely parallel solution of finite element discretized elliptic second-order boundary value problems. The problems are solved by an approximate Symmetric Succesive Over-Relaxation (SSOR) preconditioned conjugate gradient method. The preconditioner is constructed from approximate inverses of various degree of accuracy of the SSOR approximate factorization of the matrix of the linear system. The solution of the preconditioning system is then performed by matrix-vector multiplications which may be performed in parallel over the total number of unknowns. [Gustafsson and Lindskog, 1992]

In order to perform an efficient implementation on the CM200, the data has to be distributed onto the processors such that the computations can be done locally in every floating point unit. Also, the communications needed for doing these computations have to be minimized. A natural way of distributing the data in our discretized version of the problem is in a logical grid of processors of the same size and shape as the grid used for the discretization. The only communications needed for the sparse matrix- vector multiplication are then nearest neighbour shifts, which may be performed by the fast NEWS communication.

In CM Fortranthese shifts may be implemented by the cshift or eoshift functions. A faster code is obtained by the parallel shift routine pshift from the library CMSSL. In Figure 9.2 we give the CM200 computing busy-time for the matrix-vector multiplication with different strategies and problem size m, where the number of unknowns is m^2. We use 128 floating point processors.

As we can see, the cshift and pshift functions are efficient for certain sizes of the problem with the number of unknowns equal to a multiple of 512 and inefficient for others. For numbers of unknowns equal to a multiple of 512, the cshift operations are nearest neighbour communications for all processors. The eoshift is less efficient for these numbers of unknowns since it requires assignments of zeros to elements associated with the boundary. The best strategy for a general size is to use extended pshift, where arrays are dimensioned in order to meet the nearest upper efficient array size. All arrays are extended by zeros and the calculations

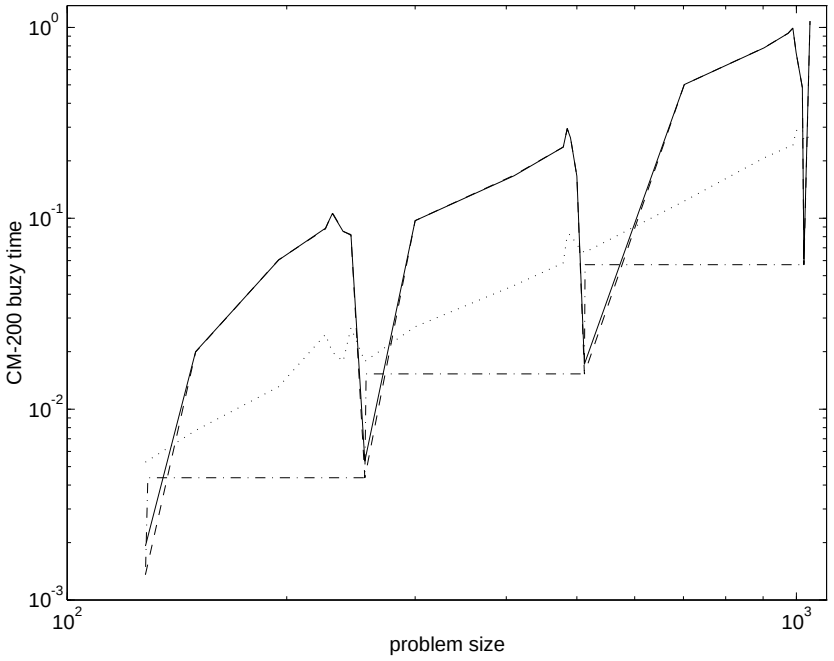


Figure 9.2. CM200 busy times for matrix-vector multiplication for various problem sizes m, solid line for pshift, dashed line for cshift, dotted line for eoshift and dashed-dotted line for extended pshift.

in the algorithms are performed also for these zeros.

The total busy-time for the solution of a model Dirichlet problem shows an improvement of 39 % for the most efficient approximate SSOR method compared to diagonal scaling. Then the extended pshift method is used. The number of MFlop/s in double precision for the diagonal scaling method and the most efficient approximate SSOR method is about 195 and 165 respectively. The problem size is then $m = 1024$. [Lindskog and Gustafsson, 1993]

9.6 Matrix Computations on the Connection Machine

Göran Svensson Helmersson
Department of Mathematics, LiTH

The purpose of the project is to investigate if certain matrix computations can be executed efficiently on the CM when programmed in a data-parallel programming style using CM Fortran, s subset of Fortran 90. Since Fortran 90 is a standardized language available for many different platforms, Fortran 90 programs will be easy to port. However, it has not been clear if it is possible to program the CM on a sufficiently low level from CM Fortran, i.e. if effi-

cient assignment of data to processors and load balancing can be achieved.

In particular we have studied the implementation of an algorithm for QR decomposition of a matrix based on Householder transformations. The QR decomposition is one of the most important matrix decompositions in numerical linear algebra. It is used in numerous applications, e.g. when solving linear least squares problems. By applying a sequence of Householder transformations, the matrix is step by step reduced to upper triangular form, which is the R matrix in the QR decomposition. This reduction starts at the top left corner of the matrix and proceeds down along the diagonal. If the Householder transformations are multiplied together, then the Q matrix in the decomposition is obtained.

Loosely speaking, an algorithm for a parallel computer is said to have a good load balance if most of the processors are busy doing useful work most of the time. For the computation of matrix decompositions, load balancing is usually achieved by performing the reduction in the standard way, and assigning matrix elements to processors in a clever way, or, alternatively, by performing the reduction in a clever way and assigning matrix elements to processors in the standard way.

The standard way of assigning matrix elements to processors on the CM is based on virtual processors: if the matrix has more elements than there are physical processors, then each matrix element is assigned to a virtual processor. However, each virtual processor is assigned to a physical processor in such a way that a block of matrix elements are stored in the same physical processor.

It is obvious that load balancing cannot be done by using the standard virtual processor assignment, since after some steps in the Householder procedure, more and more of the processors would become idle. We avoid this problem by storing the matrix in a rank-4 array, with two dimensions parallel and two serial. This corresponds to a block cyclic assignment of matrix elements to processors, and using this we can design a block matrix algorithm, where the computations are parallel within the blocks (to a large extent matrix-vector operations), and serial over the blocks. Further, this algorithm can be coded completely in CM Fortran, with data-parallel operations within the blocks. Since the algorithm is serial over the blocks, all processors will be active almost all the time, and the algorithm is load balanced.

This is the kind of parallelisation that also works well on MIMD parallel computers of hypercube type, e.g. *i*PSC/2.

One temporary setback in the work was caused by a odd effect in the compiler: if the serial dimensions were chosen to be the last two in the rank-4 array, then the execution times were about ten times longer than if the serial dimensions were taken to be the first two dimensions. This difficulty was not well presented in the system documentation.

Our results indicate that at present matrix algorithms written in CM Fortran cannot compete in speed with low-level codes, such as the ones in CMSSL. However, using knowledge about the architecture and the compiler, it is possible to get much higher efficiency, than that which would have been obtained by straight forward translation of a serial algorithm into CM Fortran. For more details see [Svensson, 1992, Svensson, 1993].

9.7 Concentrator Location

Olof Damberg
Department of Mathematics, LiTH

We apply an algorithm of N. Z. Shor – space dilation along the difference of two successive subgradients, to a capacitated simple plant location problem – the concentrator location problem. This problem could be described as follows:

A number of terminals are to be connected, via concentrators, to a CPU. It is assumed that the CPU and the terminal locations are known. It is also assumed that the cost of connecting a terminal to a concentrator and the cost of concentrators are known. The concentrator location problem is the problem of determining the number and location of concentrators and allocating terminals among these concentrators without violating capacities of concentrators, at minimum cost. This problem is known to be NP-hard and several solution procedures have been proposed in the literature.

We consider a Lagrangian relaxation approach to the problem, where the subproblems will be essentially semi-assignment problems, which are easily solved in a data parallel fashion. As a dual updating procedure we apply a space dilation subgradient algorithm, which has better convergence properties than ordinary subgradient methods. The drawback, however, is that a dilation

matrix of order $n \times n$, where n is the number of elements in the subgradient, has to be stored and updated. This drawback is much less accentuated on a massively parallel machine where matrix-vector operations are performed very efficiently.

Numerical comparisons between the sequential and parallel algorithm are performed on a Sun SPARC IPX and a 8K CM200 respectively. The largest problem that could be solved on both machines was of size 1024 terminals and 1024 possible concentrator locations. Each main iteration took about 23 seconds (SPARC) and 0.1 seconds (CM200). The largest problem solved on the CM200 was a 4096×4096 problem, and each iteration took about 0.7 seconds. For details see [Damberg and Migdalas, 1994].

10 Computer Science

In 1989, when the CM2 was installed at PDC, many regarded its architecture as strange and predicted that most of its users would come from computer science related research areas. This prediction has turned out not to be true; the bulk of users come from areas that are not computer science related. In fact, rather few of the large number of projects at the center are strictly computer science related.

Three different areas: data-parallel programming languages, parallel logic programming, and architecture simulations

The four current computer science related projects represent three different areas: data-parallel programming languages, parallel logic programming, and architecture simulations. In the language projects (Sections 10.1 and 10.2) a new language model, and analysis techniques, suitable for describing data-parallel computations are presented. This model can act as a guideline for implementing new data-parallel programming languages suitable for scientific computing; the logic project (Section 10.3) presents possible extensions to the logic programming language Prolog, and also their implementation; and finally the architecture project (Section 10.4) presents some aspects of low-level process synchronization primitives of interconnection networks, which are being studied for a future hardware realization.

10.1 Analysis Techniques for Lazy Data-Parallel Functional Programming Languages

Björn Lisper
IT, KTH
Jean-Francois Collard
LIP, France

In order to *implement* lazy functional data-parallel languages efficiently, it is absolutely necessary to employ state-of-the-art techniques for program analysis, parallelization, and code optimization. New techniques that are specific for this class of languages must also be developed and applied.

We have proposed a technique called Extent Analysis [Lisper and Collard, 1994] and it is currently under investigation. The

aim of this analysis is to estimate the "size and shape" (i.e. extent) of a data-parallel entity from its functional specification. Extent Analysis is formulated as a kind of abstract interpretation. Actually, two instances of Extent Analysis can be formulated: the input-output analysis seeks to estimate the maximal extent of a data-parallel entity from the extent of the inputs defining the data field. A simple example is matrix multiplication: given a recursive definition of matrix product (i.e. a functional program), the size of a product matrix AB can be inferred from the sizes of A and B, respectively. The output-input analysis, on the other hand, tries to find what parts of a data-parallel entity will possibly be requested (and thus may possibly need to be computed), given a request for some part of some (possibly other) data-parallel entity. So for instance, if only a small part of the matrix product AB is requested, this analysis can sometimes find what parts of A and B, respectively, are actually needed.

When Extent Analysis succeeds, the result can be used for compile-time allocation of distributed memory and static scheduling of operations on parallel processors. The potential benefit is thus very large.

10.2 Data-Parallel Functional Programming Languages

Per Hammarlund
SANS, NADA, KTH
Björn Lisper
IT, KTH

Data-parallel programming is becoming an increasingly important tool for exploiting parallelism in data-intensive applications, especially on SIMD and vector computers. Many algorithms appearing in such applications can be succinctly expressed in data-parallel languages: this indicates that data-parallel programming can be a powerful abstract programming paradigm rather than just a high-level syntax for explicit programming of SIMD computers. The data-parallel languages in practical use today are, however, exponents of exactly the latter point of view: even though they incorporate some elements of abstraction, their semantics is in every case to some extent based on a SIMD execution model. Therefore it is hard to use these languages to express algorithms in the problem domain in an abstract, machine-independent way. This is

Data-parallel programming, a powerful abstract programming paradigm

likely to make programming in these languages more error-prone and programs less portable than if they had been designed with a more clean-cut abstract semantics.

We have analyzed and evaluated existing data-parallel programming languages and parallel execution techniques [Hammarlund and Lisper, 1992]. Existing data-parallel programming languages are mainly extensions of existing sequential languages, e.g. C* and *LISP are extensions of ANSI C and Common Lisp respectively [TMC, 1991a, TMC, 1991b]. This makes them very specific to the target machine hardware architecture and therefore difficult to port.

Capture the machine-independent aspects of data-parallel programming

In order to capture the more machine-independent aspects of data-parallel programming we have made mathematical definitions of some data-parallel primitives. These can be used to guide the design of data-parallel languages with a higher level of abstraction [Hammarlund and Lisper, 1993]. The key idea is to view data-parallel entities as tabulated functions, where the tables are stored in a distributed fashion. Operations on data-parallel entities are then simply operations on functions, just as operations in pure functional languages. Thus, from a programming perspective there is no difference between the function that generates a data-parallel entity and the data-parallel entity as such. This implies that transformation techniques for functions can be applied to data-parallel entities. An interesting observation is that traditional data structures, like lists and arrays, are also covered by our definitions. This illustrates the level of abstraction achieved.

An especially interesting possibility is to integrate data-parallel and lazy higher order functional languages

An especially interesting possibility is to integrate data-parallel and lazy higher order functional languages. "Lazy data-fields", i.e. data-parallel entities whose entries are computed on demand, are the result. With such fields, many data-parallel algorithms can be succinctly expressed. This is especially true when the domain of computation is irregular and varies strongly with the input. We thus believe that such languages are eminently suitable as specification languages for data-parallel algorithms.

10.3 Massively Parallel Logic Computation

Jonas Barklund, Henrik Arro, Johan Bevemyr
Computing Science Department, Uppsala University

Unlike most other projects carried out on PDC's Connection Machine, our work is research within parallel computation *itself*, rather than on *using* parallel computation for doing research in an application domain.

Advocates of logic programming languages have always claimed that these languages are good for computation on parallel computers, because their semantics is not based on a sequence of changes to a store. Still, imperative languages, such as Fortran 77, have so far been more successful for parallel computation than logic programming languages, even taking into account the results of the Japanese project for Fifth Generation Computer Systems.

The reason for this is that the Single Program Multiple Data (SPMD) model of computation has been successfully applied in these languages, by exploiting parallelism when running definite iterations (so-called *for* or *do* loops). Logic programming languages have mainly attempted to implement more general methods for achieving parallelism; so far these methods have unfortunately had quite significant overheads.

Computer programming languages based on logic usually have recursion as their only means for repetition. Theoretically this is sufficient, and in practice it often works fine. However, for essentially the same reasons that parallelizing so-called *while* loops is difficult, recursion is not easy to run in parallel.

We have therefore turned our attention to an iterative construct for repetition in logic, namely bounded quantification [Barklund, 1994], an example of which is the expression

$$\forall i : 0 \leq i < n \rightarrow a[i] = b[i] + c[i].$$

That a quantification is bounded means that we know *a priori* that we only need to evaluate the body, $a[i] = b[i] + c[i]$ for a finite number of values of the bound variable i, here the values 0 to $n-1$. Seen as a truth-valued statement in a programming language, this expression says that the first n elements of the vector a are the sums of the corresponding elements of the vectors b and c, provided that they are all vectors whose indices go from 0 and upwards. This clearly corresponds to a *do* loop in Fortran 77.

An extension of Prolog

We have designed an extension of the logic programming language Prolog that contains bounded quantifications [Barklund and Bevemyr, 1993]. For example, a goal can be a universal bounded quantification $all(\rho, \beta[\iota])$ where ρ is a *range formula* that restricts some locally scoped variable ι to a finite number of values, and β is a goal, presumably containing ι. A goal can also be an arithmetic bounded quantification $\kappa(\rho, \beta[\iota], \tau)$ where κ is *sum* or *product* (or any other supported quantifier), ρ is as above, β is an arithmetic expression and τ is a variable, which is unified with the sum or product of the values of β. For example, $sum(I\ index\_of\ A, A[I] * A[I],\ R)$ unifies R with the sum of the square of every element of the array A.

An implementation of a data-parallel version of Prolog

Our work on PDC's CM has been to add this construct to an implementation of the logic programming language Prolog that has previously been developed at our department [Bevemyr, 1992]. This gives us a data-parallel version of Prolog where ordinary Prolog expressions are run sequentially, but bounded quantifications are run on the parallel processors of the Connection Machine [Arro *et al.*, 1993].

The implementation is based on a compiler that translates the Prolog programs to sequences of instructions for an abstract machine [Warren, 1983]; the abstract machine is then realized by an emulator. This sequential emulator was written in C; therefore we could extend the machine with parallel instructions for bounded quantifications by adding some data-parallel C* code to the existing C code. This was a relatively modest effort in terms of programmer time. We must conclude that the idea of having a data-parallel language as a conservative extension of a sequential programming language has worked very well.

The work in 1993 has focused on design and implementation of conditional expressions in bounded quantifications, and on implementation of previously designed constructs such as vectors and various range formulas. We have also looked at implementation of bounded quantifications on other data-parallel machines, e.g. the CM5. Most considerations remain the same, because C* provides an execution model that generalizes also to theses machines.

10.4 Barrier Synchronization for Multicomputers

Abdel-Halim Smai, Lars-Erik Thorelli
IT, KTH

Barrier synchronization is an important mechanism for coordinating parallel processes. In shared-memory multiprocessors common memory space is used to implement barrier synchronization. In a distributed memory parallel machine, however, synchronization is accomplished by passing messages between processors.

In most existing multicomputers, wormhole routing is adopted to support the underlying interprocessor communication. Apart from deadlock, a notable source of problem with this technique is congestion. Because a packet is not removed from the network when it is blocked, it can rapidly lead to more blocked packets and therefore to more congestion, especially with large message lengths and network sizes. This situation can be difficult to manage, and system performance can be significantly affected. In particular, this occurs with non-uniform traffic patterns, for instance in the realization of software barriers. There are readily apparent drawbacks in such a case: not only is the delay for a barrier operation increased but it can also affect the communication delay of the other types of messages in the network.

A physical communication channel may be split into multiple virtual channels. A virtual channel is a logical channel which has its own buffer, data- and control-paths. A network can thus provide better node connectivity, and multiple paths and routes for packet transmission are made available. Our first goal is to test and compare different scheduling methods for virtual channels in order to optimize the delay for barrier synchronization operations and to limit the network congestion.

This work is part of the EDA (Extended Dataflow Architecture) project at the IT department. In this study, we are considering some aspects of interconnection networks for a future hardware realization of an architecture supporting the EDA execution models.

We are using C* as our programming language. Its parallel features suit quite well the natural parallelism present in the model being simulated. From the design point of view, C* has been very helpful; performance is still somewhat of a handicap though. A few hours' execution time for a simulation run is common.

The parallel features of C suit quite well the natural parallelism present in the model being simulated*

Glossary

Amdahl's law When using specialized hardware to gain better performance only the part of the computation that can make use of this hardware will show an increase in computing speed. Two examples: for vector architectures only the parts of the code that vectorize will run faster and scalar performance will remain important for the remaining part; for parallel computers only the parts of the computation that parallelizes will gain from using a parallel computer, serial performance will remain important for the rest of the program. [Amdahl, 1967, Gustafson, 1988]

Clusters A clusters is a group of loosely interconnected computers. A typical cluster is a set of "off the shelf" workstations connected via a communication network. A driving force behind developing cluster technology are claims that cheap workstations already on the desks in a typical office can be used as a supercomputer when they are arranged in a cluster and set to work on a problem in parallel. There is free software, e.g. PVM, that allows experimentation with cluster technology today. Software for faster communication, fault tolerance, and processor allocation within the group of "available" workstations are research topics.

HIRLAM, High Resolution Limited Area Model This program is a state-of-the-art analysis and forecast system for numerical weather prediction. The HIRLAM system is being developed within a common research project among the weather services in the Nordic countries, Ireland and the Netherlands. Currently the system is used operationally in Denmark, Finland, the Netherlands, and Sweden. Present implementations of the HIRLAM system can describe scales of motion of the order of 50 km. In order to accurately describe many important weather phenomena it is necessary to describe scales of motion of the order of 1 to 5 km. This would require sustained 10 to 1000 GFlop/s performance. Today vector supercomputers (CRAY, Convex) are used operationally with a moderate degree of parallelization (2–8 processors). In order to achieve the needed performance several initiatives were taken to port HIRLAM to massively parallel computers. Today HIRLAM is running on, among others, MasPar MP-1, CRAY T3D and Intel Paragon.

The HIRLAM forecast model is based on the so called primitive equations. A combination of a terrain-following and a pressure coordinate system is used in the vertical and the forecast model equations are applied on a latitude longitude projection with a shift of the geometrical North pole to minimize the convergence of longitude lines. Finite differ-

ences are used to represent vertical derivatives while in the horizontal direction, finite difference as well as a spectral transform versions are available. A semi-implicit time stepping scheme is utilized for both versions of the model, the linear parts of the adjustment terms are treated implicitly. Schemes for a semi-Lagrangian treatment of the advection terms are being developed for both versions.

CFD, Computational Fluid Dynamics CFD is the study of flow phenomena by computational methods. In fluid dynamics, like in many other branches of natural science, the traditional physical methods of theoretical analysis and experimental observations can now be complemented by computational experiments. Present advances in computer performance and algorithm design allow serious efforts both in basic science, such as the understanding of the properties of the Navier–Stokes equations, and in practical applications to aerospace engineering.

FLOPS, Floating Point Operations Per Second A measure of numerical performance of a computer. It is not uncommon to see people using MFlop/s/s when they mean MFlop/s. The former expression is not formally correct, but will usually not cause any misunderstandings.

FFT, Fast Fourier Transform Any signal can be seen as the superposition of harmonics with different frequencies. The set of amplitudes of the different frequencies is the *Fourier Transform* of the signal. Multiple-variable functions can be decomposed by treating one variable after the other, and we then speak of multi-dimensional Fourier Transforms. When the decomposition is performed on a sampled signal we call it a *Discrete Fourier Transform* (DFT). The actual computation of the DFT can be seen as a complex matrix-vector multiplication where the matrix elements are roots of unity and the vector is the set of N samples. The many symmetry properties of the matrix allows one to perform the calculation in only $O(N \log N)$ arithmetic operations as compared to $O(N^2)$ required for a general matrix-vector multiply, and these algorithms are therefore called *Fast Fourier Transforms* (FFT). FFTs are used extensively in scientific computations. They perform filtering operations in image and signal processing applications, such as speech recognition and synthesis and computed tomography, and they make spectral methods for solving differential equations computationally tractable.

HUGO, Human Genome Project The Human Genome Project aims at creating a database of the human genome, the DNA. This database will be an aid research in medicine. Collecting this information is a formidable task that includes work and development in many different areas – from biotechnology to computer science. As a first step the laboratory has to find the sequences; much of this work has been automated. The sequences are stored in a database. When the information on the sequences has been collected, researchers can start using it. New techniques have to be created for maintaining and using this data.

Load Balancing Good utilization and performance of a parallel computer requires that its processors have nearly the same amount of work. Processors with little work will spend much of their time waiting for the ones with more work. The load balancing problem has both static and dynamical aspects. The static aspect is that one must, already when writing the program or when starting the application, divide the work evenly between processors. The dynamical aspect is that during the computation the problem in itself may cause a situation where the work is unevenly spread on the processors, e.g. galaxy simulations using particle in cell techniques; it may become necessary to redistribute the work during the computation.

MIMD, Multiple Instruction Multiple Data Computer A term describing a specific parallel computer architecture. In a MIMD computer the processing elements (PEs) have their own code and can all perform different instructions on their local data. The MIMD programming model is more general than the SIMD model. Recently there has been a revival of the MIMD computer as the microprocessors have become more powerful – this can be seen in the CM5 from Thinking Machines Corporation, the Intel Paragon, and the CRAY T3D.

NN, Neural Networks Neural Networks is a large field of research where one searches for inspiration in how the nervous system of animals works, i.e. many small simple connected units perform tasks together. What the neural network performs is decided by the simple actions the units perform and how they are connected; all computations are in this sense local. The PDC computers are used for research on both artificial NNs and biologically realistic NNs. Artificial NNs have simplified units and are used to perform, e.g. , multispectral image classification. Biologically realistic NNs use more complex models of the neural network. The models try to capture the precise biological behaviour of the neural system and the programs are used to study possible circuits and their behaviour as accurately as possible.

SHPCNet, Swedish High-Performance Computing Network The intention of the project is to connect the Swedish supercomputer centers in Linköping (CRAY), Stockholm (CM200) and Skellefteå (IBM 3090) with higher speed, 34 Mbit/s, links. This is a first step towards *realistic distributed supercomputing*. Apart from a number of projects that aim directly at the distributed possibilities, it is also reasonable to foresee a more efficient sharing of existing computer resources among high-performance users in Sweden.

SIMD, Single Instruction Multiple Data Computer The acronym SIMD describes a specific parallel computer architecture. In a SIMD computer instructions are broadcast to all processing elements (PEs) from the control processor. The PEs all perform the *same* instruction on their own local data, but may optionally decide to skip one based

on local data. One reason for building computers like this is that the architecture reduces the complexity of the PEs – they do not have to have local program code and hardware for parsing this code. The fact that all PEs perform the same instruction is a restriction in the programming model; a SIMD computer can however perform any operation a MIMD computer can, it only increases computing time by a constant factor. A SIMD computer will usually have more PEs than a MIMD computer.

Speedup and Scaled Speedup Parallel computers enables us to execute problems faster or to execute larger problems within reasonable time. Sometimes we can run larger problems faster. It is not obvious how we should compare this performance increase. Two measures have emerged that capture different aspects: speedup, how much faster does the problem run if we throw additional processor at it; scaled speedup, how does the computer perform if we increase both problem and machine size. *Speedup* can be defined as the time for running a problem of a certain size on a single processor divided by the time for running the same problem size on many processors. *Scaled Speedup* can be defined as the time it would take, if it was possible, to execute a large problem on a single processor of parallel machine divided by the time to run that problem on many processors. Speedup, loosely speaking, captures how much overhead the machine adds when we use more processors and scaled speedup measures if all the parts of the machine's architecture scales with increased machine size. [Worley, 1990, Nussbaum and Agarwal, 1991, Singh *et al.*, 1993]

SPMD, Single Program Multiple Data One can loosely define SPMD as the programming model where one has a single program that operates in parallel on several sets of data, e.g. a CFD code communicating with message passing. This programming model can be used on many different architectures.

Bibliography

[Amdahl, 1967] Amdahl, G. M. (1967). Validity of the single-processor approach to achieving large scale computing capabiliites. In: *AFIPS Conference Proceedings*, volume 30, 1967, p. 483–485. 92

[Andersson *et al.*, 1992] Andersson, J.-O., Mattsson, J., and Svedlindh, P. (1992). *Physical review. B. Solid state* **46**: 8297. 56

[Andersson *et al.*, 1994] Andersson, J.-O., Mattsson, J., and Svedlindh, P., (1994). appeared in the January 1, issue of Phys. Rev. B. 56

[Andersson and Sibani, 1993] Andersson, J.-O. and Sibani, P., (1993). Monte Carlo studies of domain growth in short range Ising spin-glasses. In preparation. 57

[Arro *et al.*, 1993] Arro, H., Barklund, J., and Bevemyr, J. (1993). Parallel Bounded Quantifications—Preliminary Results. *ACM SIGPLAN Notices* **28**: 117–124. 90

[Aurell *et al.*, 1994] Aurell, E., Frick, P., and Shaidurov, V. (1994). Hierachical tree-model of 2D-turbulence. *Physica D* **D72**: 95–109. 48

[Aurell *et al.*, 1993] Aurell, E., Gurbatov, S., and Wertgeim, I., (1993). 79

[Barklund, 1994] Barklund, J. (1994). Bounded Quantifications for Iteration and Concurrency in Logic Programming. *New Generation Computing* **12**: 161–182. 89

[Barklund and Bevemyr, 1993] Barklund, J. and Bevemyr, J. (1993). Prolog with Arrays and Bounded Quantifications. In: *Proceedings of the 4th Intl. Conf. on Logic Programming and Automated Reasoning*, edited by V. Andrei, LNCS, Berlin. Springer-Verlag. 90

[Berglund, 1994] Berglund, M. (1994). Parallel Computer Simulation of Ground Vibrations. Technical Report TRITA-NA-9408, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, S-100 44 Stockholm, Sweden. 70

[Bevemyr, 1992] Bevemyr, J. (1992). The Luther WAM Emulator. UPMAIL Technical Report 72, Computing Science Department, Uppsala University. 90

[Binder and Privman, 1992] Binder, P. and Privman, V. (1992). *Mod. Phys. Lett. B* **6**: 1835. 54

[Bower *et al.*, 1990] Bower, R., Tamayo, P., and York, B., (1990). A Parallel Multigrid Clustering Algorithm for Percolation Clusters. 59

[Brunak *et al.*, 1991] Brunak, S., Engelbrecth, J., and Knudsen, S. (1991). Prediction of human mRNA donor and acceptor sites from the DNA sequence. *Journal of Molecular Biology* **220**: 49–65. 63

[Chaté and Manneville, 1991] Chaté, H. and Manneville, P. (1991). *Europhys. Lett.* **14**: 409. 54

[Damberg and Migdalas, 1994] Damberg, O. and Migdalas, A. (1994). A Massively Parallel Space Dilation Algorithm for the Concentrator Location Problem. Working paper, Department of Mathematics, Linköping Institute of Technology, S-581 83 Linköping. 85

[Ekeberg *et al.*, 1993] Ekeberg, Ö., Hammarlund, P., Levin, B., and Lansner, A. (1993). SWIM — A Simulation Environment for Realistic Neural Network Modeling. In: *Neural Network Simulation Environments*, edited by J. Skrzypek. Kluwer. (In press). 22

[Engström, 1993] Engström, S. (1993). *Studies of spiral structure in disk galaxies*. PhD thesis, Chalmers University of Technology, Göteborg, Sweden. 46

[Erlingsson, 1993] Erlingsson, S. (1993). *Dynamic Soil Analysis with an Application to Rock Music Induced Vibrations in Ullevi Stadium*. PhD thesis, Royal Institute of Technology, Stockholm, Sweden. 70

[Fincham, 1993] Fincham, D. (1993). *CCP5 Information Quarterly* 17–24. 66

[Fischer and Hertz, 1991] Fischer, K. and Hertz, J. (1991). *Spin Glasses*. Cambridge University Press. 55, 56

[Fisher and Huse, 1988] Fisher, D. S. and Huse, D. A. (1988). *Physical review. B. Solid state* **38**: 373. 56

[Fransén and Lansner, 1994] Fransén, E. and Lansner, A., (1994). Low Spiking Rates in a Population of Mutually Exciting Pyramidal Cells. (Submitted). 23, 23

[Fransén *et al.*, 1993] Fransén, E., Lansner, A., and Liljenström, H. (1993). A Model of Cortical Associative Memory based on Hebbian Cell Assemblies. In: *Computation and Neural Systems*, edited by F. H. Eeckman and J. M. Bower, Boston, MA. Kluwer, p. 431–435. 23

[Glansdorff and Prigogine, 1975] Glansdorff, P. and Prigogine, I. (1975). *Thermal Theory of Structure Stability and Fluctuations*. Wiley-Interscience Publishers. 58

[Gould and Tobochnik, 1988] Gould, H. and Tobochnik, J. (1988). *An Introduction to Computer Simulation Methods, Part 2*. Addison-Wesley Publishing Company. 58

[Gradin and Ledfelt, 1993] Gradin, U. and Ledfelt, G. (1993). Computational Electromagnetics in 2D. Technical Report TRITA-NA-E9338, NADA, KTH, Stockholm, Sweden. 35, 36

[Grinstein *et al.*, 1993] Grinstein, G., Mukamel, D., Seidin, and Bennett, C. (1993). *Phys. Rev. Lett.* **70**: 3607. 54

[Gustafson, 1988] Gustafson, J. L. (1988). Reevaluating Amdahls's Law. *Communications of the ACM* **31**: 532–533. 92

[Gustafsson, 1981] Gustafsson, B. (1981). The convergence rate for difference approximations to general mixed initial boundary value problems. *SIAM Journal on Numerical Analysis* **18**: 179–190. 69

[Gustafsson and Lindskog, 1992] Gustafsson, I. and Lindskog, G. (1992). Complete parallelizable preconditioning methods. Report 19, Department of Computing Science, Chalmers University of Technology, Göteborg, Sweden. To appear in Numerical Linear Algebra with Applications. 81

[Gustafsson, 1991] Gustafsson, N. (1991). The HIRLAM model. In: *Seminar on Numerical Methods in Atmospheric Models*. ECMWF, Reading, UK, 1991. 41

[Hammarlund *et al.*, 1991] Hammarlund, P., Levin, B., and Lansner, A. (1991). BIOSIM—A Program for Biologically Realistic Neural Network Simulations on the Connection Machine. In: *Proceedings of ICANN-91*, edited by T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, Amsterdam. North-Holland, p. 1477–1480. Espoo, Finland, June 24-28, 1991. 21

[Hammarlund *et al.*, 1992a] Hammarlund, P., Levin, B., and Lansner, A. (1992a). Biologically Realistic and Artificial Neural Network Simulators on the Connection Machine. In: *Science on the Connection Machine*, edited by T. Lippert, K. Schilling, and P. Ueberholz. World Scientific, 1992, p. 49–63. (Proceedings of the First European CM Users Meeting, June 16–17). 21

[Hammarlund *et al.*, 1992b] Hammarlund, P., Levin, B., and Lansner, A. (1992b). BIOSIM—A SIMD Parallel Simulator for Biologically Realistic Simulations of Neural Networks. (Submitted.). 21

[Hammarlund and Lisper, 1992] Hammarlund, P. and Lisper, B. (1992). Data Parallel Programming—A Survey and a Proposal for a New Model. Technical report, TDS—Department of Telecommunication and Computer Systems, Royal Institute of Technology, S–100 44 Stockholm, Sweden. 88

[Hammarlund and Lisper, 1993] Hammarlund, P. and Lisper, B. (1993). On the Relation between Functional and Data Parallel Programming Languages. Accepted for publication at the Sixth Conference on Functional Programming Languages and Computer Architecture. 88

[Hedman and Laaksonen, 1993] Hedman, F. and Laaksonen, A. (1993). A Data-parallel Molecular Dynamics Method for Liquids with Coulombic Interactions. *Molecular Simulation* , Accepted for publication. 66

[Hemmingsson and Herrmann, 1993] Hemmingsson, J. and Herrmann, H. (1993). *Europhys. Lett.* **23**: 15. 54

[Hemmingsson and Peng, 1994] Hemmingsson, J. and Peng, G. (1994). *J. Phys. A: Math. Gen.* **27**: 2735. 55

[Hemmingsson, 1993] Hemmingsson, L. (1993). Toeplitz preconditioners with block structure for first-order PDEs. Report 156, Dept. of Scientific Computing, Uppsala University, Sweden. 80

[Hemmingsson and Otto, 1994] Hemmingsson, L. and Otto, K., (1994). Analysis of semi-Toeplitz preconditioners for first-order PDE. Submitted to SIAM J. Sci. Comput. 80

[Hib, 198] Hibbitt Karlson & Sorensen Inc. (198?). *ABAQUS User Manual. Version 4.8.*, 198? 70

[Hillis and Boghosian, 1993] Hillis, W. and Boghosian, B. (1993). *Science* **261**: 856–863. 57

[Holmgren and Otto, 1992] Holmgren, S. and Otto, K. (1992). Iterative solution methods and preconditioners for block-tridiagonal systems of equations. *SIAM J. Matrix Anal. Appl* **13**: 863–886. 80

[Holmgren and Otto, 1994] Holmgren, S. and Otto, K. (1994). Semi-circulant preconditioners for first-order PDE. *SIAM J. Sci. Comput* **15**: 385–407. 80

[Holst and Lansner, 1993] Holst, A. and Lansner, A. (1993). A Bayesian neural Network Model with Extensions. Technical Report TRITA-NA-P9325, NADA, Royal Institute of Technology, Stockholm, Sweden. 20

[Jackson and Persson, 1992] Jackson, B. and Persson, M. (1992). A quantum mechanical study of recombinative desorption of atomic hydrogen on a metal surface. *J. Chem. Phys.* **96**: 2378. 52

[Johnsson and Ho, 1989] Johnsson, S. and Ho, C. (1989). Embedding Hyper-Pyramids into Hypercubes. *SIAM J. Sci. Stat. Comput.* **10**: 607–630. 49

[Karlsson and Goscinski, 1994] Karlsson, H. and Goscinski, O. (1994). A Direct Recursive Residue Generation Method. Application to Photoionization of Hydrogen in Static Electric Fields. *Journal of Physics B* **27**: 1061–1072. 67

[Kolafa and Perram, 1992] Kolafa, J. and Perram, J. W. (1992). Cutoff Errors in the Ewald Summation Formulae for Point Charge Systems. *Molecular Simulation* **9**: 351–368. 66

[Lander *et al.*, 1991] Lander, E. S., Langridge, R., and Saccocio, D. M. (1991). Mapping and Interpreting Biological Information. *Communications of the ACM* **34**: 32–39. 60

[Lansner and Ekeberg, 1989a] Lansner, A. and Ekeberg, Ö. (1989a). A One-Layer Feedback Artificial Neural Network with a Bayesian Learning Rule. *International Journal of Neural Systems* **1:1**: 77–87. 20

[Lansner and Ekeberg, 1989b] Lansner, A. and Ekeberg, Ö. (1989b). A One-layer Feedback, Artificial Neural Network with a Bayesian Learning Rule. *Int. J. Neural Systems* **1**: 77–87. 23

[Lansner and Fransén, 1994] Lansner, A. and Fransén, E. (1994). Improving the Realism of Attractor Models by Using Cortical Columns as Functional Units. In: *Computation and Neural Systems*, edited by F. H. Eeckman and J. M. Bower, Boston, MA. Kluwer. (to appear). 23

[Lee and Kim, 1991] Lee, M. and Kim, J. (1991). The structure of turbulence in a simulated plane Couette flow. In: *8th Symposium on Turbulent Shear Flow, Münich*, 1991. 37

[Levander, 1988] Levander, A. (1988). Fourth order finite difference P-SV seismograms. *Geophysics* **53**: 1425–1436. 72

[Levin *et al.*, 1990] Levin, B., Hammarlund, P., and Lansner, A. (1990). BIOSIM—A Program for Biologically Realistic Neural Network Simulations on the Connection Machine. Tech. Rep. TRITA-NA-9021, Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden. 21

[Levin and Lansner, 1992] Levin, B. and Lansner, A. (1992). Document Retrieval, Protein Sequence Matching and Sensor Selection Methods using a Neural Network. Technical Report TRITA-NA-P9238, NADA, Royal Institute of Technology, Stockholm, Sweden. 20

[Liljenström, 1991] Liljenström, H. (1991). Modelling the Dynamics of Olfactory Cortex Using Simplified Network Units and Realistic Architecture. *International Journal for Neural Systems* **2**: 1–15. 26

[Lindskog and Gustafsson, 1993] Lindskog, G. and Gustafsson, I. (1993). Some experiences on the CM-200 in the solution of a numerical linear algebra problem. Report 26, Department of Computing Science, Chalmers University of Technology, Göteborg, Sweden. Talk at Second European CM Users meeting at Observatoire de Paris-Meudon, Oct 11-14, 1993. 82

[Lisper and Collard, 1994] Lisper, B. and Collard, J.-F. (1994). Extent Analysis of Data Fields. Accepted for presentation at the International Symposium on Static Analysis, September 28–30 1994, Namur, Belgium. 86

[Loewenthal *et al.*, 1991] Loewenthal, D., Wang, C., Johnson, O., and Juhlin, C. (1991). High order finite difference modeling and reverse time migration. *Exploration Geophysics* **22**: 533–545. 72

[Lundbladh *et al.*, 1992] Lundbladh, A., Henningson, D., and Johansson, A. (1992). An efficient spectral integration method for the solution of the Navier-Stokes equation. Technical Report FFA-TN 1992-28, Aeronautical Research Institute of Sweden, Bromma. 36

[Luo *et al.*, 1991] Luo, W., Nagel, S. R., Rosenbaum, T. F., and Rosensweig, R. E. (1991). *Physical review letters* **67**: 2721. 55

[Munthe-Kaas, 1993] Munthe-Kaas, H., (1993). Super Parallel FFTs. (to appear in SIAM J. on Scientific and Stat. Comput.). 42

[Noullez, 1992] Noullez, A. (1992). A Fast Algorithm for discrete Legendre transforms. Technical report, Observatoire de Nice. 79

[Noullez and Vergassola, 1993] Noullez, A. and Vergassola, M. (1993). A Fast Legendre Transform algorithm and applications to the adhesion model. Technical report, Observatoire de Nice. Submitted to Journal of Scientific Computing. 79

[Nussbaum and Agarwal, 1991] Nussbaum, D. and Agarwal, A. (1991). Scalability of Parallel Machines. *Communications of the ACM* **34**: 57–61. 95

[Orzag, 1970] Orzag, S. (1970). Transform method for calculation of vector-coupled sums. Application to the spectral form of the vorticity equatio. *J. Atmos. Sci.* **27**: 890–895. 41

[Palm, 1993] Palm, T. (1993). Self-consistent calculations of an electron-wave Y-branch switch. *J. Appl. Phys.* **74**: 3551–3557. 49

[Pernica, 1988] Pernica, G. (1988). Dynamic live loads at a rock concerts. *Canadian Journal of Civil Engineering* **10**: 185–191. 71

[Per, 1991] (1991). Time-Dependent Methods for Quantum Dynamics. Comp. Phys. Comm. See articles in volume 63. 51

[Persson and Jackson, 1993] Persson, M. and Jackson, B., (1993). A flat surface model study of Eley-Rideal dynamics of recombinative desorption of hydrogen on a metal surface. Submitted to J. Chem. Phys. 51

[Pomeau, 1993] Pomeau, Y. (1993). *J. Stat. Phys.* **70**: 3607. 54

[Roland *et al.*, 1993] Roland, P., Levin, B., Kawashima, R., and Åkerman, S. (1993). Three Dimensional Analysis of Clustered Voxels in ^{15}O-Butanol Brain Activation Images. *Human Brain Mapping* **1**: 3–19. 64

[Rosensweig, 1985] Rosensweig, R. (1985). *Ferrohydrodynamics*. Cambridge University Press. 55

[Rots, 1990] Rots. (1990). *Astrophysical Journal* **178**: 623. 45

[Sahlin, 1989] Sahlin, S. (1989). On Site Measurements of Soil and Structure Response of the Soccer Stadium "Nya Ullevi" in Göteborg. In: *A course in Fundamentals of Earthquake Engineering 1989*. Statens Provningsanstalt, 1989. 71

[Sawley, 1993] Sawley, M. (1993). Control- and Data-Parallel Methodologies for Flow Calculations. In: *Supercomputing Europe '93*, edited by R. Tucker, Utrecht. Royal Dutch Fairs, p. 169–187. 38, 38

[Sawley and Bergman, 1994] Sawley, M. and Bergman, C. (1994). A Comparative Study of the Use of the Data-Parallel Approach for Compressible Flow Calculations. *Parallel Computing* **20**: 363–373. 38, 39

[Sawley *et al.*, 1993a] Sawley, M., Tegnér, J., and Bergman, C. (1993a). A serial data-parallel multi-block method for compressible flow computations. Technical Report T-93-22, IMHEF, EPFL. To appear in Proceedings of Parallel CFD '93 (Paris, May 1993). 39

[Sawley *et al.*, 1993b] Sawley, M., Tegnér, J., Leyland, P., and Bomholt, L. (1993b). Computational Fluid Dynamics: parallelism, portability and performance. *Speedup Journal* **7**: 64–71. 40

[She *et al.*, 1992] She, Z., Aurell, E., and Frisch, U. (1992). The Inviscid Burgers Equation with Initial data of Brownian type. *Communications in Mathematical Physics* **148**: 623–641. 79, 79

[Sibani and Andersson, 1993] Sibani, P. and Andersson, J.-O., (1993). Excitation morphology of short range Ising spin-glasses. Preprint. 57

[Singh *et al.*, 1993] Singh, J. P., Hennessy, J. L., and Gupta, A. (1993). Scaling Parallel Programs for Multiprocessors: Methodology and Examples. *IEEE Computer* 42–50. 95

[Smith and Zipser, 1989] Smith, C. and Zipser, D. (1989). Learning Sequential Structure with the Real-time Recurrent Learning Algorithm. *International Journal of Neural Systems* **1**: 125–131. 63

[Soldal *et al.*, 1993] Soldal, O., Rye, N., Thunvik, R., and Halvorsen, E. (1993). Field investigations and modeling of average hydraulic gradient in a coastal aquifer, Sunndalsøra, Norway. NHP Report 35, Nordic Hydrological Programme. 73

[Svensson, 1992] Svensson, G. (1992). Matrix computations on the CM-200. Technical report, Department of Mathematics, Linköping University. 84

[Svensson, 1993] Svensson, G. (1993). Matrix Computations on Parallel Computers. Licentiat Avhandling LiU-TEK-LIC-1993:26, Department of Mathematics, Linköping University. ISBN: 91-7871-128-2, ISSN: 0280-7971. 84

[Thi, 1993] Thinking Machines Corporation. (1993). *CMSSL for CM Fortran. Version 2.2*, 1993. 76

[TMC, 1991a] TMC, Thinking Machines Corporation, 245 First Street, Cambridge, Massachusetts 02142–1264. (1991a). *Connection Machine: Programming in C**, 6.1 edition, 1991. 88

[TMC, 1991b] TMC, Thinking Machines Corporation, 245 First Street, Cambridge, Massachusetts 02142–1264. (1991b). *Connection Machine: Programming in *Lisp*, 6.1 edition, 1991. 88

[Toomre and Toomre, 1972] Toomre and Toomre. (1972). *Astrophysical Journal* **100**: 387. 45

[Wallin, 1992] Wallin, E. (1992). Optimized Sequence Matching on the CM-2. Master's thesis, Royal Institute of Technology, Stockholm, Sweden. 20

[Warren, 1983] Warren, D. H. D. (1983). An Abstract Prolog Instruction Set. SRI Technical Note 309, SRI International, Menlo Park. 90

[Wilhelmsson, 1994] Wilhelmsson, T. (1994). Simulating the Dynamics of Olfactory Cortex on the Connection Machine. Master's thesis, Linköping Institute of Technology, Linköping, Sweden. In preparation. 24

[Williams and Zipser, 1989] Williams, R. and Zipser, D. (1989). Experimental Analysis of the Real-time Recurrent Learning Algorithm. *Connection Science.* **1**: 87–111. 63

[Worley, 1990] Worley, P. H. (1990). The Effect of Time Constraints on Scaled Speedup. *SIAM Journal on Scientific and Statistical Computing* **11**: 838–858. 95

[Załuska-Kotur and Cieplak, 1993] Załuska-Kotur, M. A. and Cieplak, M. (1993). *Europhysics letters* **23**: 85. 56

Index