COMMON ELEMENTS OF THE VARIOUS DISCIPLINES

IN COMPUTATIONAL SCIENCE

Talk at Symposium on Computational Physics

Swiss Physical Society, Fall Meeting 1987

R. Henzi, University of Berne

ABSTRACT

The advent of parallel computers will have an
important influence on many areas of science, just
as have other new tools, such as particle
accelerators, neutron sources, large telescopes, or
space labs. Its great advantage is its general
applicability to all branches of science. The
computational scientist is faced with fundamentally
new challenges because the high rate of computation
is only fully realized if the algorithms and
programming techniques are tailored to the
architecture of the computer. This is illustrated
by an example of vectorization and optimization of
an IF-loop. Two classes of examples are described
where important new scientific or engineering
results are being obtained by large-scale computer
simulation: simulation of classical or
pseudo-classical many-body systems, and of
statistical and quantum systems. Common
computational techniques in these classes are
described. Many other examples of application areas
of supercomputers are given, and the case is made
that in the computational approaches to these
pressing and challenging problems the mathematical
and computational aspects cannot and should not be
considered in isolation. A new forum of
collaboration of scientists in Switzerland, SPEEDUP,
is presented. It forms an interface between
advanced computers, such as parallel computers, and
the physicist or engineer with the problem.

1. INTRODUCTION

Current nominal computational performance rates of the types of supercomputers used in large-scale simulations in science and engineering are of the order of 100s of MFLOPS (an MFLOP is 1 million floating point operations per second). The supercomputers available during the early 1990s will have rated performances in the 10,000 MFLOP range. The mid 1990s machines are forecast to be 300,000 MFLOP machines, and those of the turn of the millenium should be over 1,000,000 MFLOPs in performance. (Figure 1.)

These computers obtain their speed principally by an increased use of what is called PARALLELISM, more precisely, vector and array parallelism, and are collectively termed parallel computers.

Applications on machines of such enormous power will be themselves tremendously large software systems developed over many years by organized groups of scientists and engineers.

The advent of such powerful computational tools will have an important influence on many areas of science, just as have other new tools, such as particle accelerators, neutron sources, large telescopes, or space labs. (Section 2.) Its great advantage is its general applicability to all branches of science.

In comparison with his work on the classical serial "FORTRAN machines", the computational scientist is faced with fundamentally new challenges because the high rate of computation is only fully realised if the algorithms and programming techniques are tailored to the architecture of the computer. (Section 3.)

To show the types of scientific problem to which parallel computers can profitably be applied, we describe two classes of examples. The first class is the simulation of classical or pseudo-classical many-body systems, where we find examples such as the simulation of spiral structure in galaxies of stars, the simulation of the flow of electrons in semiconducting devices, and the simulation of melting and glass formation in ionic materials. The second class is the simulation of statistical and quantum systems, where we find examples such as the simulation of a highly idealized model, e.g. the Ising model in equilibrium or far from equilibrium, or the simulation of lattice gauge theories, the simulation of continuum and lattice models of polymers, and the simulation of atomic models of quantum liquids and solids. For each of these two classes a common computational technique (the particle-mesh method in the first class, the Monte Carlo method in the second class) has been developed and is used in all the diverse applications. (Sections 4 and 5). This makes apparent the importance of supporting research in Computational Science, as an activity in its own right, in order to develop methods suited to the new parallel computers.

We have only given two classes of examples where a parallel computer can have an important effect on the advancement of science and engineering. Other examples abound in e.g. hydrodynamic systems, chemical systems and combustion, plasma
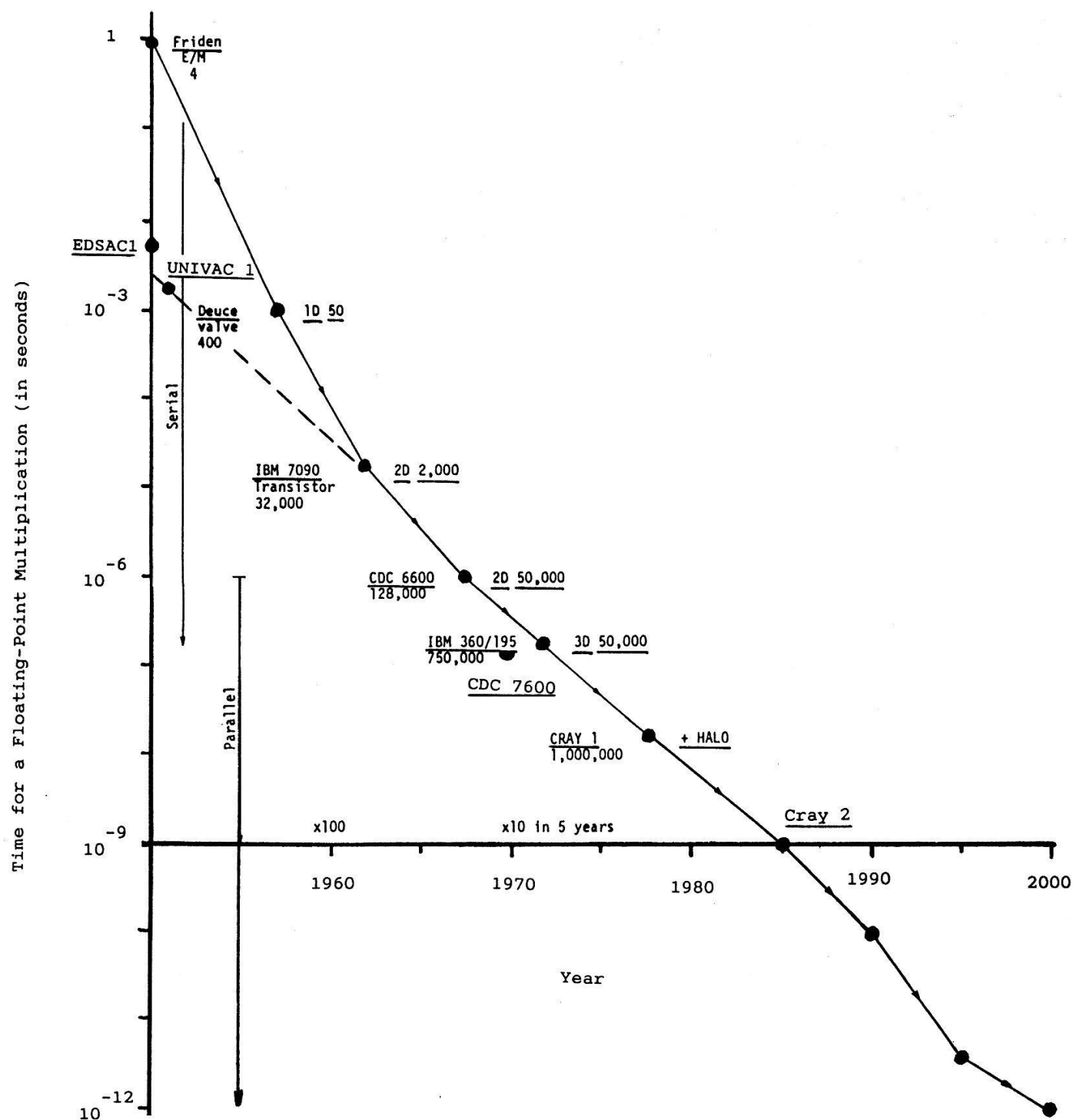
Figure 1.   History (since 1950) and Forecast of Computer
Arithmetic Speed.

physics, geophysical applications, meteorology, structural
mechanics, nondestructive testing and tomographic
reconstruction, and mathematical models in the biological
sciences. In the computational approaches to these pressing and
challenging problems the mathematical and computational aspects
cannot and should not be considered in isolation. (Section 6).

A Cray 1 was installed at the Swiss Federal Institute of
Technology Lausanne in early 1986. A national initiative for
supercomputing in Switzerland [Hochstrasser 88] was voted by
Swiss Parliament in Summer of 1986. Early installation of more
powerful supercomputers is also necessary in this framework and
is eagerly awaited. Currently, a transitory solution is being
considered in which a Cray X-MP at the Swiss Federal Institute
of Technology Zurich and a Cray 2 at the Swiss Federal Institute
Lausanne would bridge the time till a supercomputer of the
latest generation becomes available.

Vast amounts of computer power will be available for users of
the new Swiss supercomputer system. Effective distribution of
these resources presupposes the use of

- advanced, high level, high-bandwidth communication
  facilities, and

- a very large user base.

Support of this large user base also presupposes the maintenance
of a substantial infrastructure of general purpose computer
facilities providing

- job preparation,

- administration, and

- general computer service support.

It is likely that such an infrastructure would have to be
distributed in structure, if only because of the complexity of
the service environment.

The real question we face is: how should this entire
supercomputing environment be conceived to turn raw computer
power into advances in human knowledge and engineering?
(Section 7.)

2. THE INFLUENCE OF COMPUTER SPEED ON THE COMPLEXITY OF
    PHYSICAL PROBLEMS THAT CAN BE SOLVED

Figure 1 shows the time to perform a floating-point multiply on
a number of computers since 1950, together with the size of
high-speed memory, and the above-metioned forecast to the year
2000. In recent years there has been an increase of a factor of
10 in speed every 5 years. This has been obtained partly by
increases in the speed of logic elements and partly by the
introduction of more parallelism in the architecture of the
computer.

Let us consider that a computer run of a few hours is the
maximum a single user can reasonably perform on an installation.

Then, in the study of galactic evolution, the first valve
computers would have been limited to the study of about 50 stars
in a rather unrealistic one-dimensional galaxy. The first
transistor computers, such as the IBM 7090, allowed the first
realistic simulations of 2000 stars moving in a plane. The CDC
6600 allowed the number of stars to be increased to 50 000. But
the move to full three-dimensional motion had to await the
arrival of the CDC 7600 and IBM 360/195. In the above
simulations only the disk stars are simulated and the halo
stars, that lie out of the plane of the galactic disc, are
represented as a fixed external gravitational field. The
opportunity to represent moving stars in the halo requires the
extra computer speed of the parallel computer, as exemplified by
the Cray 1 computer.

3. PROGRAMMING VECTOR AND OTHER PARALLEL COMPUTERS AND ITS IMPLICATION FOR INTERDISCIPLINARY GROUPINGS

In large-scale simulation codes, the calculation is best
performed entirely in high-speed memory. The calculation time
is determined primarily by the time to perform floating-point
arithmetic because in any carefully written code accesses to and
from memory, index calculations and house-keeping calculations
can be performed in parallel with the floating-point arithmetic.
Provision for doing this exists on most top-of-the-range
machines (e.g. IBM 3090, Crays, Cybers, Fujitsus).
Accordingly, such organizational operations may be ignored and
the performance of computers may be measured in terms of the
number of millions of floating-point operations performed per
second. For this purpose, all floating-point additions,
subtractions, multiplications and divisions are counted with
equal weight. (To distinguish separately between the different
times for these operations would complicate matters without
changing the conclusions much.)

Normally, any standard ANSI FORTRAN 77 program can be executed
on a vector computer, such as the Cray 1, Cray X-MP, Cray 2,
Cyber 205, ETA 10 or Fujitsu, through a compiler which
recognizes DO-loops that can be executed by the explicit vector
instructions of the machine. Existing FORTRAN programs, which
were written without knowledge of the machine architecture, will
in many cases run faster than on the IBM 3090. But remember,
running a program on a vector processor does not necessarily
mean that one uses the vector instructions of the machine - the
real power of these new computers. The full vector speeds, in
terms floating point operations per second, of 5 to 25 times
that of the scalar speed can be obtained in FORTRAN provided
simple rules, designed to permit the compiler to vectorize, are
obeyed. This increased performance gain results in increased
motivation to perform the optimization of programs for these
machines by applying these rules.

Another issue which should motivate the programmer is that
vector and some multi-processors are here to stay in scientific
computing, and we will probably even see them in other areas of
computing, e.g. transaction processing. The vector processor
is not a passing fad, and the scientific computer programmer
should learn how to use it.

As examples of the type of vectorization and optimization work

referred to above, we mention three that have recently been
completed within the framework of the PARCOM Project on
Scientific Applications and Algorithm Design for Parallel
Computing at the University of Berne.   These involve Monte Carlo
updating [Decker and Henzi 87; Decker 87], problems to test
parallel computers [Henzi 88a], and inverse problems [Henzi
88b].

For illustration, let us vectorize an example which used to be
considered non-vectorizable. All too often people cop out on a
little work by saying, "This code isn't conductive to
vectorization." In this IF-loop there is a good calculation we
would like to vectorize:

```
          M=0
 100   CONTINUE
          M=M+1
          A(M)=B(M)**2 + .5*C(M)*D(M)/E(M)
       IF (A(M).GT.0) GO TO 100
```

The problem we have is that we are not sure when the IF-loop,
whose timing is illustrated in Fig. 2a, is terminated. Split
the calculation out from the decision:

```
       DO 21  M=1,N
          TEMP(M)=B(M)**2 + .5*C(M)*D(M)/E(M)
 21    CONTINUE
          M=0
 100   CONTINUE
          M=M+1
          A(M)=TEMP(M)
       IF (A(M).GT.0) GO TO 100
```

This restructuring will not run faster on a scalar machine;
however, it should on a vector computer. Of course, as is
illustrated in Fig. 2b, a lot depends on which value of A is
the first not to satisfy the test. Quite often a DO-loop may
vectorize; however, it could be written to vectorize even
better. On the Cray we get best performance for a vector length
of 64. Therefore, we stripmine this example and use the ILLZ
function from SCILIB to find the first occurrence where the A
array is less than zero:

```
       DO 21  I=1,N,64
          LENGTH=MINO(64,N-I+1)
          DO 20  MM=1,LENGTH
             M=I+MM-1
             TEMP(MM)=(B(M)**2+.5*C(M)*D(M)/E(M))-1.E-1000
 20       CONTINUE
          NTOP=ILLZ(LENGTH,TEMP,1)
          DO 31  MM=1,NTOP
             M=I+MM-1
             A(M)=TEMP(MM)
 31       CONTINUE
          IF(NTOP.LT.LENGTH) GO TO 22
 21    CONTINUE
 22    CONTINUE
```

The corresponding timing is illustrated in Fig. 2c. Figures
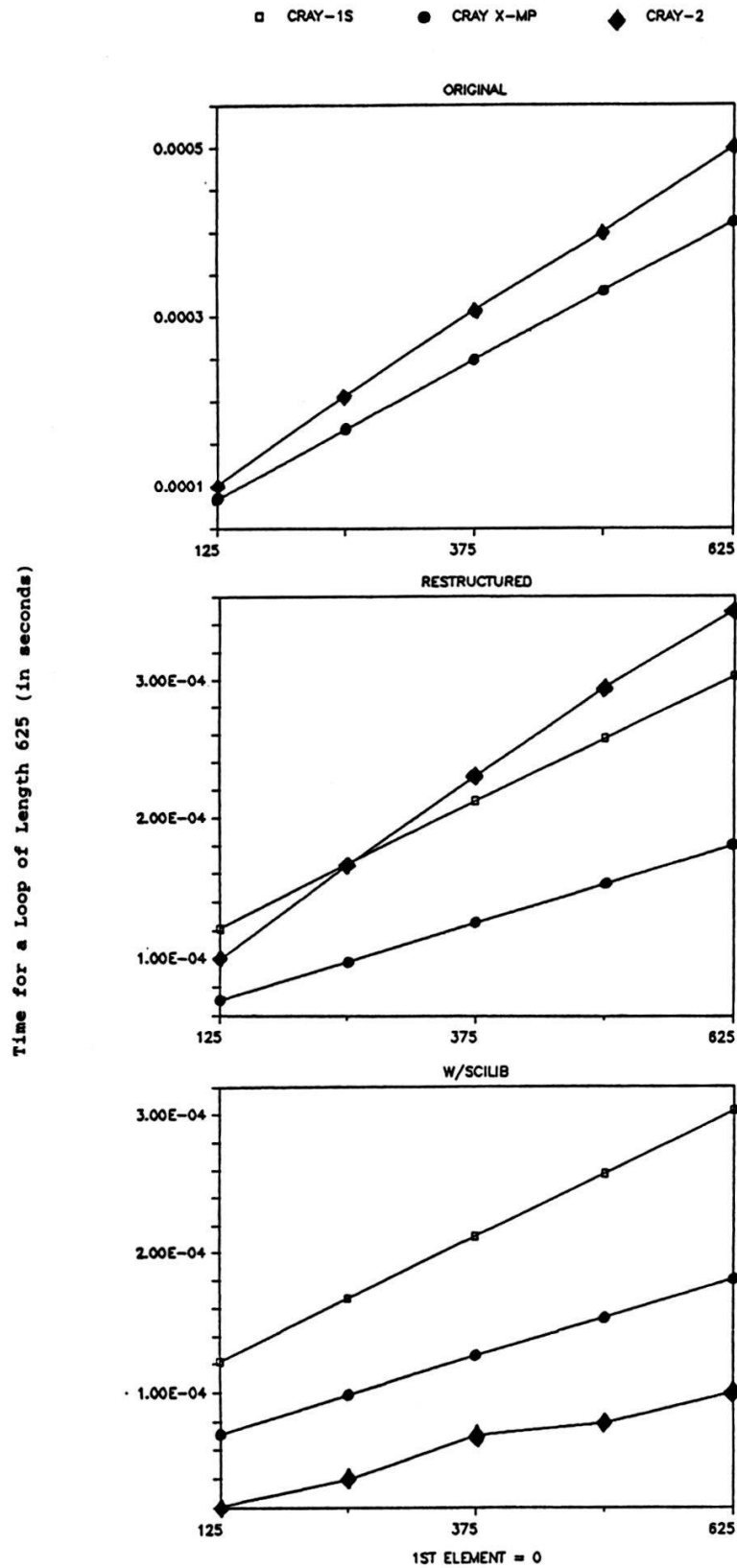2a-c also illustrate that the performance gain due to a code

Figure 2.   Example of Performance Gain Due to Vectorization
and Optimization.
(a) Original, (b) Restructured, (c) With SCILIB.

restructuring can not only be quite machine dependent but also
quite considerable, as e.g. for the Cray 2. On the one hand,
this makes apparent that optimization can be profitable. On
the other hand, optimization of a code for large-scale
simulation can entail a considerable investment of human
resources. But then it becomes interesting to exploit such an
optimized code by applying it to simulate as many different
physical systems as possible, optimizing in this way that
investment. This in turn favors the formation of a
computational community in the form of an interdisciplinary
grouping based on the common underlying mathematics and
algorithmics of research areas. Similar considerations also
apply to computers incorporating array parallelism, either
without or in addition to vector parallelism.

4. SIMULATION OF CLASSICAL OR PSEUDO-CLASSICAL MANY-BODY
   SYSTEMS

Three principal kinds of simulation of such systems may be
identified: particle-particle simulation, e.g. molecular
dynamics simulation to study the structure of liquids, or
simulation of stellar clusters, or of clustering of galaxies;
particle-mesh simulation, e.g. galaxy simulation, or simulation
of dilute plasmas, or of microscopic semiconductor devices; and
particle-mesh/particle-particle simulation, e.g. simulation of
phase changes in ionic solids and liquids (such as KCl, CaO), or
of clustering of galaxies, or of a two-dimensional electron
film.

For the first kind, the action-at-a-distance formulation of the
force law is used. For the second kind, the force is treated as
a field quantity approximating it on a mesh. The third kind is
a hybrid of the former two. Partly the physics of the
phenomenon under investigation and partly the consideration of
the computational cost determine the choice. Pioneering work,
especially on particle-mesh simulation, has been done by Hockney
and Eastwood [1981] and their collaborators.

To illustrate these kinds of simulation, let us now consider a
particle-mesh/particle-particle simulation of an ionic
solid/liquid system. The long-range interaction is of Coulomb
type. It may be found by accumulating the charges into a charge
density as a source and computing the electrostatic potential
from Poisson's equation. The long-range force on any particle
is then found by computing the gradient of the potential at the
particle's position. In addition, pair-wise short-range forces
from nearby particles have to be included to take into account
the repulsion of the electronic shells.

The system is completely described by storing the positions and
velocities of all the particles in the computer. To solve
Poisson's equation, a computational mesh is introduced. During
the charge assignment phase, first, a given number of
neighbouring mesh points of a particle are found. Then the
charge is assigned to them so that computational errors are
minimized for the mesh points used, or equivalently to obtain
the best quality/cost compromise. Different assignmemt schemes
arise by the choice of different numbers of neighbouring points.

The state of the physical system at some time t is described by

the set of particle position and velocity vectors xi(t), vi(t),
where i = 1,....,N. The timestep loop updates these values
using the forces of interaction and equations of motion to
obtain the state of the system at a slightly later time t + DT,
as follows:

```
C  Build up charge distribution rho(x)

   LOOP FOR all particles

       Find neighbouring mesh points of particle

       Assign charge to mesh

   ENDLOOP

C  Solve Poisson's equation on mesh to obtain potential phi(x)

   LOOP FOR all mesh points

       Solve divgrad*phi = - rho/epsilon0

   ENDLOOP

C  Compute forces and accelerate each particle of mass m for
C  time step DT by changing particle velocity according to
C  total force

   LOOP FOR all particles

       Find mesh force FM on particle

       Find short-range force FS from near-neighbour particles

       vi(new) := vi(old) + (FM+FS)*DT/m

   ENDLOOP

C  Move each particle to new position

   LOOP FOR all particles

       xi(new) := xi(old) + vi(new)*DT

   ENDLOOP
```

The position and velocities of all particles are now known at
the new time-level. Repeated application of the time-step loop
allows to follow the evolution of the system in time. For a
system without short-range forces FS is omitted. The point of
this algorithm is that it has an arithmetic operation count
proportional to N, and not to N*N as would be the case if all
N*N pair interactions were evaluated directly. Furthermore,
Fast Fourier Transform techniques are used to solve Poisson's
equation.

A galaxy simulation is done by changing the sign in Poisson's
equation, and replacing the specific charge of a particle by the

square root of the gravitational constant G. Submicron
semiconductor simulation differs from this scheme in that
important short time-scale effects due to electron scattering by
the crystal lattice must be taken into account. This is done
by replacing the mobility approximation, valid in devices whose
dimensions are large compared with the electron mean free path,
by a microscopic description in which statistical Monte Carlo
methods are used to simulate the scattering of the electrons by
the crystal lattice.

5. SIMULATION OF STATISTICAL UND QUANTUM SYSTEMS

In the simulation of statistical systems in physics, the
equation of state , transport coefficients (pressure, viscosity,
and thermal conductivity, for example), and other macroscopic
properties of matter are related to and derived from
intermolecular forces. The area has diverse and important
applications, ranging from metallurgy to polymer chemistry to
semiconductors. It is an active area of research from the
points of view of theory, numerical methods, and experiment.
One numerical method for statistical physics is basically of the
type of molecular-dynamics calculations already mentioned in the
previous section.

The equations of statistical physics involve a large or infinite
number of degrees of freedom. Hence, the mathematical theory of
use here is the analysis over infinite-dimensional spaces.

In the study of quantum fields, almost the same mathematical
structure arises, the relationship between the two being that a
quantum field is a continuum limit of a statistical physics
(crystal or lattice) model [Wilson 74]. This analogy is
illustrated by the equations for the equilibrium distribution $d\omega$
and the partition function Z:

$$d\omega = e^{-\beta V} \prod_i dx_i \, , \, i=1,\ldots,N \qquad\qquad d\omega = e^{-S} \prod_x d\phi_x$$

$$Z = \int d\omega \qquad\qquad\qquad\qquad Z = \int d\omega$$

$$V = \sum_{i<j} U(x_i - x_j) \qquad\qquad\qquad S = \int L(\phi_x, \partial\phi_x/\partial x) d^4x$$

where V is the intermolecular potential energy taken as a sum of
pair potentials U, and S is the classical action taken as a
space-time integral of the Lagrange density depending on the

quantum field Φ_x.

Important numerical methods are the Monte Carlo methods used in:

- the quadrature of very-large-dimensional spaces, such as the determination of $d\omega$ above, as well as in

- the direct simulation of stochastic systems, an example of which we have briefly discussed at the end of the previous section.

These methods have become an "experimental" tool of theoretical physics. In studying the qualitative and quantitative behaviour of the kind of systems described in the introduction, one can carry out numerical studies in which the high-dimensional (i.e., many-body) character of the problem is not distorted.

In the application to the calculation of the equilibrium distribution $d\omega$, the essence of the Monte Carlo method is as follows:

- Starting from an arbitrary point in a given ensemble, one modifies a single particle position x_i "at random" but usually so as to lower the potential energy V, occasionally so as to raise V.

- After enough such elementary steps, convergence to the distribution $d\omega$ is obtained.

The procedure is analogous for quantum fields. As described here, the method is extremely simple, and complications arise from the necessity to obtain convergence in a reasonable time for realistic problems.

In statistical physics, perhaps the most significant success has been in the microscopic theory of fluids, where Monte Carlo simulation has provided the "experimental" basis for accurate series expansions in $\exp(-\beta U)-1$ of the expression for $d\omega$, which are especially useful to give weak corrections, such as real-gas corrections to an ideal gas. Other conspicuous successes are the extraction of critical exponents by joining the ideas of the renormalization group to Monte Carlo simulation. Monte Carlo simulation of a kinetic Ising model gives a quantitative account of scattering experiments on a real alloy, and Monte Carlo methods applied to the many-body nonrelativistic Schrödinger equation give a quantitative account of the energy of real liquid helium as a function of the density.

In quantum field theory the most significant successes have perhaps been in the determination of the quark-antiquark potential and of the nature of phase transitions, in studies on the existence of a renormalizable continuum theory and on the correct interpretation of lattice results, and in the computation of the "glue ball" spectrum. Examples of this type of work, in collaboration with the Institute for Theoretical Physics of the University of Berne, have recently been completed within the framework of the PARCOM Project already mentioned in section 3 [Carrupt et al. 1987; Decker et al. 1987].

6. UNITY AMONG THE VARIOUS STEPS OF THE OVERALL SIMULATION PROCESS

In the computational approaches to most of today's pressing and challenging scientific and technological problems the mathematical and computational aspects cannot and should not be considered in isolation. There is a unity among the various steps of the overall simulation process:

- from the formulation of the physical problem

- to the construction of appropriate mathematical models

- to the design of suitable numerical methods

- to their computational implementation, and, last but not least

- to the validation and interpretation of the computed results.

The steps are more often than not deeply interconnected and the computational process may indeed be part of the model construction. We may illustrate this by viewing the same problem from three different standpoints [Henzi 87a].

In Table 1 (Tabelle 1) the traditional approach is taken of looking at a number of scientific and engineeering disciplines leading to computational simulation from the viewpoint of the mathematical formulation of typical problems to which they give rise:

HSy	Hydrodynamic Systems
CSy	Chemical Systems and Combustion
PlPh	Plasma Physics
TPh	Particles Physics
PhM	Condensed-Matter Physics
GPh	Geophysical Applications
Met	Meteorology
Ast	Astrophysical Applications
StM	Structural Mechanics
3D	Nondestructive Testing and Tomographic Reconstruction (three-dimensional)
BioM	Mathematical Models in the Biological Sciences
ElK	Electronic Components

In Table 2 (Tabelle 2) these disciplines are touched upon again, this time from the viewpoint of the computational and mathematical difficulties that arise in connection with them. E.g., these difficulties may involve nonlinearities, large numbers of degrees of freedom, different scales of length and time, or singularities of various types.

In Table 3 (Tabelle 3) the viewpoint becomes that of the numerical algorithms involved in the computations, such as various discretization methods, continuation approaches, splitting techniques and Monte Carlo methods.

Of necessity, many topics have been left out. Some of these topics as well as some of those above have been considered more

	HSy	CSy	PlPh	TPh	PhM	GPh	Met	Ast	StM	3D	BioM	EIK
Partielle Differentialgleichungen	□	□	□	□	□	□	□	□	□	□	□	□
Inverse Probleme, Integralgl.	□			□		□		□		□	□	
Minimalisierung unter Zwangsbed.		□		□					□			
Gewöhnliche Differentialgl.	□	□			□	□		□	□			□
Hamiltonsche Dynamik	□	□	□	□								
Gruppen und Quantenfeldtheorie				□								
Verteilungen, Zustandssummen			□	□	□							
Quadratur hochdimens. Räume			□	□	□							
Simulation stochastischer Syst.	□	□	□	□	□			□	□			
Informations-(Signal-)Verarb.						□	□				□	□
Kombinatorik und Graphen				□	□							□

Tabelle 1   Mathematische Formulierungen

	HSy	CSy	PlPh	TPh	PhM	GPh	Met	Ast	StM	3D	BioM	EIK
Nichtlinearitäten	□	□	□	□	□	□	□	□	□	□	□	□
Freiheitsgrade	□	□	□	□	□	□	□	□	□	□	□	□
Versch. Zeit- & Längenskalen	□	□	□	□	□	□	□	□	□	□	□	□
Singularitäten in Koeff., Daten	□	□	□	□	□	□		□	□		□	
Randbedingungen	□	□	□	□			□	□			□	□
Bifurkation, Chaos, Symm.-Brech.	□	□		□	□	□	□	□	□		□	
Schlechtgestellte inv. Probleme	□			□		□				□	□	
Eff. Medien, Homogenis., Random	□				□	□			□			
Validierung, Fehlerabschätzung	□	□		□	□	□	□	□	□	□	□	□

Tabelle 2   Numerische und mathematische Schwierigkeiten

	HSy	CSy	PlPh	TPh	PhM	GPh	Met	Ast	StM	3D	BioM	EIK
Diskretisierungsmethoden:												
Finite Differenzen	□	□	□	□	□	□	□	□	□	□	□	□
Varitationsmethoden	□			□					□			
Finite Elemente	□	□	□	□	□	□	□	□	□	□	□	□
Transformationsmethoden	□		□	□								
Semidiskrete Methoden		□										
Charakteristiken	□	□										
Adaptive Grid Methoden	□	□	□	□	□	□	□	□	□	□	□	□
Meth. zur Lsg. diskretisierter Diff.-gl.:												
Direkte Methoden		□							□			
Indirekte, Multigrid Methoden	□	□	□	□	□	□	□	□	□	□	□	□
Fortsetzungs- und Homotop. M.	□			□	□			□	□			
Optimierungsmethoden	□			□				□	□			
Graphentheoretische Methoden	□			□	□							□
Splitting M. & Defect Corr.	□	□		□	□				□			
Monte Carlo Techniken	□	□	□	□	□		□	□				
Problemabhängige Methoden:												
Riemann Probl. & Wellenmoden	□											
Asymptotic Analysis		□										
Front Tracking	□	□										
Wirbelmethode	□	□							□			
Skaleninv. & Renormierungsgr.	□			□	□				□			
Konturdynamische Methoden	□		□									

Tabelle 3   Numerische Algorithmen

closely in an analysis of the importance of supercomputers for the telecommunications industry [Henzi 87b]. Neither was it attempted to address all computational and mathematical difficulties nor all variations of numerical algorithms.

7. TURNING RAW COMPUTER POWER INTO ADVANCES IN HUMAN KNOWLEDGE

From a computer modelling point of view, the unified framework provided e.g. by statistical mechanics, that was illustrated in Section 5, appears as the similarity of mathematical models and thence as the similarity of numerical algorithms and of computer programs for a wide range of physical phenomena. One benefit of the unified approach to a wide range of physical phenomena is that advances in computational methods in one area of application more rapidly find their way into others.

The principal benefit of considering computational science in general and e.g. simulation of many-body or of statistical and quantum systems in particular as entities appears when we look at the computer program. As we have seen, programs developed to study particular phenomena may, with minor modifications, be used to study something completely different. Large economies in human power and computer resources can be made by recognizing the similarities of physical, mathematical, numerical and computational problems, and by embodying the similarities in standardized structured programs. ASTRID, presented by Ralf Gruber [Gruber 88] at this symposium, is an example of such an approach.

Last year, the SPEEDUP organization was established in Switzerland with the objective of promoting vector and parallel computing in science and industry, as well as the interest that academic and industrial research workers always have at their disposal a system based on the current top-of-the-range number-crunching computer. The SPEEDUP activities have started out with the publication of the journal SPEEDUP, of which the first issue has appeared on November 1, 1987, and with the organization of workshops, of which the first took place on November 20, 1987, on the theme "Supercomputing - The State of the Art". Coming workshops in 1988 will take place as follows:

January 15	Parallel Computers
March 18	The ASTRID Project of EPF Lausanne: A Family of Simulation Programs Adapted to Parallel Vector Computers
May 20	Simulation of Quantum and Statistical Systems
September 16	Computational Quantum Chemistry (tentative)
November 18	The New National Supercomputing System for Switzerland (tentative)

Subscriptions to the SPEEDUP journal and requests for information should be mailed to: SPEEDUP Secretariat, IAM, Uni Bern, Länggassstrasse 51, CH-3012 Bern, Switzerland; Telephone 031/65 86 81; E-Mail u02e@cbebda3t.bitnet.

Traditional subject divisions lead academia to teach
experimentation and theory to the physicist, numerical analysis
to mathematicians, and computer architecture to computer
scientists, so that it is rare to find an individual who has the
combination of backgrounds required for the type of work in
large-scale simulation discussed here. SPEEDUP and, more
generally, Computational Science form interfaces between
advanced computers, such as parallel computers, and the
physicist or engineer with the problem; these interfaces intend
to help the research worker to find the combination of the best
computer architecture, the best numerical algorithms and the
computer simulation that can bring new insight and discoveries
in science and make a contribution to the better design of
industrial products.

REFERENCES

Carrupt, P.A., Gmür, T., Gruber, R., Haldi, P.A., Henzi, R., Posternak, M., Rougemont, F., and Therre, J.P. (1987): "Experiences faites sur le Cray 1S/2300", report for "ZETHA Projektoberleitung" (ETH Zürich), GASOV EPF Lausanne, August 1987.

Decker, K. (1987):   "Supercomputer Benchmark with a Monte Carlo Updating Routine: Benchmark Report on Program GCEQ08P", University of Berne, PARCOM Project Preprint BU-IAM-PPP-01-1987.

Decker, K., Hasenfratz, A., and Hasenfratz, P. (1987):   "Singular Renormalization Group Transformations and First Order Phase Transitions - Monte Carlo Renormalization Group Results", University of Berne, Preprint BUTP-87/11.

Decker, K., and Henzi, R. (1987):   "Supercomputer Benchmark Using a Monte Carlo Updating Routine", written version of talk at Cray User Group Meeting, Garmisch-Partenkirchen Fall 1986, University of Berne, January 1987.

Gruber, R. (1988):   Talk in these Proceedings.

Henzi, R. (1987a):   "Supercomputing - Neue Stossrichtungen für Forschung und Entwicklung in der Schweiz", Output, Nummer 7/Juli 1987, Schweizerische Vereinigung für Datenverarbeitung (SVD), Zürich, pp.   33-38; and Informatik Bulletin, Nummer 49/April 1987, ETH Zürich, pp.   2-7.

Henzi, R. (1987b): "Supercomputer - Analyse ihrer Bedeutung für das Nachrichtenwesen", University of Berne, PARCOM Project Preprint BU-IAM-PPP-02-1987.

Henzi, R. (1988a):   "Testing Parallel Computers with PARPACK - A Package of Application-Oriented Problems with Parallelism ", to be published.

Henzi, R. (1988b):   "Vectorization and Optimization of Inverse Problems - Case Study in Particle Scattering at High Energies", to be published.

Hockney, R.W., and Eastwood, J. W. (1981):   "Computer Simulation Using Particles", McGraw-Hill Inc., New York.

Hochstrasser, U. (1988):   "The National Initiative for Super-computing in Switzerland", SPEEDUP, Volume 2, Number 1, January 1, 1988, Bern, pp.   8-11.

Wilson, K. (1974):   "Confinement of Quarks", Phys. Rev. D10, p. 2445.