

Institute *of* Physics

# Newsletter

*of*

**The Computational Physics Group**



Fall 2003

## MEMBERS OF THE COMMITTEE

Peter Borcherds (chairman)	p.h.borcherds@birmingham.ac.uk
Allan Boardman (treasurer, secretary)	a.d.boardman@salford.ac.uk
Richard Ansorge	rea1@phy.cam.ac.uk
Roger Barrett	R.Barrett@surrey.ac.uk
Alan DuSautoy	alan.dusautoy@npl.co.uk
Andrew Horsfield (newsletter)	a.horsfield@ucl.ac.uk
Dave Hutchings	d.hutchings@elec.gla.ac.uk
Geraint Lewis	dg.lewis@physics.org
Jim McNiff	jemc@tinyonline.co.uk
Ian Morrison (web page)	i.morrison@salford.ac.uk
Massimo Noro	Massimo.Noro@Unilever.com
Matt Probert	mijp1@york.ac.uk

Our web page can be found here: <http://www.iop.org/IOP/Groups/CP/>

Comments about the newsletter should be sent to Andrew Horsfield

Picture of Tux by Larry Ewing. See <http://www.linux.org/info/logos.html>

# Contents

<b>Some Linux information available on the web</b>	<b>1</b>
<b>What on Earth are...BEOWULF CLUSTERS?</b>	<b>5</b>
<b>High Performance Computing using a Beowulf Computer</b>	<b>8</b>
<b>Reports on Meetings</b>	<b>15</b>
A Gentle Introduction to Quantum Information . . . . .	15
Free CD-ROMs of Simon Singh's book "The Code Book" . . . . .	17
Computational Physics Symposium at CMMP03 - Belfast . . . . .	17
<b>Upcoming Computational Physics Group Events</b>	<b>20</b>
The 2 <sup>nd</sup> Annual Computational Physics Thesis Prize . . . . .	20
<b>Other Upcoming Meetings</b>	<b>20</b>
International Conference on Computational Nanoscience and Nanotechnology . . . . .	20
10th UK Monte Carlo User Group meeting (MCNEG 2004) . . . . .	21
NPL Workshop on Monte Carlo codes . . . . .	21
3 <sup>rd</sup> International Conference on Computational Modeling and Simulation of Materials . . . . .	21

## **Some Linux information available on the web**

*Andrew Horsfield (Editor)*

There is an enormous amount of information about Linux. Below are some web sites that might be of general interest to computational physicists.

### **General Websites**

*RedHat Distribution*

<http://www.redhat.com/>

*Suse Distribution*

<http://www.suse.com/>

*Mandrake Distribution*

<http://www.mandrakelinux.com/en/>

*Debian Distribution*

<http://www.debian.org/>

*Linux News*

<http://linuxtoday.com/>

*Linux Format Magazine*

<http://www.linuxformat.co.uk/>

### **Application Repositories**

*SAL*

SAL (Scientific Applications on Linux) is a collection of information and links to software that will be of interest to scientists and engineers. There are currently 3,070 entries in SAL.

<http://sal.kachinatech.com/>

### *Sourceforge*

SourceForge.net is the world's largest Open Source software development website, with the largest repository of Open Source code and applications available on the Internet.

<http://sourceforge.net/>

### *Freshmeat*

Freshmeat maintains the Web's largest index of Unix and cross-platform software, themes and related "eye-candy", and Palm OS software.

<http://freshmeat.net/>

## **Clustering Software**

### *Beowulf*

<http://www.beowulf.org/>

### *SCore*

<http://www.pccluster.org/score/dist/score/html/en/index.html>

## **Graphics**

### *Xfig*

Xfig is an interactive drawing tool which runs under X Window System Version 11 Release 4 (X11R4) or later, on most UNIX-compatible platforms. It is freeware, and available via anonymous ftp. In xfig, figures may be drawn using objects such as circles, boxes, lines, spline curves, text, etc. It is also possible to import images in formats such as GIF, JPEG, EPSF (PostScript), etc. Those objects can be created, deleted, moved or modified. Attributes such as colors or line styles can be selected in various ways. For text, 35 fonts are available. Text can also include Latin-1 characters.

<http://www.xfig.org/>

### *Xmgrace*

Grace is a WYSIWYG 2D plotting tool for the X Window System. Grace runs on practically any version of Unix-like OS. As well, it has been successfully ported to VMS, OS/2, and Win9\*/NT/2000/XP (some minor functionality may be missing, though).

<http://plasma-gate.weizmann.ac.il/Grace/>

### *The GIMP*

The GIMP is the GNU Image Manipulation Program. It is a freely distributed piece of software suitable for such tasks as photo retouching, image composition and image authoring.

[http://www.gimp.org/the\\_gimp.html](http://www.gimp.org/the_gimp.html)

## **Writing Papers**

### *TeX*

teTeX is a complete TeX distribution for UNIX compatible systems, maintained by me, Thomas Esser.

<http://www.tug.org/teTeX/>

### *LyX*

LyX is an advanced open source document processor that encourages an approach to writing based on the structure of your documents, not their appearance. LyX lets you concentrate on writing, leaving details of visual layout to the software. It exports LaTeX and can produce RevTeX papers.

<http://www.lyx.org/>

### *TeXmacs*

GNU TeXmacs is a free scientific text editor, which was both inspired by TeX and GNU Emacs. The editor allows you to write structured documents via a wysiwyg (what-you-see-is-what-you-get) and user friendly interface. New styles may be created by the user. The program implements high-quality typesetting algorithms and TeX fonts, which help you to produce professionally looking documents.

<http://www.texmacs.org/>

### *OpenOffice*

This is a complete office suite.

<http://www.openoffice.org/>

## **Software Development**

### *Emacs*

Emacs is the extensible, customizable, self-documenting real-time display editor.

<http://www.gnu.org/software/emacs/emacs.html>

### *Vim*

Vim is a highly configurable text editor built to enable efficient text editing. It is an improved version of the vi editor distributed with most UNIX systems.

<http://www.vim.org/>

### *NEdit*

NEdit is a multi-purpose text editor for the X Window System, which combines a standard, easy to use, graphical user interface with the thorough functionality and stability required by users who edit text eight hours a day. It provides intensive support for development in a wide variety of languages, text processors, and other tools, but at the same time can be used productively by just about anyone who needs to edit text.

<http://www.nedit.org/>

### *GCC*

GCC is the GNU Compiler Collection, which currently contains front ends for C, C++, Objective-C, Fortran, Java, and Ada.

<http://gcc.gnu.org/>

*Intel Compiler*

<http://www.intel.com/software/products/compilers/>

*Portland Compiler*

<http://www.pgroup.com/>

## **What on Earth are...BEOWULF CLUSTERS?**

*Biagio Lucini*

*The following article is an extract from a longer article first published in the Linux magazine LinuxFormat. The full article can be found at:*  
<http://www.linuxformat.co.uk/archives/LXF21.woe.pdf>

*"Famed was this Beowulf: far flew the boast of him, son of Scyld, in the Scandian lands. So becomes it a youth to quit..."*

No, no, no. That's the wrong Beowulf. This one is to do with clustered computing.

*Ah, sorry. I've heard people going on about clusters, I've just never associated PCs with epic poetry. So what are they then?*

No need to apologise. In simple terms, a cluster of PCs is an ensemble of single computers that behave almost as if they were just the one entity, with the individuality of all the single machines reduced as much as possible.

*So why are these clusters all over the news like at the moment?*

PC-related achievements are growing at an impressive rate. Thanks to the technological progress, manufacturers are able to build more and more efficient computers. However, it is almost impossible to catch up with the ever-more demanding requirements of the market. Clustering is based on the simple concept that a number of PCs linked together can do a better job than a single machine. Another practical reason is to simplify the administration of a group

of workstations, by centralising as many operations as possible in such a way that, for instance, you don't have to upgrade a software package on every single workstation.

*What are clusters good for?*

There are several useful applications. In general, every time you need a fast response or to balance loads and tasks among several machines a cluster can be a good idea.

*It all sounds rather complicated. How are clusters set up?*

It depends on the problem you are addressing. Some clusters are just a cheap way for a system administrator to deal with many workstations. In this case, a good choice is to have a central server which contains the back-up units and all the data on big disks and client workstations with the operating system running locally. Other times a massive computational unit is needed. In this case, it is better to interconnect several PCs with fast links, in order to allow them to share the CPU cost of the required operations among each other. But there are several other possible choices, related to different problems.

*How can a bunch of computers pretend to be just one PC?*

This is mainly a software issue. If you have a number of PCs interconnected in some way, in an ideal world you shouldn't be able to distinguish them from a computer whose computational power is the sum of the computational abilities of each machine. For instance, let's call a single PC of the cluster a node. If a request can be processed by a constituent node, say, in 10 seconds, in an ideal cluster of 10 nodes we should expect that the same request is processed in one second. Another example: suppose that a node can accept 10 requests per second; then we expect that the ideal cluster above can process 100 requests per second. Ideally, users shouldn't realise that they're dealing with a cluster, it should appear as just a fast computer. Of course, the clustering software must take care of accepting requests, splitting them to the various nodes, collecting the results and giving the answer.

*What is the clustering software? Is it a particular operating system for clusters?*

No. It's best to think of it as an ensemble of applications. Basically, things work in the following way: each node is an independent PC, which has its own operating system running locally. Each node also runs the clustering software: this is aware of how many nodes the cluster is composed of, and how any single node can be reached (by recording the IP addresses of all nodes). A good clustering application must be flexible enough to allow the easy addition, removal and upgrade of a single node.

*Is Linux the only operating system used on clusters?*

No. Most of the commercial descendants of Unix have clustering extensions and even Windows NT/2000 can be used for binding PCs in a cluster.

*Why has Linux managed to corner the market in clustering software?*

The first reason is stability. Clusters are particularly common in the scientific area, where people use clusters because they run applications that require a lot of resources and a lot of time to give an answer. These mission critical applications are required to be 'up' for long periods of time and, as we know, some operating systems cannot guarantee stability on such a large time scale. Another big advantage is the cost: Linux, and some cluster software, can be obtained very cheaply (even free) so to install it on 500 machines costs the same as installing it on one (not including hardware costs of course). Also, it runs on almost all hardware, including the cheapest architectures. This enables organisations to build a cluster system from cheaper, off-the-shelf components rather than having to invest heavily in a proprietary system. This is why Linux's reputation has improved so much, and why many commercial vendors base their market strategy on it.

## High Performance Computing using a Beowulf Computer

M. I. J. Probert

*This article is an abridged version of a longer article "Molecular Dynamics Studies of Liquids using a Beowulf Computer" which appears in Contemporary Physics, vol. 44, no.5, p435-450.*

The use of computer simulation as a fundamental tool in the physical sciences has a long and honourable history, stretching back to the earliest days of computational physics in 1953 when the first molecular dynamics (MD) calculations were performed. Since then, there has been an explosive growth in available computing power and algorithms.

One popular route to high performance computing is the massively parallel approach where a computer is built around a large number (usually 64 - 1024) of processors with a special interconnect between the processors. The result is an architecture that can be scaled up to handle the largest problems and can offer the highest peak performance for a fraction of the cost of the equivalent vector computer. However, this architecture has many subtle difficulties, such as managing the communications between the processors, and the distribution of memory amongst the processors ("shared memory" or "distributed memory"). Exploiting the theoretical performance of such a machine is also non-trivial - it typically requires rewriting serial codes (having to "think parallel") and using new algorithms and programming paradigms. A detailed exposition of the hardware aspects may be found in the text by Hennessy and Patterson, whilst the software implications are thoroughly covered in the text by Dowd and Severance.

With the rise in power of commodity processors and desktop PCs, it was inevitable that attempts would be made to turn a network of PCs into a massively parallel computer. The earliest attempts (*circa* 1994) used standard networking technology, such as 10 MBit/s Ethernet, to connect the separate PCs together. Whilst this cluster approach worked, it suffered from very poor communication latency and bandwidth compared to the dedicated approach. However, since that time there have been many developments in the hardware available, as dedicated high-speed network interfaces have become available,

and in the software required to turn the cluster into a single computer - known as a Beowulf. An excellent resource for information about Beowulf computing can be found online at the Beowulf cluster site homepage.

There are numerous design decisions that need to be made when buying or building a Beowulf computer. There are now numerous vendors which sell custom-built solutions, including rack-mounted cabinets and customised software, for a price. Or there is the D.I.Y. approach, using COTS (Commodity Off-The-Shelf) components and the Open Source software that is readily available on the Internet. Whichever approach is taken, the two most important decisions that will impact on the overall performance of the machine are the choice of processor and the choice of interconnect. Almost invariably, the operating system of choice for a Beowulf is some flavour of Linux, due to the wide range of Open Source software that has been created for the task of running a Beowulf computer - such as Open PBS for scheduling jobs in a multi-user environment - although there have been a handful of instances when Microsoft WindowsNT has been used instead. What is beyond doubt is that the Beowulf approach represents the lowest cost route into parallel computing at the moment.

Whilst most people will have some familiarity with the choice of processors available for a Linux-based Beowulf, it is often found that in many parallel codes that it is the performance of the interconnect which is more important than the raw power of the processor. Hence the choice of interconnect is crucial and the available technologies will be less familiar to most people. Whilst Ethernet is always an option (and the cheapest one at that) it suffers from higher latency and smaller bandwidth than many of the more specialised offerings, from companies such as Quadrics, Myrinet or Scali(Dolphin). The performance of an interconnect can be crudely characterised in terms of bandwidth and latency. Latency is the time delay due to setting up the communication channel, handshaking, etc. and is a constant overhead associated with every communication. Bandwidth is a measure of the speed of the interconnect, but can be heavily dependent on other factors, such as the network protocol in use, the load on the send/receive processors, whether the connection is direct or goes via a switch, the motherboard chipset, etc. A rough comparison of the four most common options is given in table 1.

A good, impartial and up-to-date source of information on the real-world performance of the different hardware combinations is provided by the UK High

Table 1: Approximate performance comparison of the four most common network technologies available for a Beowulf computer.

Product	Latency ( $\mu s$ )	Bandwidth ( $MBytes/s$ )	Cost
Gbit Ethernet	>60	>40	low
Myrinet 2000	<7	>480	moderate
SCI	<5	>320	moderate
QsNet	<5	>680	high

End Computing initiative (UKHEC) and can be found online at the UKHEC homepage. UKHEC also publish benchmarks of real-world codes running on a wide range of machines, so it is possible to see what the likely performance of any given hardware and software configuration is likely to be - which is a better indicator than the many industry standard benchmarks available.

In October 2000, the Department of Physics at the University of York decided that a Beowulf computer would be the most cost effective way of providing the computational resources needed across a range of different research groups. At the same time, EPSRC had a Multi-Project Research Equipment initiative, which seemed to be an appropriate source of funding. Therefore a feasibility project was conducted to evaluate the (then) current state of hardware. The decision was then taken that maintenance and upgrading the computer would be much simpler if it was single-sourced, and so a variety of Beowulf vendors were approached. The resulting quotations were used to prepare a grant proposal, and the machine (by now an upgraded specification) was duly installed in December 2001.

The chosen configuration was a rack-mounted Intel P4 Xeon processor (1.7 GHz) system with 1 GB memory per node with Myrinet 2000 interconnect. The motherboards were dual-processor but with only one processor present, in order to gain the maximum performance from the interconnect (as the dual-processor motherboard PCI bus was 64-bit, 66 MHz compared to a standard PC motherboard of 32-bit, 33 MHz). Each node is connected to a Myrinet switch to reduce the latency of a multiple-hop communication. The choice of Myrinet was based upon the predicted performance of the typical codes that were going to be hosted on the Beowulf - with lots of short messages, very low latency is essential to get reasonable performance. The cost of the

Myrinet cards and switches was almost 50% of the overall hardware cost. The choice of Intel P4 Xeon processors was based upon the numeric performance in certain matrix algebra tests (representative of the typical codes used) and the availability of dual processor motherboards. This was the second most significant cost - closely followed by the 800 MHz RDRAM modules.

There are 32 compute nodes and one master node which governs job submission and queueing - essential in a multi-user machine. Each node was later upgraded by the addition of another 33 identical processors to make each node a dual-processor node. The machine is therefore a hybrid of shared memory (between the two processors per node) and distributed memory (between the nodes). All nodes, including the master node, are identical except that the master node has additional disks for central file storage. There is an additional Ethernet network between all the nodes for disk traffic.

The machine has proven to be very reliable, with only a small number of hardware problems which have been fixed under the maintenance contract, and has provided excellent uptime. It is currently used by around 30 researchers from 6 different research groups for a wide variety of tasks. It has dramatically increased the throughput of many research projects, and has also enabled the (relatively) straightforward development of new parallel codes which would not have been possible on national facilities. For users of serial codes, the addition of the extra processor per node has produced a typical  $\times 1.5$  speedup without any code rewrite, using auto-parallelising compilers and/or threaded system libraries. For users of parallel codes, the use of a high performance interconnect with fast processors means that a 16-node calculation on the Beowulf typically out performs a 16 or 32-node calculation on a Cray T3E. However, to make the most out of the distributed memory nodes requires the implementation of parallel algorithms with explicit parallelisation using MPI library calls.

In order to fully exploit the power of the Beowulf computer (or any parallel computer), it is necessary to have a suitable parallelisation strategy. Each different strategy has its own strengths and weaknesses and the choice of which one is most appropriate depends upon the number of particles in the system, the characteristics of the parallel hardware and the nature of the questions being asked of the calculation.

With any such strategy, there will be considerations about load balancing and scaling. Load balancing is the equal distribution of work between the different processors in a parallel computer - if there is load imbalance, then

the processor with the greatest load will become a bottleneck and the rest of the calculation will stall until the slowest processor has caught up. An implementation with good load balancing will make maximum concurrent use of each processor. The other primary consideration is that of scaling - that is, how does the overall execution time and resources (e.g. memory) of the calculation depend on the size of the problem (often called "type 1 scaling") and the number of processors ("type 2 scaling").

Any parallel code will have some portion that cannot be parallelised (the "serial" part), and some sections that execute in parallel on the different processors. As the number of processors is increased, the overall time to execute the parallel sections will decrease and the fraction of time spent in the serial sections will become more significant. This is known as Amdahl's Law:

$$t_n = t_s + \frac{t_p}{n} \quad (1)$$

where  $t_n$  is the total execution time on  $n$  processors of a parallel computer,  $t_s$  is the amount of time spent in the serial sections and  $t_p$  is the amount of time spent in the parallel parts if only a single processor is active.

However, the scaling predicted by Amdahl's Law is rarely seen in practice - a more typical behaviour is shown in figure 1. In this case, a highly parallelised code ( $t_p/t_s = 49$ ) is predicted by Amdahl's Law to have a useful speedup with 50 processors, whereas a real-world implementation would probably have a maximum speedup with 25 processors and a degradation in performance if any more processors are used.

The reason for this non-ideal behaviour is the performance of the interconnect between the processors. The ideal time taken to transmit a message between nodes is

$$time = latency + \frac{size}{bandwidth} \quad (2)$$

and so if the messages get too small,  $size < latency * bandwidth$  and so the latency will dominate. Increasing the number of processors will, in general increase the number of messages and reduce the typical size of each message - hence the communications cost increases with the number of processors. If too many processors are used, then the cost of the extra communications outweighs the speedup gained by the increased number of processors. Hence, for any given code and parallel computer, there will be an optimum number of

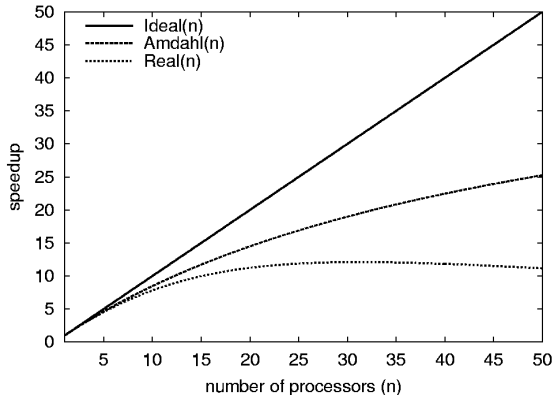


Figure 1: Effect of increasing the number of processors on a reasonably paralised code ( $t_p/t_s = 49$ ). Ideal scaling would be linear, Amdahl's Law predicts a leveling off with increasing number of processors, whereas the more realistic real-world scenario has a maximum speedup beyond which the speedup actually decreases with increasing number of processors.

processors to use on any particular size problem. This increases the challenge of writing highly scalable codes and can be expressed as a modified form of Amdahl's Law:

$$t_n = t_s + \frac{t_p}{n} + \delta_s + \delta_p n \quad (3)$$

where  $\delta_s$  is the cost of sending messages between processors due to the finite bandwidth (and hence is approximately independent of  $n$  as the typical size of each message  $\sim \frac{1}{n}$ ), and  $\delta_p$  is the cost due to the latency of each message. This dramatically illustrates the effect of the quality of the interconnect on the performance of any given code as already discussed.

Traditional supercomputers, use proprietary (and much more costly) interconnect - and hence they typically outperform a Beowulf system with a large enough number of processors - but at a much higher financial cost. At what point the proprietary solution starts to outperform the Beowulf solution depends on the size of the problem and the parallelisation strategy - it can be as

little as 16 nodes or as much as 512 nodes. However, one point is clear - for small to medium-size problems, the Beowulf solution is the most cost-effective parallel computing solution currently available.

## Reports on Meetings

Reports from the organisers of the meetings of the Computational Physics Group are presented below.

### A Gentle Introduction to Quantum Information

*Organised by: Andrew Horsfield*

On Friday, September 12, 2003, a meeting was held at the Institute of Physics, 76 Portland Place, London, that was an introduction to the theory of quantum information. The meeting was aimed at those who are interested in the field but are not practitioners. It was very well attended with 79 people registered, about half being from industry. There was notable even interest from investors.

The abstracts of the talks are as follows:

- **A Brief History of Cryptography** *Simon Singh*

Simon gave a brief overview of the development of encryption and introduce some basic concepts. He started with ancient substitution ciphers and went on to explain the significance of the Enigma cipher, which involved a demonstration of a genuine Enigma machine. He then discussed modern internet encryption, the key distribution problem and public key cryptography. He concluded with a brief introduction to quantum cryptography.

- **Quantum Theory** *Ian Percival*

The talk included some aspects of quantum theory that are important for quantum information, including qubits, cloning, mixed states, Schmidt decomposition, entanglement, quantum parallelism and the role of quantum measurement.

- **Quantum Computation** *Samuel Braunstein*

Imagine a computer whose memory is exponentially larger than its apparent physical size; a computer that can manipulate an exponential set of inputs simultaneously; a computer that computes in the twilight zone

of Hilbert space. You would be thinking of a quantum computer. Relatively few and simple concepts from quantum mechanics are needed to understand the possibilities allowed by quantum computers. We discussed these possibilities and explained how a simple quantum algorithm works.

- **Optical Quantum Cryptography** *Gerald Buller*

In recent years quantum information research has led to the discovery of a number of remarkable new paradigms for information processing and communication. These developments include quantum cryptography schemes that offer unconditionally secure information transport guaranteed by quantum-mechanical laws. Such potentially disruptive security technologies could be of high strategic and economic value in the future. This presentation specifically addressed the state-of-the-art and performance limitations of optical fibre-based quantum key distribution systems. System performance was discussed in the context of component development: single-photon detectors; weak coherent sources; phase and polarisation modulators, data acquisition and processing hardware; and temporal synchronisation issues. The implementation of recent and potential future developments was discussed in the context of the realisation of practical fibre-based quantum key distribution links.

- **Quantum Computing in the Solid State: The Challenge of Decoherence** *Andrew Fisher*

The potential advantages of embodying a quantum information processing device in a solid-state system include: scalable architectures, ease of manufacturing, and of integration with conventional computing hardware. On the other hand, the intimate contact of quantum bits ("qubits") with a condensed-phase environment means that quantum mechanical decoherence must be carefully controlled if the demanding thresholds for quantum error correction are to be met. In particular, one must achieve this control while effectively coupling the qubits in the presence of external fluctuations. In this talk I reviewed a few of the many proposed solid-state routes to quantum information processing. I then introduced the concept of decoherence and very briefly outlined how to quantify the trade-off between the desirable inter-qubit interac-

tions and the undesirable decoherence they introduce. Finally, I discussed the prospects for a novel class of quantum computing proposals which invoke optical excitation to couple the qubits and perform quantum logic operations.

## Free CD-ROMs of Simon Singh's book "The Code Book"

There are 33 CD-ROMs of "The Code Book" by Simon Singh that are left over from the *A Gentle Introduction to Quantum Information* meeting and which we are giving away free to members of the Computational Physics Group of the Institute of Physics. If you are interested in obtaining one, please send an e-mail to Andrew Horsfield to register your interest. If there are still CD-ROMs available then you will receive an e-mail inviting you to send a stamped self-addressed padded envelope in which the CD-ROM will be sent back to you.

## Computational Physics Symposium at CMMP03 - Belfast

*Organised by: Ian Morrison*

The Computational Physics group organised a half day symposium at the annual Condensed Matter and Materials Physics conference held this year at Queens University Belfast on the afternoon of 9th April. The session was very well attended and comprised one invited and two contributed talks. The talks gave examples of how computer simulation facilitated interpretation of a variety of phenomena in condensed matter physics including catalysis, anomalous properties of water and conductance in low dimensional systems.

Abstracts of the talks were as follows:

- **Modelling oxide surfaces using embedded cluster methods** *R Catlow, S A French, A A Sokol*

Understanding of oxide surfaces poses major challenges relating in particular to their defect structures and to the nature of reactions taking place on their surfaces. Although substantial progress has been made in modelling their surfaces using periodic methods, based on both interatomic potential methods and plane wave DFT methods, many of the

problems, particularly those relating to reactivity, can be effectively investigated using localised methods based on embedded cluster methods. The lecture will therefore review recent developments on embedded cluster methodologies and their applications to oxide surfaces, focussing on the case of zinc oxide, for which we will describe recent work on

1. the mechanism of the synthesis of methanol from  $\text{CO}_2$  and  $\text{H}_2$  and
2. the structures and energetics of atoms and small clusters of copper supported on their surfaces.

● **Quantum effects in liquid water** *M J Glover, M I J Probert*

The structural behaviour of liquid water close to the anomalous density maximum is believed by some to be governed by second nearest neighbours. We present a path integral molecular dynamics simulation of liquid water at a range of different temperatures and densities to investigate the mechanism by which the density maximum occurs. Path integrals are used to treat the quantum behaviour of nuclei, which for hydrogen in water is significant even at room temperature. We analyse the structural behaviour using a variety of techniques, including the second nearest neighbour behaviour, in an attempt to find out why fish do not die in winter. The water was modelled using the recently developed TIP5P potential, which has been shown in previous studies to accurately reproduce the structure of liquid water.

● **Inelastic Current-Voltage Spectroscopy of Atomic Wires** *M J Montgomery, J Hoekstra, T N Todorov, A P Sutton*

The mechanical and electronic properties of metallic atomic chains have been the subject of intense study, both experimental and theoretical. One technique, long used to probe the phonon structure of point contacts and recently applied to atomic chains, is point contact spectroscopy, where the current-voltage characteristics reflect the inelastic electron-phonon coupling in atomic wires under an applied voltage. By determining the phonon modes supported by a given structure and calculating the associated coupling to the electrons, the inelastic current-voltage spectroscopy of the system is modelled. Results are presented for the application of this formalism to a gold atomic chain between two electrodes.

Particular longitudinal phonons are found to have greatly enhanced coupling to the electronic states of the system. This leads to a large drop in the calculated differential conductance at threshold energies associated with these phonons. It is found that with increasing tension these energies decrease, while the size of the conductance drop increases, in agreement with experiment.

- Please also note that the final advertised talk had to be cancelled at short notice as Prof Dzhurakhalov was unable to attend the conference

## Upcoming Computational Physics Group Events

### The 2<sup>nd</sup> Annual Computational Physics Thesis Prize

Note: the deadline has been extended

The Committee of the Institute of Physics Computational Group has endowed two annual prizes. £500 will be awarded to the author of the PhD thesis that contributes most strongly to the advancement of computational physics. Two runners-up will receive £250. The Committee will select the recipients and its remit will be very broad, in order to capture a broad spectrum of modelling activity.

- The deadline for applications is December 31<sup>st</sup>, 2003. The competition is open to all students whose PhD examination has taken place in 2003.
- The submission format is a 4 page (A4) abstract together with a citation (max. 500 words) from the PhD supervisor.

- The submission address is:

PROFESSOR A D BOARDMAN  
HON. SEC, IOP COMPUTATIONAL PHYSICS GROUP  
JOULE PHYSICS LABORATORY  
SCHOOL OF SCIENCES  
UNIVERSITY OF SALFORD,  
SALFORD, M5 4WT

*Applicants must have carried out their thesis work at a University in the United Kingdom or the Republic of Ireland.*

## Other Upcoming Meetings

### International Conference on Computational Nanoscience and Nanotechnology

This meeting will be held March 7-11, 2004, at the Boston Sheraton Hotel & Hynes Convention Center, Boston, Massachusetts, U.S.A.

Web page: <http://www.cr.org/ICCN2004/>

## **10th UK Monte Carlo User Group meeting (MCNEG 2004)**

This meeting will be held March 15-16, 2004, at the National Physical Laboratory, Teddington, UK.

Web page: <http://www.npl.co.uk/ionrad/training/montecarlo/>

## **NPL Workshop on Monte Carlo codes**

This meeting will be held March 17-18, 2004, at the National Physical Laboratory, Teddington, UK.

Web page: <http://www.npl.co.uk/ionrad/training/montecarlo/>

## **3<sup>rd</sup> International Conference on Computational Modeling and Simulation of Materials**

This meeting will be held in Acireale, Province of Catania, Sicily on May 30th June 4, 2004. This conference is part of the CIMTEC series.

Web page: <http://www.technagroup.it/modeling.htm>