Role of Parallel Architectures in Periodic Boundary Calculations [and Discussion]

M. C. Payne, L. J. Clarke, I. Stich and A. M. Stoneham

Phil. Trans. R. Soc. Lond. A 1992 **341**, 211-220
doi: 10.1098/rsta.1992.0095

Role of parallel architectures in periodic boundary calculations

By M. C. Payne[1], L. J. Clarke[2] and I. Stich[1, 3]

[1] *Cavendish Laboratory, Madingley Road, Cambridge CB3 0HE, U.K.*
[2] *Edinburgh Parallel Computer Centre, University of Edinburgh, Mayfield Road, Edinburgh EH9 3JZ, U.K.*
[3] *Institute of Inorganic Chemistry of the Slovak Academy of Science, Bratislava, Czechoslovakia*

Ab initio computations can be used to determine the values of a wide variety of physical properties of atoms, molecules and solids. The calculations are computationally demanding and can only be applied to small systems. One possible method for overcoming this limitation is to use parallel computers which, in principle, can provide unlimited computational power. In this paper the technical difficulties associated with parallelizing *ab initio* calculations are reviewed and a case study detailing the implementation of total energy pseudopotential codes on parallel machines is presented.

1. Introduction

There are a large number of *ab initio* or first principles methods available for calculating the physical properties of atoms, molecules and solids. These methods can be used to calculate the values of a wide range of physical properties and the results are consistently found to be in very good agreement with experimental values. The success of these methods has been so great that they are now generally regarded as predictive. However, all these methods are computationally demanding and even calculations for small systems, containing only a few atoms, can only be performed on powerful workstations or supercomputers. At the same time it is increasingly apparent that many complex processes in physics, chemistry, biology and materials science can only be solved with *ab initio* modelling of systems a hundred or thousand times larger than those now studied. There are two ways of increasing the size of system that is accessible to *ab initio* modelling. One is to improve the algorithms and hence increase the computational efficiency of the method. The other is to use a more powerful computer. It is now accepted that the only way to achieve significant increases in computational power in the immediate future is by harnessing together the power of many processors in a parallel computer. The most powerful parallel machines presently available achieve computational performances of the order of 10 GFlop but it is clear that within the next five years at least one teraflop machine will become available.

Although the number of processors in a parallel machine can, in principle, be increased without limit it is not clear what ultimately limits the technology that links the processors together into a useful machine and so the ultimate performance that can be achieved from such machines is not known. However, it is obvious that computational power of the order of ten thousand times greater than present supercomputers and one hundred thousand times greater than present workstations will be available. These figures are so large that it appears that the only obstacle to *ab initio* modelling of complex processes is the implementation of existing techniques on parallel machines. Unfortunately this is not the case because of the severe scaling of the computational time with system size of most *ab initio* techniques. If the computational time scales as the fourth power of the system size then the 10000 fold increase in computational power yields only a ten fold increase in the size of the system that can be studied; still one or two orders of magnitude removed from the target figure for complex processes. This is not to say that the *ab initio* method cannot be applied to a large range of important scientific problems. However, it is clear that implementation of *ab initio* codes on parallel computers alone cannot be viewed as the solution to the limitations on system size. Another problem that is particularly acute in many *ab initio* methods is that it may not be possible to implement the current numerical algorithms on parallel machines. One example is the QR factorization method for matrix diagonalization (Wilkinson 1965). The computational time for the QR method scales as N^3, where N is the size of the matrix, but this algorithm cannot be implemented on a parallel computer. Other matrix diagonalization techniques such as the Jacobi method (Wilkinson 1965) can be implemented on parallel machines. However, the computational cost of these methods scales as N^4. Hence, in the case of matrix diagonalization the additional power of a parallel computer would be completely negated by the reduction in efficiency of the numerical algorithm. To achieve a real gain from the use of parallel computers it is crucial to develop numerical methods that have the lowest possible scaling of computational cost with system size. Only after this has been achieved will an increase in computational power yield a significant increase in the size of system accessible to the *ab initio* technique.

In the remainder of this paper we concentrate on just one *ab initio* modelling method, the total energy pseudopotential method. In §2 it will be shown how a number of algorithmic developments in recent years have drastically improved the computational efficiency of this method. The implementation of total energy pseudopotential calculations on parallel machines will be described in §3 and prospects for the future are briefly summarized in §4.

2. The total energy pseudopotential method

A number of reviews (Cohen 1984; Joannopoulos 1985; Pickett 1989) provide an overview of the total energy pseudopotential method. Technical details can be found in Ihm *et al.* (1979) and Denteneer & van Haeringen (1985). Recent algorithmic developments are described in detail in Payne *et al.* (1992). The elements of a total energy pseudopotential calculations are as follows: the ions are represented by pseudopotentials (Phillips 1958; Heine & Cohen 1970) so that only the wavefunctions of the valence electrons are included in the calculation; density functional theory (Hohenberg & Kohn 1964; Kohn & Sham 1965) is used to represent the effects of electron–electron interactions; calculations are performed on a periodically repeated unit cell (referred to as a supercell) so that the electron wavefunctions at each k-point in the Brillouin zone can be expanded in terms of a discrete plane wave basis set as follows

$$\psi_{n,\boldsymbol{k}} = \sum_{\boldsymbol{G}} c_{\boldsymbol{G}} \exp(\mathrm{i}[\boldsymbol{k}+\boldsymbol{G}]\cdot\boldsymbol{r}), \tag{1}$$

where $\boldsymbol{k}$ is the k-point in the Brillouin zone and the summation is over reciprocal lattice vectors. This basis set has to be truncated and the convenient choice is to include all basis states for which $|\boldsymbol{k}+\boldsymbol{G}|$ is less than a cut-off wavenumber, G_{cut}.

The electronic states are given by the self-consistent solutions of the Kohn–Sham equations (Kohn & Sham 1965)

$$(-(\hbar^2/2m)\nabla^2 + V_{\text{ion}}(\boldsymbol{r}) + V_{\text{H}}(\boldsymbol{r}) + V_{\text{XC}}(\boldsymbol{r}))\,\psi_{n,\boldsymbol{k}} = \epsilon_n\,\psi_{n,\boldsymbol{k}}, \tag{2}$$

where V_{ion} is the ionic potential, V_{H} is the Hartree potential, V_{XC} is the exchange-correlation potential and ϵ_n is the Kohn–Sham eigenvalue. Self-consistent solutions are required because the Hartree and exchange-correlation potentials depend on the electronic density, $n(\boldsymbol{r})$.

The earliest implementations of the total energy pseudopotential method used standard matrix diagonalization techniques to obtain the Kohn–Sham eigenstates at a computational cost proportional to N_{PW}^3, where N_{PW} is the number of plane wave basis states used to expand the electronic wavefunctions. This appears to be a reasonable scaling of the computational time with system size. However, the number of plane wave basis states required is at least one hundred times larger than the number of atoms in the unit cell. Restrictions on computer memory and computational time limited such calculations to the order of a few tens of atoms in the unit cell. Furthermore, it is harder to achieve self-consistency as the size of the system increases so that the true scaling of computational time with system size is more accurately represented by the fourth or higher power of the number of atoms in the unit cell. Since only the few occupied Kohn–Sham eigenstates are required to compute the total energy it is clear that a significant saving in computational time can be achieved by using iterative matrix diagonalization techniques and including only the occupied electronic states in the calculation. A number of such methods have been developed but the most significant breakthrough in total energy calculations was achieved by Car & Parrinello (1985). They developed the molecular dynamics method for performing total energy pseudopotential calculations which included several novel techniques for increasing the efficiency of the calculations. In addition to the use of iterative matrix diagonalization techniques the process of iterating to self-consistency was overlapped with the process of determining the Kohn–Sham eigenstates thus decreasing the additional computational cost of achieving self-consistency. The basic operation common to all iterative matrix diagonalization methods involves the multiplication of the trial wavefunction by the hamiltonian matrix. Car & Parrinello significantly increased the speed of this multiplication by dividing the operations between real and reciprocal space. The product of the Kohn–Sham hamiltonian H and the trial wavefunction is given by

$$H\psi_{n,\boldsymbol{k}} = -(\hbar^2/2m)\,|\boldsymbol{k}+\boldsymbol{G}|^2\psi_{n,\boldsymbol{k}}(\boldsymbol{G}) + (V_{\text{ion}}(\boldsymbol{r}) + V_{\text{H}}(\boldsymbol{r}) + V_{\text{XC}}(\boldsymbol{r}))\,\psi_{n,\boldsymbol{k}}(\boldsymbol{r}), \tag{3}$$

where the first term on the right-hand side is the product of the wavefunction and the kinetic energy operator, which is diagonal in reciprocal space, and the second term is the product of the wavefunction and the potential energy operator, which is diagonal in real space if the ions are represented by local pseudopotentials. It requires N_{PW}^2 operations to evaluate the product of a matrix and a vector but by rewriting this product as shown in (2) this cost is reduced to N_{PW} operations for the kinetic energy operator and $16N_{\text{PW}}$ operations for the potential energy operator. The factor of 16 arises because the Fourier transform mesh must be larger than the wavefunction array to avoid 'wrapround' error in the calculation. A further significant improvement is that the hamiltonian matrix can be stored in $17N_{\mathrm{PW}}$ words of memory by dividing it between real and reciprocal space in contrast to the N_{PW}^2 words of memory required to store the complete matrix in a single space. However, to exploit this reduction in computational cost the wavefunction must be transformed from reciprocal space to real space and the product of the wavefunction and the potential energy operator must be transformed from real space to reciprocal space. These transformations can be performed in $16N_{\mathrm{PW}} \ln(16N_{\mathrm{PW}})$ operations using fast Fourier transform techniques.

A number of other algorithms have been developed for performing total energy pseudopotential calculations, most notably conjugate gradients techniques that directly minimise the Kohn–Sham energy functional (Gillan 1989; Teter *et al.* 1989). These methods converge the electronic configuration to its groundstate faster than the molecular dynamics method but exploit all the features of the original method. In all these methods there are two operations that dominate the computational cost: the Fourier transforms and the operation of orthogonalizing the electronic wavefunctions. If there are N_{B} occupied electronic bands, which is typically of the order of $0.01N_{\mathrm{PW}}$, then the total cost of the Fourier transforms scales as $16N_{\mathrm{B}}N_{\mathrm{PW}} \ln(16N_{\mathrm{PW}})$ and the cost of orthogonalizing the wavefunctions scales as $N_{\mathrm{B}}^2 N_{\mathrm{PW}}$. Therefore the cost of the Fourier transforms dominates for small systems and the cost of orthogonalizing the electronic wavefunctions dominates for large systems yielding a computational time that scales as the cube of the size of the system for large systems. The cost of implementing non-local pseudopotentials used to dominate the computational time for large systems, typically the cost was a factor of 10 greater than the cost of that orthogonalizing the electronic wavefunctions. Recently a method has been developed for implementing non-local pseudopotentials in which the computational time scales as $N_{\mathrm{B}}N_{\mathrm{PW}}$ (King-Smith *et al.* 1991) so this operation no longer dominates the computational cost for large systems. Using any of the recently developed methods total energy pseudopotential calculations can now be routinely performed for systems containing up to 100 atoms in the unit cell on workstations and conventional supercomputers.

It has only been possible to provide the briefest overview of recent algorithmic developments in the total energy pseudopotential technique. The implementation of these methods on parallel computers is described in the following section. It is remarkable that the original matrix diagonalization method used for total energy pseudopotential calculations could not be implemented efficiently on a parallel computer but that recently developed algorithms are not only much more efficient but are very well suited to implementation on parallel machines.

3. Implementation of total energy pseudopotential codes on parallel computers

The central quantities in a total energy pseudopotential calculation are the electronic wavefunctions which can be represented by a complex array of the form

$$\Psi(N_{\mathrm{PW}}, N_{\mathrm{B}}, N_{\mathrm{K}}), \qquad (4)$$

where N_{K} is the number of k-points used for Brillouin zone sampling and N_{PW} and N_{B} have been defined above.

The structure of the wavefunction array suggests several methods for dividing the data across a parallel machine, for instance by k-point or by band or by plane waves.

To determine which of these strategies is possible for large systems we must consider the variation of $N_{\mathrm{PW}}, N_{\mathrm{B}}$ and N_{K} with the number of atoms in the unit cell, N_{A}. These are:

$$\left.\begin{aligned} N_{\mathrm{PW}} &\approx (100\text{–}1000)\,N_{\mathrm{A}}, \quad N_{\mathrm{B}} \approx N_{\mathrm{A}}, \\ N_{\mathrm{K}} &\approx \begin{cases} 100/N_{\mathrm{A}}, & N_{\mathrm{A}} \leqslant 100, \\ 1, & N_{\mathrm{A}} \geqslant 100. \end{cases} \end{aligned}\right\} \tag{5}$$

It is obvious from these scalings that parallelization by k-point is not possible for large systems. However, it does appear possible to parallelize by band. If this strategy is adopted it is necessary to store the hamiltonian 'matrix' and a number of other arrays of similar size on each compute node. To store these arrays a MWord of memory per node would be needed for systems containing of the order of 100 atoms in the unit cell and calculations for larger and larger systems would require more and more memory per compute node. This is uneconomic and hence parallelization by band cannot be adopted. There is no alternative but to distribute the plane wave basis states of each wavefunction over the machine. To determine the most efficient choice for the distribution it is necessary to consider the operations that are performed on the wavefunctions. As described in the previous section, the efficiency of modern algorithms relies on the division of operations between real and reciprocal space and the use of fast Fourier transform (FFT) techniques to transform between the two spaces. Unfortunately, the FFT is a highly non-local operation and hence it places extreme demands on the communications system of the parallel computer. Apart from the FFT all the operations required to perform a total energy calculation are local (or localized in the case of the most efficient implementation of non-local pseudopotentials) in either real or reciprocal space. The actual distribution of the data is irrelevant provided that the distribution of the electronic wavefunctions and the hamiltonian in each space are identical. In this case, local operations require no communication between nodes. Therefore, the distribution of the wavefunctions is determined purely by the requirements of the FFT. Since the properties of each individual machine will determine the most efficient distribution of the wavefunctions we can only proceed by considering specific examples. We consider only machines that are now available on which total energy pseudopotential codes have been successfully implemented.

The first example considered is the Connection Machine (Brommer *et al.* 1992). The Connection Machine is an example of a massively parallel computer that contains a very large number of compute nodes of modest performance. Although the performance of a single processor may be only a fraction of 1 MFlop the combination of tens of thousands of processors yields a theoretical performance in excess of 10 GFlop. The Connection Machine is a particularly interesting example as it was designed to perform FFTS particularly efficiently and thus the connectivity of the communications system is well suited to this use. A library of microcoded FFT subroutines exists for performing FFTS and the distribution of the wavefunctions is dictated by these routines. Implementation of total energy pseudopotential codes on the Connection Machine can be achieved by rewriting code in FORTRAN 90 using vector oriented Fortran 90 statements to ensure that relevant operations are performed in parallel across all the nodes of the machine and calling the FFT library where appropriate.

The other class of machine that will be considered is typified by the machines manufactured by Intel and Meiko, which consist of a relatively modest number of

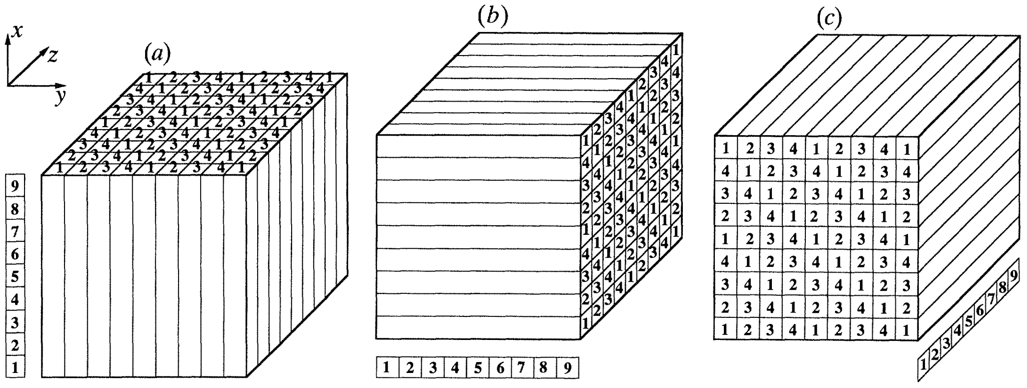

Figure 1. Illustration of the distribution of data over processors for a sequence of three one-dimensional Fourier transforms in (*a*) x, (*b*) y and (*c*) z directions required to perform a three-dimensional Fourier transform. The figure shows a $9 \times 9 \times 9$ Fourier transform performed on a machine with four compute nodes. The numbers in boxes (1–9) label the points of the Fourier transform and the node indices (1–4) show which processor performs the one-dimensional transform.

powerful compute nodes. In both cases the compute nodes are Intel i860 processors capable of a performance in excess of 40 MFlop. There are no fully distributed three-dimensional FFT routines available on these machines and the majority of the effort involved in implementing total energy codes on these machines involves writing programs to perform the FFT (Clarke *et al.* 1992). However, this does provide the freedom to choose a distribution of data that minimizes the communications requirement of the FFT. All multi-dimensional FFTs can be performed as a sequence of one-dimensional transforms. Each one-dimensional transform requires a large exchange of data but this can be avoided by ensuring that the data for each transform lies on the same compute node. Therefore, a distribution of data by columns along the direction of the transform ensures that each one-dimensional transform can be performed without inter-processor communication. However, a global exchange of data is required between each set of one-dimensional transforms to redistribute the data by columns along the next transform direction. The distribution of data for a transformation from a reciprocal space to real space is illustrated schematically in figure 1. The data is initially distributed over the processors by columns along the x direction, as shown in figure 1*a*. The first set of one-dimensional Fourier transforms are performed along the x direction. This is followed by the first global exchange of data between processors so that the data is now arranged by columns along the y direction, as illustrated in figure 1*b*. The one-dimensional transforms along the y direction are then carried out. A second global exchange of data is then performed so that the data is finally arranged by columns along the z direction, as illustrated in figure 1*c*. The final set of one-dimensional transforms along the z direction are then performed. At the end of this sequence of steps the transformation from reciprocal space to real space is complete.

If the communications performance of the machine is relatively poor and the number of processors is not greater than the size of the largest one-dimensional Fourier transform size a significant increase in efficiency can be achieved by combining the final two steps of the above sequence by distributing the data by yz planes, as illustrated in figure 2. The first step of the three-dimensional Fourier

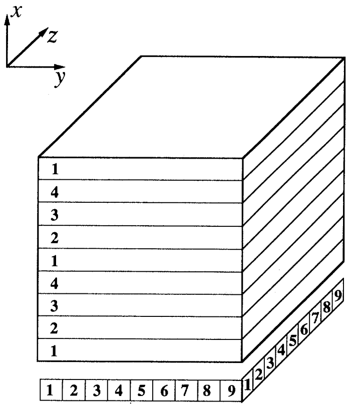

Figure 2. Distribution of data over nodes to replace the transforms shown in figure 1*b*, *c*. The data is distributed by *yz* plane and the *y* and *z* Fourier transforms are performed without redistribution of data over processors.

transform proceeds as above then a global exchange of data is performed to arrange the data as shown in figure 2. The transformations in the *y* and *z* directions can then be performed without further redistribution of data. This scheme thus halves the communications requirement of the FFT compared with the first technique and is useful if the number of processors is relatively modest and the communications are slow compared with the compute speed of the processors.

It is well known that the performance of computer codes on vector machines is crucially dependent on the ratio of vector to scalar operations. The performance of codes on parallel machines is even more critically dependent on the ratio of parallel to sequential operations. Essentially this ratio must be greater than N:1 to benefit from the use of an N processor machine. It is clear that even the very smallest fraction of sequential operations will limit the size of machine on which calculations can be usefully performed. Figure 3 shows the operations involved in a total energy pseudopotential calculation performed using the conjugate gradients method. The letters in the boxes on the left of the figure show whether the operation is performed in real or Fourier space and the letters in the boxes on the right of the diagram show whether the operation is performed sequentially or in parallel. It can be seen that only a small number of set-up operations have to be performed sequentially. In particular the conjugate gradients loop shown in figure 3*b*, which dominates the computational cost of the total energy pseudopotential calculation, is executed totally in parallel. Unfortunately this is not always the case. In the case of metallic systems it is necessary to transform the trial wavefunctions to the Kohn–Sham eigenstates to determine the occupancies of the electronic bands. This transformation is determined using conventional matrix diagonalization routines which cannot be implemented efficiently on parallel machines. This technical problem has yet to be overcome.

The performance of total energy pseudopotential codes on the Daresbury Laboratory Intel iPSC/860 and the 'Grand Challenge' Meiko i860 Computing Surface at the University of Edinburgh are shown in figure 4. These timings are for small systems, containing just 64 silicon atoms in the unit cell, to allow comparison with the performance of a conventional supercomputer, one processor of a CRAY X-MP. Figure 4*a* is for the Meiko i860 computing Surface and figure 4*b* is for the Intel

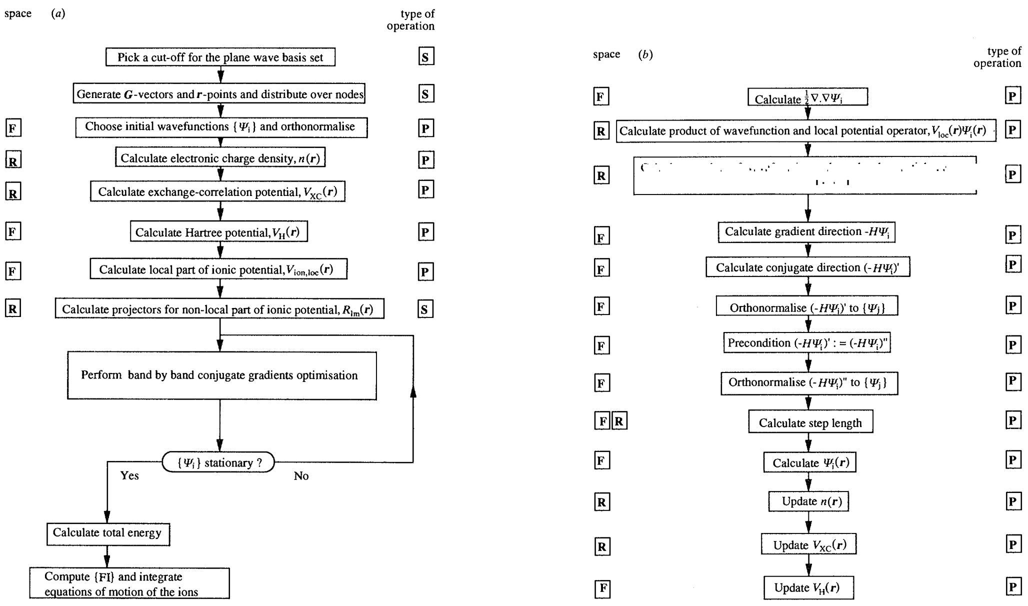
space
(a)
type of operation
Pick a cut-off for the plane wave basis set
Generate G-vectors and r-points and distribute over nodes
Choose initial wavefunctions {Ψ_i} and orthonormalise
Calculate electronic charge density, n(r)
Calculate exchange-correlation potential, V_XC(r)
Calculate Hartree potential, V_H(r)
Calculate local part of ionic potential, V_ion,loc(r)
Calculate projectors for non-local part of ionic potential, R_lm(r)
Perform band by band conjugate gradients optimisation
{Ψ_i} stationary ?
Yes
No
Calculate total energy
Compute {FI} and integrate equations of motion of the ions
(b)
Calculate ½∇.∇Ψ_i
Calculate product of wavefunction and local potential operator, V_loc(r)Ψ_i(r)
Calculate gradient direction -HΨ_i
Calculate conjugate direction (-HΨ_i)'
Orthonormalise (-HΨ_i)' to {Ψ_j}
Precondition (-HΨ_i)' : = (-HΨ_i)''
Orthonormalise (-HΨ_i)'' to {Ψ_j}
Calculate step length
Calculate Ψ_i(r)
Update n(r)
Update V_XC(r)
Update V_H(r)
F
R
S
P

Figure 3. Flow diagram of total energy pseudopotential calculation showing (*a*) principle operations and (*b*) band by band conjugate gradients optimization. The letters in boxes on the left of the diagrams show whether the operation is performed in real space (R) or Fourier space (F) and the letters in boxes on the right of the diagram show whether the operation is performed in parallel (P) or sequentially (S).

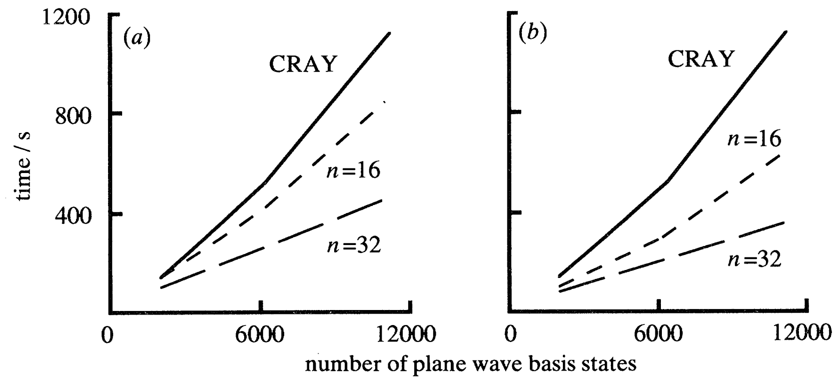

Figure 4. Variation of CPU time with number of plane wave basis states for total energy pseudopotential calculation for a 64 atom silicon cell. Figures show timings for 16 and 32 processors of (*a*) Meiko i860 Computing Surface and (*b*) Intel iPSC/860 against a single processor CRAY X-MP.

iPSC/860. The figures show the time required for one iteration of the conjugate gradients loop as a function of the number of plane wave basis states for $n = 16$ and $n = 32$ processors. It is clear that if the calculation is too small there is no advantage in using a larger number of compute nodes since the computational time is dominated by communications rather than compute speed. However, for larger calculations the advantage of using more processors is clearly shown in the figures. In particular, the Intel iPSC/860, which slightly outperforms the Meiko because the communications system is better suited to our particular application, 32 processors provide a performance three times faster than the CRAY X-MP.

4. The future

Already total energy pseudopotential calculations for 400 atom systems (in unit cells of 800 atomic volumes to allow for the presence of a surface) have been performed on a 64 node Meiko i860 Computing Surface (Stich *et al.* 1992) and a 16K Connection Machine (Brommer *et al.* 1992). These machines yield compute speeds on the total energy pseudopotential codes of the order of 1 GFlop. Intel have now installed the Delta machine at Caltech. This machine has 570 processors and a communication performance 10 times faster than that of the iPSC/860. This machine offers both compute and communications performance 10 times greater than the Meiko i860 Computing Surface used for the 400 atom calculations. Hence, the Delta machine would allow calculations to be performed for systems containing in excess of 1000 atoms. Intel have now announced the Paragon XP/S System which offers processing speeds of up to 300 GFlop and inter-processor communications speeds of 280 MBytes s^{-1}, a hundred times faster than the iPSC/860. This machine would be capable of carrying out calculations on systems containing several thousand atoms in the unit cell. With this machine many of the complex processes that require *ab initio* investigation will be within the power of the total energy pseudopotential method. Furthermore, these figures show that the teraFlop machine is an achievable goal.

This work was performed as part of the 'Grand Challenge' collaborative project, coordinated by Professor D. J. Tildesley. We acknowledge financial support from the Science and Engineering Research Council under grants GR/G32779, GR/G49524 and B18534:NACC. We are grateful for the help we have received from Dr N. M. Harrison and Dr W. H. Purvis at the Daresbury Laboratory, from Dr K. C. Bowler and Dr S. P. Booth at Edinburgh University, and from Mr D. Hewson at Meiko Scientific Ltd. We also thank Dr C. M. M. Nex for useful discussions.

References

Brommer, K., Needels, M., Larson, B. E. & Joannopoulos, J. D. 1992 *Phys. Rev. Lett.* **68**, 1355.
Car, R. & Parrinello, M. 1985 *Phys. Rev. Lett.* **55**, 2471.
Clarke, L. J., Stich, I. & Payne, M. C. 1992 *Comp. Phys. Commun.* (In the press.)
Cohen, M. L. 1984 *Phys. Rep.* **110**, 293.
Denteneer, P. & van Haeringen, W. 1985 *J. Phys.* C **18**, 4127.
Gillan, M. J. 1989 *J. Phys.* **1**, 689.
Heine, V. & Cohen, M. L. 1970 *Solid State Phys.* **24**.
Hohenberg, P. & Kohn, W. 1964 *Phys. Rev.* **136**, B864.
Ihm, J., Zunger, A. & Cohen, M. L. 1979 *J. Phys.* C **12**, 4409.
Joannopoulos, J. D. 1985 In *Physics of disordered materials*, p. 19. Plenum.
King-Smith, R. D., Payne, M. C. & Lin, J-S. 1991 *Phys. Rev.* B **44**, 13063.
Kohn, W. & Sham, L. J. 1965 *Phys. Rev.* **140**, A1133.
Payne, M. C., Teter, M. P., Allan, D. C., Arias, T. & Joannopoulos, J. D. 1992 *Rev. Mod. Phys.* (In the press.)
Phillips, J. C. 1958 *Phys. Rev.* **112**, 685.
Pickett, W. 1989 *Comp. Phys. Rep.* **9**, 115.
Stich, I., Payne, M. C., King-Smith, R. D., Lin, J-S. & Clarke, L. J. 1992 *Phys. Rev. Lett.* **68**, 1351.

Teter, M. P., Payne, M. C. & Allan, D. C. 1989 *Phys. Rev.* B **40**, 12255.
Wilkinson, J. H. 1965 *The algebraic eigenvalue problem.* Oxford: Clarendon.

Discussion

A. M. Stoneham (*Harwell Laboratory, Didcot, U.K.*): In addition to your approach to increasing the number of atoms or electrons considered, there are other routes to the really useful level, namely embedding (of which Green's function methods are a special case) and mesoscopic modelling (Harding, this volume).

M. C. Payne: We agree that the two approaches suggested are useful. Even with thousands of atoms in a unit cell *ab initio* calculations will still be limited to the nanometre lengthscale and alternative methods must be used to study longer lengthscale systems. Mesoscopic modelling attempts to study such systems by parametrizing the behaviour of the system on these longer lengthscales. The values of the parameters should be determined by independent experiment (rather than adjusting the parameters so that the model 'agrees' with experiment) but in many cases this is not feasible and then the results of the modelling may be questionable. *Ab initio* calculations provide an alternative method for determining the values of physical parameters that cannot be determined experimentally and so combining *ab initio* calculations with mesoscopic or macroscopic modelling provides a technique for extending *ab initio* studies to large systems and makes such modelling significantly more realistic and rigorous. Embedding schemes provide an efficient method for studying isolated defects in bulk systems. However, present embedding schemes only allow ionic relaxation within the embedding surface as this surface separates the defective region from the perfect bulk. In favourable cases, such as some close-packed metal surfaces where ionic relaxation is essentially limited to one or two atomic layers, only a few atoms are required within the embedding surface. Less favourable situations might require as many as a thousand atoms within the embedding surface and considerable work is required before enbedding techniques can be applied to systems containing this number of atoms.