Conjugate-Gradient Based Electronic Structure Calculations on the Cray T3E and SGI PowerChallenge [*]

Bernd G. Pfrommer[†] Steven G. Louie[†] Horst Simon[‡]

Abstract

Electronic structure calculations based on Density Functional Theory (DFT) are an important branch of computational physics. We present a convergence analysis for the conjugate-gradient minimization of a Kohn-Sham energy functional, using a functional with implicit orthonormality constraints. We discuss the effects of preconditioning, and show with a numerical example that preconditioning makes the rate of convergence sensitive to the choice of the shift η. We also present performance numbers for the 3d-FFT and matrix-matrix multiplies which are involved in the conjugate-gradient minimization within a pseudopotential/plane-wave approach. We compare the application performance of the SGI PowerChallenge as a typical symmetric multiprocessor to the one of the scalable Cray T3E at NERSC.

1 Introduction

The enormous increase in available computing power is certainly one of the key factors for the recent advances in computational condensed matter theory. Using density functional theory (DFT) in the local density approximation (LDA) or combined with generalized gradient approximations (GGAs), it is now possible to compute with sufficient accuracy the ground state properties of quantum-mechanical systems up to hundreds of atoms large. A wealth of information can be extracted from these CPU-intensive calculations, e.g. binding energies, stable configurations, magnetic and dielectric properties, and insight into electronic properties such as the charge density, band structure, and the quantum-mechanical wave functions.

To reduce the computational effort, we use pseudopotentials to represent the atomic nucleus and the core electrons, leaving only the set of m valence wave functions $\{|\psi\rangle\}$ to be dealt with. Those are determined by the requirement that they minimize the DFT/LDA energy functional

$$E_0 = \min_{\{|\psi\rangle\}} E[\{|\psi\rangle\}] \tag{1}$$

subject to the constraint that the $|\psi_k\rangle$ be mutually orthogonal and normalized. With a suitable construction of the pseudopotentials, the wave functions in (1) can be represented in a plane-wave basis of size N, thus turning (1) into a minimization problem of dimension Nm, with N often as large as 10^5, but m being substantially smaller, typically 1/1000 to 1/100 of N.

[*]Supported by the National Science Foundation under Grant No. DMR-9520554, and by the Office of Energy Research, Office of Basic Energy Sciences, Materials Sciences Division of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098. Computer time was provided by the NSF at the NCSA in Illinois and by NERSC at LBNL.

[†]Department of Physics, University of California at Berkeley, CA.

[‡]NERSC Division, Lawrence Berkeley National Laboratory, Berkeley, CA.

Fortunately, the derivative

$$\frac{\delta E}{\delta \langle \psi_i |} = \hat{H} | \psi_i \rangle \tag{2}$$

(not including the constraints!) is available to aid the search for the minimum. In a plane-wave basis, the Hamiltonian operator $\hat{H}$ turns into the $(N \times N)$ matrix H, which can easily become too large to fit into memory. However, most iterative schemes require only a rule describing how to apply $\hat{H}$ to a given wave function, thus removing the need for explicit storage of H. The separable Kleinman-Bylander pseudopotentials allow for a particularly efficient representation of $\hat{H}$, namely a decomposition of the form:

$$\hat{H} = \hat{T} + \sum_{k=1}^{n_{nloc}} (-1)^{\alpha_k} |\phi_k\rangle\langle\phi_k| \quad + \quad \hat{V}_{local} \qquad\qquad \text{where } \alpha_k = 0, 1 \ . \tag{3}$$

In Eq. (3), the first term $\hat{T}$ is the kinetic energy operator, which is proportional to the Laplacian and therefore diagonal in a plane wave basis $\propto e^{i\vec{G}\vec{r}}$. However, it grows like G^2 with the wave number $G = |\vec{G}|$, and gives the matrix H its characteristic diagonally-dominant shape. It also motivates our convention for arranging the plane wave basis according to increasing G. That way, the lower right corner of H will be dominated by the diagonal elements arising from $\hat{T}$. The second so-called nonlocal term in (3) consists of a sum of $n_{nloc} \ll N$ projectors $|\phi_k\rangle\langle\phi_k|$ onto one-dimensional subspaces, and requires only $\mathcal{O}(n_{nloc} m N)$ operations to be applied to a set of trial wave functions. Finally, the local potential term $\hat{V}_{local}$ in (3) is diagonal in a realspace basis. Therefore, it is applied to a trial wave function $|\psi_{trial}\rangle$ by first inverse Fourier transforming $\psi_{trial}(\vec{G}) \to \psi_{trial}(\vec{r})$ (operation count $\mathcal{O}(N \log N)$), then multiplying with a diagonal operator in realspace (operation count $\mathcal{O}(N)$), and finally Fourier transforming back to the plane wave basis. Since $n_{nloc} \approx m$, and m and N are proportional to the number of atoms N_{atom}, the second term applied to a set of wave functions requires $\mathcal{O}(N_{atom}^3)$ operations[1]. The local term only scales like $\mathcal{O}(N_{atom}^2 \log N_{atom})$, but has a large prefactor in front, such that it dominates the computational cost for systems with less than 50 atoms.

Given the gradient (2), the minimization problem (1) can be tackled with a standard optimization scheme. The use of the popular conjugate gradients method is impeded by the need to include the orthonormality constraints[2]. We follow the work of Mauri et al [1] and use a functional with implicit orthonormality constraints, which allows us to use a textbook conjugate gradients scheme. In section 3, we exploit this to assess the convergence properties of the algorithm for a representative, but small model problem. A numerical example verifies our analysis and demonstrates the effect of preconditioning.

Even with the efficient representation (3), the task of minimizing E in (1) is formidable. Only the use of parallel computers will allow to explore the very large, interesting systems such as surfaces, interfaces, alloys, amorphous solids, liquids, and large molecules. There is agreement that the best way to parallelize the minimization problem is by partitioning all vector quantities (i.e. wave functions) according to their plane wave basis indices[4]. Then, dot products between two vectors are completely local, and only require a global summation at the end. Concerning the parallel performance, the challenge lies in the Fourier transform. We will show that even with an MPI explicit message passing approach, a high-bandwidth

[1]By exploiting the locality of the projectors in realspace, the scaling of this term can be reduced to $\mathcal{O}(N_{atom}^2)$. We have not implemented this optimization yet, since maintaining orthonormality between the wave functions also scales like N_{atom}^3, thus reducing the benefits.

low-latency interconnect like the one of the Cray T3E is necessary to give good application performance with a larger number of processors. We also compare the performance of the T3E to the one of the SGI PowerChallenge as a prototypical symmetric multiprocessor.

2 Unconstrained Energy Functional

Although originally invented to solve large systems of equations iteratively, often the conjugate gradient algorithm is also the method of choice for finding the minimum of multi-variable functions. It requires only the knowledge of the first derivative, and, because of its moderate storage requirements, is particularly well suited when the dimensionality N_{dim} of the parameter space is large. For quadratic forms, the convergence becomes super-linear when the number of iterations approaches N_{dim} – a condition never realized in problems of the size considered here, where $N_{dim} = mN \approx 10^3 \dots 10^5$. Nevertheless, conjugate gradients are robust and efficient, and have been widely used to compute the electronic structure of solids by means of a direct minimization of the energy functional[2].

The inclusion of the orthonormality constraints, which are explicitly present in the original DFT/LDA functional of equation (1), requires a modification of the conjugate-gradient scheme [2]. In contrast to this, we follow the approach by Mauri et al[1], where a functional $E[\{|\psi\rangle\}]$ is adopted which has the orthonormality constraints included implicitly:

$$(4)\quad E_0 = \min_{\{|\phi\rangle\}} E[\mathbf{A}, \{|\phi\rangle\}] = 2\left(\sum_{ij=1}^{m} A_{ij}\langle\phi_j| - \frac{1}{2}\nabla^2|\phi_i\rangle + F[\tilde{\rho}]\right) + \eta\left(N_{el} - \int d\mathbf{r}\tilde{\rho}(\mathbf{r})\right).$$

In Eq. (4), we have the matrix $\mathbf{A} = 2\mathbf{I} - \mathbf{S}$, with $\mathbf{I}$ being the identity, and $\mathbf{S}_{ij} = \langle\phi_i|\phi_j\rangle$ the overlap matrix between the wave functions. $N_{el} = 2m$ is the number of electrons, and $F[\tilde{\rho}]$ represents the ionic, exchange-correlation, and Hartree energy of the Kohn-Sham [5] functional, which depend on the modified charge density $\tilde{\rho}(\vec{r}) = 2\sum_{ij}^{m} A_{ij}\phi_j^*(\vec{r})\phi_i(\vec{r})$ (the factors of two in front of the sums takes care of the spin degeneracy). The wave functions $\{|\phi\rangle\}$ are *not constrained* to be orthonormal. However, one can show[1] that a) the minimum E_0 of the unconstrained functional (4) is the same as in Eq. (1), and b) at the minimum, the wave functions $\{|\phi\rangle\}$ become orthonormal, i.e. minimization of (4) automatically yields a set of orthonormal wave functions. The parameter η in (4) has to be chosen such that the Hessian matrix associated with the minimization problem (4) becomes positive definite. In section 3 we will have a closer look at the impact of η on the performance of the conjugate gradient algorithm used to minimize (4).

3 Convergence Analysis

Due to the large dimensionality of the parameter space, it is important to understand the performance of the algorithm when the optimization problem (4) is approached. For the conjugate gradients scheme, it is possible to get rigorous upper bounds[3] to the error $\rho_k = E^{(k)} - E_0$, where $E^{(k)}$ is the energy computed with the trial wave functions at iteration step k:

$$(5)\qquad \rho_k \leq 2\left(\frac{\sqrt{c}+1}{\sqrt{c}-1}\right)^{-k}\rho_0 .$$

In Eq. (5) the condition number c of the Hessian matrix associated with (4) appears. To get an efficient scheme and a high rate of convergence $r = (\sqrt{c}+1)/(\sqrt{c}-1)$, we want the condition number c to be close to 1, in other words the spread of the Hessian matrix should be as small as possible. In this section, we will extend the convergence analysis by Mauri et al [1], and discuss the effects of preconditioning.

To simplify matters, we first assume the so-called non-selfconsistent case, where the gradient $\hat{H}$ does not depend on the charge density. This corresponds to minimizing the simplified unconstrained energy functional:

$$E_{non-sc}[\{|\phi\rangle\}] = 2\sum_{ij=1}^{m}(2I_{ij} - \langle\phi_i|\phi_j\rangle)\langle\phi_j|(\hat{H} - \eta\hat{I})|\phi_i\rangle \ . \tag{6}$$

The operator $\hat{H}$ is still the same as in (3), but now does not depend on the wave functions. It turns out that E_{non-sc} is minimized by the eigenfunctions $|\chi_i\rangle$, $i = 1, \ldots m$ of the matrix H which have the smallest eigenvalues ϵ_i. We will henceforth call those the occupied eigenfunctions, since the corresponding electronic levels are occupied. From a more mathematical point of view, (6) provides a convenient means of finding the smallest eigenpairs of a large hermitian matrix H. The relationship of the conjugate gradient approach (6) with popular Krylov subspace methods such as the Davidson and Lanczos schemes will be subject of future work.

Since the minimum of E_{non-sc} is given by eigenfunctions of H, we can find the Hessian matrix of E_{non-sc} by performing an *a-posteriori* analysis. Following Mauri et al[1], we expand the occupied wave functions around the minimum in terms of the complete basis of *all* N eigenfunctions of H, which we label in order of increasing eigenvalues:

$$|\phi_i\rangle = |\chi_i\rangle + \sum_{l=1}^{N} c_l^i |\chi_l\rangle \ . \tag{7}$$

We further restrict ourselves to real wave functions, and therefore the expansion coefficients c_l^i in (7) are real. By substituting (7) into (6) and using $\hat{H}|\chi_l\rangle = \epsilon_l|\chi_l\rangle$, we obtain *to second order* in the expansion coefficients:

$$\Delta E_{\text{non-sc}} = 2\sum_{i=1}^{m}\sum_{k>m}^{N}(\epsilon_k - \epsilon_i)(c_k^i)^2 + \sum_{i=1}^{m} 8(\eta - \epsilon_i)(c_i^i)^2 + \sum_{i,j>i}^{m} 8(\eta - \frac{\epsilon_i + \epsilon_j}{2})\left(\frac{c_i^j + c_j^i}{\sqrt{2}}\right)^2 \ . \tag{8}$$

The first term on the r.h.s. of (8) is determined only by the spectrum of H, and does not contain η. The smallest and largest eigenvalues it contributes to the Hessian matrix $\mathcal{H}$ of (6) are given by $\epsilon_{m+1} - \epsilon_m$ and $\epsilon_N - \epsilon_1$, respectively. In other words, the separation between the occupied and the unoccupied, and the spread of H determine the eigenvalues entering $\mathcal{H}$ from the first term. The second term on the r.h.s of (8) is directly related to the implicit *normalization* constraints, whereas the third term is due to the implicit *orthogonalization* constraints between different wave functions. To see what new eigenvalues enter $\mathcal{H}$ due to the orthonormality constraints, it is sufficient to just analyze the second term, as it also captures the extreme eigenvalues brought about by the third term.

First of all, η must be chosen such that $(\eta - \epsilon_m) > 0$ in order to get the Hessian matrix to be positive definite. To get good performance, η should be picked such that the eigenvalues of $\mathcal{H}$ stemming from the second term fall within the range of eigenvalues already present from the first term:

$$\frac{\epsilon_{m+1} - \epsilon_m}{4} + \epsilon_m \leq \eta \leq \frac{\epsilon_N - \epsilon_1}{4} + \epsilon_1 \tag{9}$$

With this choice of shift parameter, the condition number of the Hessian matrix is just given by the spectrum of H, and the conjugate gradient scheme should perform optimally according to (5).

We point out that equation (8) is in disagreement with Mauri et al[1], who have an additional factor of two in front of the second and third term. We also note that there is another set of modes corresponding to $(c^i_j - c^j_i)/\sqrt{2}$, but those have zero eigenvalues, and therefore do not appear in Eq. (8).

We now briefly discuss the effects of preconditioning on the performance of the algorithm. Preconditioning requires a matrix $\mathcal{K}$ which, when applied from the left to the Hessian matrix $\mathcal{H}$, brings the condition number of $\mathcal{KH}$ as close as possible to one. Preferably, the application of $\mathcal{K}$ to a set of trial wave functions should not increase the operation count significantly. For the problem we are considering here, indeed such a preconditioner exists[2]. It is based on the fact that the matrix H is diagonally dominant in the lower right corner, and therefore an approximate inverse exists for this part of the matrix. Since H enters $\mathcal{H}$ via the first term of Eq. (8), it is possible to construct[2] a $\mathcal{K}$ which will reduce the large ϵ_k's in the first term of (8), and thereby the maximum eigenvalues of $\mathcal{H}$. The downside is that reducing the spread in eigenvalues arising from the first term renders it more difficult to pick an optimal η using equation (9), as we will show in the next section.

4 A numerical example

It is instructive to look at a simple, but relevant example for testing the statements of the preceding section. Here, we study the performance of the conjugate-gradient algorithm applied to a diamond crystal. Since there are two atoms in the unit cell with four valence electrons each, one needs to compute $m = 4$ wave functions (spin degeneracy) at each $\vec{k}$-point in the Brillouin zone. We just consider the center of the Brillouin zone ($\vec{k} = 0$), and also restrict our study to the non-selfconsistent functional (6). This way, we are essentially computing the smallest eigenpairs of the Hamiltonian matrix H, which has a size of $N = 609$. This is much smaller than typical problem sizes studied today, but it allows to use MATLAB and an explicit representation of H for numerical experiments.

We first perform a direct diagonalization of the full matrix to get the spectrum shown in the inset of figure 1. The smallest four (occupied) eigenvalues are grouped into a low-lying single eigenvalue and a triplet. They are well separated from the higher, unoccupied eigenvalues. This gap is critical for achieving fast convergence. Using Eq. (9) we should get optimal performance for $2.02 \leq \eta \leq 15.42$. The starting guess for the conjugate gradient procedure is generated by diagonalizing a 27 by 27 submatrix from the upper left corner of H, and selecting the smallest four eigenpairs. The other (609-27) components of the start vectors are filled up with 0.001*rand() to ensure that the full spectrum is represented in the starting guess.

When sweeping the shift η through, we measure a rate of convergence as shown in figure 1. If the shift approaches $\eta \approx \epsilon_4$, the rate of convergence deteriorates dramatically because the condition number increases. In the case without preconditioning (solid line), indeed the performance is almost constant within the range given by Eq. (9) (indicated by vertical dot-dashed lines).

With preconditioning (dashed line), the rate of convergence is substantially larger, but only for a good choice of η. Since the large eigenvalues of H are now effectively reduced, η must be chosen smaller accordingly. From this numerical experiment it appears that there might not even be an optimal choice of η in this case, in other words there is no shift for which the orthonormality modes do not degrade the performance.

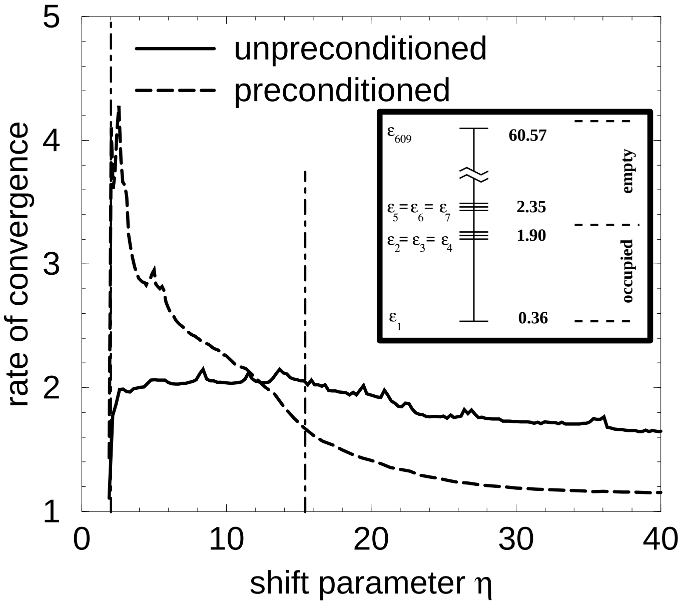

FIG. 1. *The rate of convergence as a function of the shift parameter η for a conjugate gradient algorithm performed on an energy functional with implicit orthogonality constraints. The inset shows the spectrum of the matrix H which appears in the energy functional.*

5 Parallel Performance on the Cray T3E and SGI PowerChallenge

While the previous sections focussed on algorithmic aspects, in this section we report *preliminary* performance numbers regarding the actual implementation of the conjugate-gradient scheme. We gauge the speed on two different platforms: The SGI PowerChallenge and the Cray T3E. Both are based on fast, super-scalar RISC processors: The 16 nodes of the PowerChallenge (at NCSA in Illinois) are equipped with 195 MHz MIPS R10000 processors, whereas the 128 nodes of the T3E have a 300 MHz Alpha EV5 Chip inside. However, in contrast to the PowerChallenge, the T3E is a scalable parallel computer.

As far as the memory hierarchies are concerned, the SGI Power Challenge used for our tests has a unified 2 MB floating point data cache per node, which is also used as a secondary cache for instructions and integer data. The nodes go through a wide, fast, shared snoopy bus to access the global shared memory. The T3E on the other hand is a distributed memory machine, where the individual nodes have 256 MB of memory, and the EV5 Chip has a 8 KB direct-mapped L1 and a 96 KB 3-way set associative L2 cache. An additional streaming buffer between L2 cache and memory is designed to boost performance in case of sequential cache misses, but is not functional at the moment and was therefore *switched off* for our performance tests. The T3E nodes communicate through a scalable high-bandwidth low-latency interconnect.

The application for which the timings were performed is the conjugate gradient algorithm as outlined in the previous section. The test problem is a sample of amorphous hydrogenated carbon with 64 carbon and 12 hydrogen atoms. We use a plane-wave cutoff of 90 Rydbergs which leads to a matrix size of $N = 57,000$. We keep the Hamiltonian H fixed for this test, i.e. we use the conjugate-gradient scheme to find the 21 smallest eigenvalues and the corresponding eigenvectors[2]. Our code is written in Fortran 90 and uses the MPI library for explicit message passing.

Since we parallelize with respect to plane-wave components, each processor holds N/N_{proc} components of each of the m wave functions, where N_{proc} is the number of processors put to work. The matrix-matrix multiplies we timed are the ones for the second (nonlocal) term in (3), and are consequently between rectangular complex matrices of size $(m \times N/N_{proc}) \times (N/N_{proc} \times n_{nloc})$, with $n_{nloc} = 64$. After the local matrix-matrix multiplies have completed, a global summation is carried out with the appropriate MPI reduction function.

The three-dimensional Fast Fourier Transform (3d-FFT) and inverse transform involved in applying $\hat{H}$ to a trial wave function requires a grid of $96 \times 96 \times 96$. The 3d-FFT is performed by means of a series of 1d-FFTs, which are done by calls to optimized library routines (the COMPLIB on the SGI and the LIBSCI on the Cray). It requires a series of 1d-FFTs say first in the z direction, then in the y direction, and finally in the x direction to complete the 3d-FFT. By partitioning the Fourier space so that the processors initially hold only complete lines along the z-direction, we can do the first 1d-FFT along z entirely locally. Before the 1d-FFTs in the y direction, we transpose the grid such that every processor holds a set of lines along the y direction. An analogous transpose is required before the 1d-FFT in the x direction is carried out. The transposition steps are communication intensive, because every processor sends data to all others. We use non-blocking MPI library routines to pass the data between the nodes.

Figure 2 shows the performance *per node* of the SGI PowerChallenge and the Cray T3E for a varying number of processors. On the T3E we had to run on 4 or more processors in order to fit the data into memory. Although the T3E at NERSC has currently 128 nodes, only a maximum of 64 are available at the moment. To compute the MFLOPS, the operation count for a 1d-FFT of length p was assumed to be $5p\log_2(p) - 6p + 6$.

Both machines fall short of their peak performances of 390 MFLOPS/node (SGI PowerChallenge) and 600 MFLOPS/node (Cray T3E). The FFTs are very memory-access intensive, and are not expected to perform too well on RISC-based machines where the memory access times are much longer than on vector architectures. Indeed we find them to run at about 53 MFLOPS/node on 4 T3E processors, decreasing gradually due to communication to about 36 MFLOPS/node on 64 processors. On the PowerChallenge, they run at about 48 MFLOPS/node on a single processor. When running on more processors, the required communication first reduces the performance on the PowerChallenge. On 8 or 16 processors though, one can see the effects of the increased aggregate cache, which leads to super-linear speedup when running on 16 nodes (54 MFLOPS/node). Notice that the absolute performance numbers one would get from a simple benchmark test on a 1d-FFT are much higher, since our number includes not only the communication overhead, but also time-consuming memory copies for the transpose operation and the time to apply the operator $\hat{V}_{loc}$ between the inverse and the forward FFT.

On the matrix-matrix multiplies, the Cray excels with a performance between 237 MFLOPS/node (on 4 processors) and 221 MFLOPS/node (on 64 processors). The performance curve of the PowerChallenge has more structure. Like in the case of the 3d-FFT, we see a super-linear speedup for the matrix-matrix multiplies, but this time much more pronounced. This is most likely also cache related. A simple matrix-matrix multiply

[2]The number of eigenvalues is lower than what would be really required. This is to allow us to fit the problem into the memory of four T3E nodes

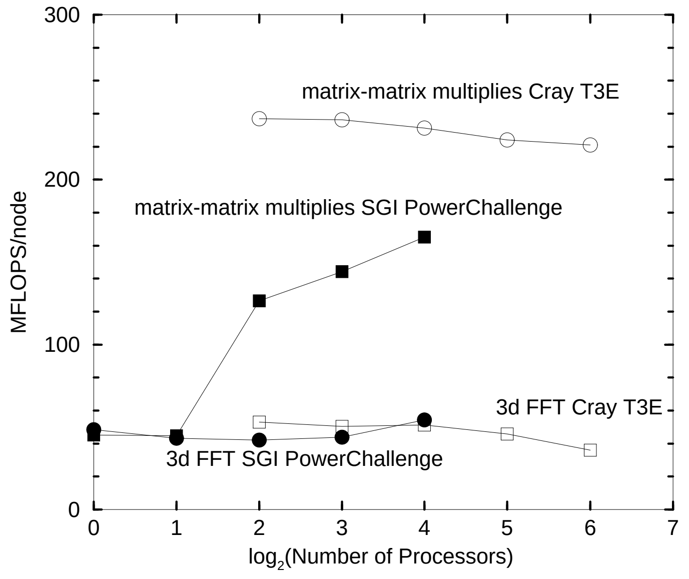

FIG. 2. *Floating point performance in MFLOPS/node on the Cray T3E and the SGI PowerChallenge for varying number of processors. The 3d-FFT has a grid size of* $96 \times 96 \times 96$*. The matrix-matrix multiplies are of size* $(21 \times 57000/N_{proc}) * (57000/N_{proc} \times 64)$.

benchmark reveals that the performance of the PowerChallenge degrades substantially when the shape of the rectangular matrices deviates strongly from the square. This is the case for the matrices multiplied here, because $N/N_{proc} \gg n_{nloc}$ and $N/N_{proc} \gg m$ if N_{proc} is small. Using more processors thus brings the matrices into a more convenient shape. The fact that the Cray T3E is not showing such sensitivity to the shape of the matrix might indicate a superior cache blocking technique of the ZGEMM routine in the LIBSCI.

In summary, we find that on both machines, our implementation shows good parallel performance and demonstrates that plane-wave calculations can make efficient use of parallel platforms with a fast communication system. Further optimization is needed though to increase the single-node performance by avoiding time-consuming memory accesses.

References

[1] F. Mauri and G. Galli, Phys. Rev. B **50**, 4316 (1994)
[2] M.C. Payne, M.P. Teter, D.C. Allan, T.A. Arias, and J.D. Joannopoulos, Rev. Mod. Phys. **64** 1045 (1992)
[3] J. Stoer and R. Bulirsch in *Numerische Mathematik 2*, 3rd edn., Springer-Verlag, Berlin (1990)
[4] L.J. Clarke, I. Stich, and M.C. Payne, Comp. Phys. Comm. **72**, 14 (1992).
[5] W. Kohn, and L.J. Sham, Phys. Rev. **140**, A1133 (1965)