Parallel fast Fourier transforms for electronic structure calculations

Peter D. Haynes and Michel Côté

Theory of Condensed Matter, Cavendish Laboratory, Madingley Road, Cambridge, CB3 0HE, U.K.

We present a new method for performing fast Fourier transforms for electronic structure calculations on parallel computers which minimises the latency cost involved in communication between nodes. We compare the new and traditional methods in theory and in practice, and thus suggest the conditions under which the new method will be more efficient than current methods.
PACS numbers: 71.15.Ap, 02.70.Hm

Key words: Fast Fourier transform.

1 Introduction

The majority of the computational effort in electronic structure calculations is spent solving the Schrödinger equation by diagonalising the Hamiltonian in the representation of a finite basis set. When the size of this basis set greatly exceeds the number of eigenfunctions of interest, then iterative diagonalisation methods are most efficient. The fundamental operation involved in these iterative methods is the operation of the Hamiltonian matrix on a trial eigenvector.

The Hamiltonian consists of the sum of the kinetic and potential operators. Within the pseudopotential approximation [1], the momentum-space formalism [2], in which plane-waves are used as the basis set for the eigenfunctions, has become widespread. In momentum-space, the kinetic operator is diagonal and hence trivially applied to a trial eigenvector. However, the operation of the potential, a multiplication in real-space, is an expensive convolution in momentum-space. The fast Fourier transform (FFT) [3] enables the trial eigenvector to be transformed cheaply between real- and momentum-space so that the potential may also be applied efficiently. The FFT is thus crucial to the viability of the momentum-space formalism. Application of the Hamiltonian matrix to a trial eigenvector thus involves two FFTs, and in most applications it is the FFT which dominates the computational effort.

The implementation of the momentum-space method on parallel computers involves the distribution of plane-waves across the nodes [4]. To perform an FFT, it is thus necessary for all of the nodes to communicate with each other. Even in the largest electronic structure calculations which can be performed with current parallel computers, the 3D FFT grid is relatively small (typically $128 \times 128 \times 128$), as compared with those found in other applications. The amount of data sent between nodes is thus small, so that the latency cost of the communication can be significant, and this prevents the method from scaling well up to very large numbers of nodes.

In this paper, we present a new method for performing FFTs on parallel computers which minimises the latency cost and thus offers the prospect of scaling plane-wave electronic structure calculations up to a larger number of nodes than is currently possible. This method will be most significant in enabling electronic structure calculations to be performed on relatively inexpensive parallel computers consisting of a large number of networked workstations, which are growing in popularity but which also suffer from high latencies.

The rest of the paper is organised as follows: in Sec. 2 we outline those aspects of the derivation of the FFT which are relevant to our description of the new parallel method. In Sec. 3 we briefly describe the current implementation and in Sec. 4 we turn to a description of the new method. The costs of each method and the conditions under which we expect them to be most efficient are compared in Sec. 5. In Sec. 6 we present results for implementations of both methods. We discuss the issue of load balancing in Sec. 7 and in Sec. 8 we draw our conclusions.

2 Fast Fourier transforms

For a full description of the FFT see, for example Ref. [5]. We consider a vector of length n with elements x_k where $0 \leq k < n$. The elements of the discrete Fourier transform (DFT) of this vector, X_k, may then be defined by

$$X_k = \sum_{j=0}^{n-1} \omega_n^{kj} x_j \tag{1}$$

where ω_n is one of the n^{th} roots of unity $\omega_n = \exp(-2\pi \mathrm{i}/n)$. The definition in Eq. 1 is simply a premultiplication of the vector with elements x_j by a matrix with elements $F_{kj} = \omega_n^{kj}$, and this multiplication requires $O(n^2)$ operations.

Consider the following result, known as the Danielson-Lanczos lemma [6], applied to Eq. 1 for even $n = 2m$:

$$
\begin{aligned}
X_k &= \sum_{j=0,\ \text{even}}^{n-1} \omega_n^{kj} x_j + \sum_{j=0,\ \text{odd}}^{n-1} \omega_n^{kj} x_j \\
&= \sum_{j'=0}^{m-1} \omega_n^{k(2j')} x_{2j'} + \sum_{j'=0}^{m-1} \omega_n^{k(2j'+1)} x_{2j'+1} \\
&= \sum_{j'=0}^{m-1} \omega_m^{kj'} x_{j'}^{\mathrm{e}} + \omega_n^k \sum_{j'=0}^{m-1} \omega_m^{kj'} x_{j'}^{\mathrm{o}}
\end{aligned}
\tag{2}
$$

in which we use the result $\omega_n^{2k} = \omega_{n/2}^k$ and introduce the notation $x_k^{\mathrm{e}} = x_{2k}$ and $x_k^{\mathrm{o}} = x_{2k+1}$ to represent the even and odd elements respectively. Equation 2 shows that a DFT of length n may be rewritten as the combination of two DFTs of length $n/2$. This lemma may be applied recursively, as long as the length of the DFT remains even, so that if n is a power of 2, it may be applied until the trivial DFT of a vector of length 1 is required. In this case, the DFT requires only $O(n \log_2 n)$ operations, and this is the (radix-2) FFT. The lemma may be generalised to apply when n contains factors other than 2.

In three dimensions, the definition of the DFT is

$$
X_{k_x k_y k_z} = \sum_{j_x=0}^{n_x-1} \omega_{n_x}^{k_x j_x} \sum_{j_y=0}^{n_y-1} \omega_{n_y}^{k_y j_y} \sum_{j_z=0}^{n_z-1} \omega_{n_z}^{k_z j_z} x_{j_x j_y j_z} \tag{3}
$$

Equation 3 demonstrates that the three-dimensional DFT is the product of 3 one-dimensional DFTs which all commute with each other since they act indepedently.

3 Traditional parallel implementation

The traditional distribution of data in electronic structure calculations is shown schematically in Fig. 1 for the case of a $4 \times 6 \times 8$ grid and 4 nodes. When applying the potential to a trial eigenvector, the data is initially represented in momentum-space (on the left-hand side of Fig. 1) and each node deals with a number of "rods" of data in the z-direction. In the first stage of the 3D-FFT, each node performs a 1D-FFT in the z-direction on each of its rods. The nodes then communicate to effect a transpose in which the data is redistributed from "z-rods" to "y-rods" (middle of Fig. 1). Each node then performs a second 1D-FFT in the y-direction on these rods. A second communication stage transposes the data to "x-rods" (right of Fig. 1), and the final stage is to perform a 1D-FFT on these x-rods. The DFT from real- to momentum-space is performed similarly by reversing these operations.

Fig. 1. Distribution of data for traditional implementation.

4 New parallel implementation

In the traditional method, the communication phases simply redistribute the data across the nodes so that the nodes can perform local 1D-FFTs. By constrast, in the new method we present here the communication phases also play a rôle in the calculation of the 3D-FFT.

Fig. 2. Distribution of data for new implementation.

The distribution of data in the new method is illustrated in Fig. 2, again for a $4 \times 6 \times 8$ grid and 4 nodes. This distribution is motivated by the Danielson-Lanczos lemma. Each node now deals with a set of xy-planes. For simplicity, consider first applying the Danielson-Lanczos lemma recursively to a 1D-FFT. In the first step, the data is partitioned into even and odd elements. In the second step, each of these two sets of data is partitioned into even and odd elements again. Thus one obtains four sets of data: even-even, even-odd, odd-even and odd-odd elemets. In the case of four nodes, we would therefore assign one of these sets to each node, to obtain the pattern shown in Fig. 2. Since the three 1D-FFTs in the x-, y- and z-directions in Eq. 3 commute, this distribution is quite flexible, and an alternative is shown in Fig. 3 in which the Danielson-Lanczos lemma has been applied in both the x- and y-directions.

Fig. 3. Alternative distribution of data for new implementation.

Thus, instead of distributing data as rods as in the traditional method, each node deals with a set of data which is picked uniformly from the full FFT grid. In the case of a distribution across N nodes, each node thus deals with one of the sets of data which would result after $\log_2 N$ applications of the Danielson-Lanczos lemma. Hence, in order to perform the FFT from momentum-space to real-space, each node first performs a local 3D-FFT on its data. It then remains for the nodes to communicate in order to combine this data to complete the full FFT (i.e. to perform the multiplication by a phase factor and addition shown in the last line of Eq. 2). When complete, each node possesses a single contiguous block of real-space data (right of Figs. 2 and 3). Again, the DFT from real- to momentum-space is performed by reversing the above operations.

Another advantage of the new method is that the Hamiltonian matrix is now effectively blocked. The FFT permits the application of the Hamiltonian on state vectors in $O(n \log n)$ operations. The new method distributes the data such that each block of the matrix can be applied in $O\left((n/N)\log(n/N)\right)$. This opens up the possibility of applying iterative diagonalisation algorithms for block matrices to the electronic structure problem to further decrease the amount of communication.

5 Cost comparison

Both of the methods described in Secs. 3 and 4 require the same number of floating point operations, but differ dramatically in their communication patterns. The traditional method requires two transposition or communication phases, and in general, each of these may require each node to communicate with every other node. By contrast, the new method requires $\log_2 N$ communication phases, but in each of these, the nodes communicate in a pairwise fashion.

In the following analysis, we assume that the cost of communicating a packet of data from one node to another consists of two parts. The first part is simply the time taken to transmit the data between the nodes and depends upon the bandwidth of the connection β and the size of the data packet. The second part is a fixed overhead, the latency α, which is independent of the size of the data packet. We define α and β for the situation in which a node is sending a data packet to another node and simultaneously receiving a data packet of the same size from another node.

In the traditional method, each of the two communication phases generally involves each node communicating with every other node. This can be achieved in $N-1$ phases in which all of the nodes simultaneously send a data packet to one node and receive a packet from another (usually different) node. The total latency cost for the traditional parallel DFT is thus $2(N-1)\alpha$. We define $n = n_x n_y n_z$ to be the total number of data elements in the full FFT grid. In the traditional method, each data packet has a size of nu/N^2 where u is the size of a single data element (typically 16 bytes for a double precision complex data type). The total communication time involved in the traditional method, τ_{trad} is thus:

$$
\begin{aligned}
\tau_{\text{trad}} &= 2\,(N-1)\left[\alpha + \frac{nu}{\beta N^2}\right] \\
&\approx 2\left[\alpha N + \frac{nu}{\beta N}\right], \text{ for large } N.
\end{aligned} \tag{4}
$$

In the new method, there are $\log_2 N$ communication phases in which each node communicates with just one other node. The total latency cost for this method is thus $\alpha \log_2 N$. However, in this method, the size of the data packets exchanged is nu/N, a factor of N larger than in the traditional method. The total communication time for the new method, τ_{new} is then:

$$\tau_{\mathrm{new}} = \log_2 N \left[\alpha + \frac{nu}{\beta N}\right] \tag{5}$$

The new method has a lower latency cost, due to the smaller number of data packets sent, but a higher transmission cost due to the larger size of those data packets. We therefore expect the new method to be advantageous in the limit of a large number of processors N. In this limit, many processors need to communicate with each other, but the packets they exchange are very small, so that the latency cost dominates. Since the latency cost of the new method increases only logarithmically instead of linearly, it should scale to a larger number of processors.

The "cross-over" point at which the new method performs more efficiently than the traditional method depends upon the hardware (and low-level software libraries which interface to it), parametrised by α and β, and the size of the FFT grid n. The smaller the FFT grid, the more competitive the new method will be. It is for this reason that the method may be most useful in electronic structure calculations in which the FFT grid sizes are relatively small.

The product $\alpha\beta$ defines the packet size which costs as much in latency as transmission to send. On a cluster of PCs connected by 100 Mbit ethernet this product is of the order of 2 Kbytes. We measured a very similar value for a 64-node SGI Origin 2000 supercomputer. In both cases, the generic Message-Passing Interface (MPI) was used. However, use of lower-level vendor-specific libraries on the Origin would reduce the latency and hence the $\alpha\beta$ product. We therefore anticipate that the new method may be more suitable for implementation on clusters of workstations than on supercomputers designed and built to run programs in parallel.

6 Results

In order to assess the validity of the analysis of Sec. 5 in practice, we compared the communication times for both methods running on the CRAY T3E-1200E at the University of Manchester. This machine was chosen because it enabled us to compare the methods running on up to 512 nodes. However, as mentioned at the end of Sec. 5, we would not expect this machine to give particularly favourable results for the new method because its latency cost is small.

In Fig. 4 we plot the results for two different FFT grid sizes: $128\times128\times128$ and $64\times64\times64$. The implementation of each method used the same communication library call. The results plotted were obtained by averaging the communication times for 50 convolutions (i.e. both a forward and backward FFT) on the 128^3 grid and 100 convolutions on the 64^3 grid. The error bars show the resulting standard deviations. The dashed and dotted lines show the best fits to Eqs. 4 and 5 respectively.

Fig. 4. Comparison of average communication times for both methods applied to $128 \times 128 \times 128$ (left) and $64 \times 64 \times 64$ (right) grid sizes.

The quality of the fits of Eqs. 4 and 5 to the data supports our analysis. In particular, the cross-over occurs at a smaller number of processors for the case of the smaller FFT grid, as expected. On the machine used here, the new method is out-performed by the old method for the reasons given above, and would only be applicable for very small FFT grids or very large numbers (over 1000) of nodes. However, from the quality of the fit to the analysis of Sec. 5, we would expect to be able to apply Eqs. 4 and 5 with confidence to estimate how the methods would scale on other machines. For example, on the cluster of PCs mentioned in Sec. 5, on which we measured a latency of about 300 μs and a bandwidth of 8.7 Mbs^{-1}, the expected cross-over points are 40 and 140 nodes for the 64^3 and 128^3 grids respectively.

7 Load balancing

The completeness of the plane-wave basis set is usually specified by a single parameter, the kinetic energy cutoff E_{cut}. Only those plane-waves with a kinetic energy less than E_{cut} are included in the basis set used to describe the eigenfunctions. The cutoff defines a sphere in momentum-space, which is shown in Fig. 5 in relation to the full FFT grid.

In order to ensure an even division of labour among the nodes (load balancing)

Fig. 5. The sphere of plane-waves included in the basis set in relation to the full FFT grid.

of the parallel computer, it is necessary to assign a roughly equal number of plane-waves to each one. In the traditional method, this is achieved by sorting the z-rods in order of their length (once they have been truncated by the cutoff sphere) and assigning the rods to the nodes in turn according to that order. This inevitably requires some book-keeping which can add to the cost of the traditional method.

We note here that the new method automatically satisifies the demand of load balancing. Since the plane-waves assigned to each node are spread uniformly throughout the grid (see e.g. Fig. 3), the effect of the cutoff sphere is to truncate the set of plane-waves on each node in the same fashion, so that the number of plane-waves on each node remains balanced. In fact, the cutoff sphere on the full FFT grid is transformed into a cutoff sphere on the FFT grid on each node in the new method. Thus the book-keeping cost is only associated with the local 3D-FFT performed on each node, whereas in the traditional method it affects the communication pattern between nodes (e.g. so that the packet sizes are no longer identical) and thus has a more significant effect on performance.

8 Conclusions

We have presented a new method for performing FFTs on parallel computers which scales to a larger number of nodes than the traditional method due to the reduced latency cost. This is achieved by taking advantage of the inherent data distribution required by the FFT algorithm. The method is applicable to electronic structure calculations, due to the small sizes of FFT grids used, and is most effective on clusters of workstations where the communication costs are high. The new method automatically satsifies the demand of load balancing, and effectively blocks the Hamiltonian matrix which may allow new iterative diagonalisation algorithms for block matrices to be applied.

Acknowledgement

We would like to thank N. M. Maclaren, G. J. McMullan and P. R. C. Kent for their assistance. The computational resources for the development of this work were provided by the University of Cambridge High Performance Computing Facility. We acknowledge the support of the U.K. Engineering and Physical Sciences Research Council and Magdalene College, Cambridge (P.D.H.), and of the Natural Sciences and Engineering Research Council of Canada (M.C.).

References

[1] V. Heine, The pseudopotential concept, in: H. Ehrenreich, F. Seitz and D. Turnbull, eds., *Solid State Physics* Vol. 24 (Academic Press, New York, 1970) 1–36.

[2] J. Ihm, A. Zunger and M. L. Cohen, Momentum-space formalism for the total energy of solids, *J. Phys. C* **12** (1979) 4409–4422.

[3] J. W. Cooley and J. W. Tukey, An algorithm for the machine calculation of complex Fourier series, *Math. Comput.* **19** (1965) 297–301.

[4] L. J. Clarke, I. Štich and M. C. Payne, Large-scale ab initio total energy calculations on parallel computers, *Comput. Phys. Comm.* **72** (1992) 14–28.

[5] G. H. Golub and C. F. Van Loan, *Matrix Computations* (John Hopkins University Press, Baltimore, 1996) 188–192.

[6] G. C. Danielson and C. Lanczos, Some improvements in practical Fourier analysis and their application to X-ray scattering from liquids, *J. Franklin Inst.* **233** (1942) 365–380, 435–452.