Metal CONQUEST

Lianheng Tong

2nd March 2011

Abstract

This report describes the work done in the one year Distributed Computational Science and Engineering (dCSE) project aimed to develop an ab initio Density Functional Theory code highly efficient for calculation on metallic systems that allows simulation of 1000s of atoms on high performance computing facilities with reasonable cost. The code is developed on the existing open-source linear scaling code CONQUEST. While the same linear scaling properties associated to insulators and semiconductors cannot be achieved for metallic systems due to the long range interactions in the density, we maximise efficiency by using ScaLAPACK, compact basis sets offered by CONQUEST and various methods for reducing the number of required diagonalisations.

Contents

1 Introduction

The understanding to many problems in material science and nanotechnology involving metallic systems attracting interests today requires calculations to be done on a large scale. While significant advancements has been made in highly scalable and efficient computer codes for insulators and semi-conductors there is comparatively less work done for metallic systems. Although most of the algorithms that make up scalable computer codes for insulators and semiconductors are transferable to metallic systems, such as the efficient evaluation of the Hamiltonian and the use of compact and flexible basis sets, it is not possible to achieve the same scalability for calculations on metals because the density matrix for these systems are generally long ranged, and the conventional linear scaling techniques for insulators that rely on the short range nature of the density matrix no longer applies. The main goal of the project

is to take an existing linear scaling Density Functional Theory (DFT) code designed for insulators and semiconductors (CONQUEST[4]) and modify it so that it can be efficiently applied to metals.

CONQUEST is a suitable basis for this modification because a metallic density matrix solver that involves direct diagonalisation of the Hamiltonian using SCaLAPACK v1.7 is already in place. It is also a mature open source linear scaling code with spectacular scaling with respect to number of processors. CONQUEST uses either B-splines or Pseudo-atomic orbitals to generate a flexible set of functions (support functions) in which to expand the wavefunctions. With B-Spline basis it can achieve plane-wave accuracy with a small number of functions. This is important for efficient metallic calculations, because the cost of diagonalisation depends directly on the number of spanning functions. Further more, CONQUEST has already been used successfully for simulating insulators on HECToR with 4096 processors (supported under the UKCP grant, project e89).

Following changes must be applied to the code for efficient calculation on metallic systems:

1. The density matrix solver cannot be based on the existing linear scaling techniques for insulators and semiconductors, hence the proposal is to use a simple matrix diagonalisation on the Hamiltonian using tools from a mature parallel processing linear algebra library, such as ScaLAPACK. This is already implemented in CONQUEST.

2. Introduce efficient Brillouin zone integration (Methfessel-Paxton method[14]) to reduce the number of diagonalisations need for achieving self-consistency. This is particularly important for metals because of the partially occupied bands which produces a discontinuity in occupation function at Fermi energy—problematic for accurate numerical integrations in Brillouin zone and hence effecting self-consistency.

3. Introduce a more efficient technique for reaching self-consistency in large systems. As the simulated system gets larger, a phenomenon called charge-sloshing which causes the inout and output electron density to oscillate and never reach self-consistency becomes a frequent occurrence for metals. This has been successfully tackled by introducing Kerker preconditioning and Wave-dependent metric in the ab initio code VASP[12] and the same techniques are to be added to CONQUEST.

4. Metallic systems in general have sharp and rapidly varying band-structure, and hence requires a high density of **k** points for accurate Brillouin zone integration. And each **k** point requires a separate diagonalisation. Further efficiency in calculation can be achieved if diagonalisation on several **k** points can be processed in parallel.

5. The metallic calculations on CONQUEST need to be profiled to spot potential bottlenecks, and optimal input parameters need to be found.

With these improvements and the already excellent scaling properties, CONQUEST can become a valuable tool for allowing large scale ab initio electronic structure simulation on metallic materials on HECToR.

This report describes the work done for implementing the above mentioned modifications and the profiling of the code. Section 2 describes the work done on implementation of Kerker preconditioning and wave-dependent metric; section 3 describes the implementation of Methfessel-Paxton Brillouin zone integration technique, in which we also discovered and solved a technical complication that was previously unknown in the field; section 4 gives the results from profiling CONQUEST, and confirms diagonalisation is indeed the bottleneck for the calculation; section 5 describes the work done on **k** parallelisation; and finally a prove of a theorem central to the solution for reliable implementation of the Methfessel-Paxton approximation is included in appendix A.

Before we move onto the first topic, we will briefly describe the physical system used for the various tests performed in this report. Bulk aluminium will be used through out (except in cases where a vacancy is introduced to break symmetry). Aluminium was chosen because it is near the top of the periodic table and therefore one does not have to worry about relativistic or semi-core corrections to the pseudopotential. At the same time it still has enough screening from core electrons so the pseudopotential will not be very complicated. The purpose of the test is to demonstrate the efficiency of the code, complications in the pseudopotential add an unnecessary overhead. The exchange-correlation functional used for our tests will be the local density approximation (LDA). This is justified because firstly it was reported by Gaudoin et al.[6, 7] that the LDA approximation is adequate for bulk aluminium calculations,

and secondly the focus of our calculations again is on the efficiency of the code, not on the accuracy of the band structure. The aluminium pseudo-potential is calculated using OPIUM[1] according to the method of Rappe et. al.[17, 18]. This method generally gives a softer pseudopotential than a standard Troullier-Martins method[19]. Two $3s$ and one $3p$ electrons are taken as valence electrons for aluminium and partial-core (also known as non-linear core) or relativistic corrections are not included. The basis set used was double-zeta-polarisation (DZP) numerical pseudo-atomic-orbital (PAO) basis. The basis was generated from SIESTA[2] with energy shift of 100 meV. For the the current study we used just one support function per basis. The optimum lattice constant was found by calculating the ground-state energies for the system (using 4 Al atoms unit cell) with the lattice parameter varying from 3.50–4.40Å, and then finding the minimum by interpolating the results. We found for the above mentioned set-up the optimum lattice parameter for the LDA calculation to be $a_0 = 4.01155$Å. This can be compared to the result of Gaudoin et al. 3.960Å and experimental value of 4.022Å. It was also found that a real-space mesh with 32 grid points in each direction, and a reciprocal space mesh with 25 **k**-points in each direction (generated using Monkhorst-Park[15] method) is sufficient for accuracy up to 10^{-6}Ha with smearing temperature of 0.001Ha ($\approx$ 300K). For unit cells of different sizes, we scaled the grid points accordingly, this means we need 64 real space points and 13 **k** points in each direction for a 32 atoms cell, and 128 real space points and 7 **k** points in each direction for 108 atoms cell. The calculations mentioned in the report will assume these parameters unless otherwise stated.

2 Kerker Preconditioning and Wave-dependent Metric

Pulay mixing[16] has became one of the standard methods implemented in the ab initio Density Functional Theory (DFT)[8, 10] codes to achieve self-consistency in ground-state energy calculations[13, pp. 172–174]. The scheme at beginning of a given self-consistent loop m, takes a set of histories of input densities ρ^i_{in} from self-consistency loops i in the range $i = m - N_{\text{Pulay}}$ to $m-1$ (for a given integer N_{Pulay} and if $m = 1$ then we use an initial guess) and forms an optimised input density for step m

$$\rho^m_{\text{in}} = \sum_{i=m-N_{\text{Pulay}}}^{m-1} \alpha_i \left(\rho^i_{\text{in}} + \lambda R[\rho^i_{\text{in}}]\right) \tag{1}$$

where λ is the mixing parameter; the parameters α_i are given as

$$\alpha_i = \frac{\sum_{j=m-N_{\text{Pulay}}}^{m-1} A^{-1}_{ji}}{\sum_{i,j=m-N_{\text{Pulay}}}^{m-1} A^{-1}_{ji}}, \qquad A_{ji} = \int \mathrm{d}^3\mathbf{r}\, R([\rho^j_{\text{in}}], \mathbf{r}) R([\rho^i_{\text{in}}], \mathbf{r}) \tag{2}$$

and satisfies the constraint $\sum_{i=m-N_{\text{Pulay}}}^{m-1} \alpha_i = 1$; the residual $R([\rho^i_{\text{in}}], \mathbf{r})$ is defined as the difference between the output (calculated from minimising the energy functional dependent on the input density) and input density:

$$R([\rho^i_{\text{in}}], \mathbf{r}) = \rho^i_{\text{out}}([\rho^i_{\text{in}}], \mathbf{r}) - \rho^i_{\text{in}}(\mathbf{r})$$

Note also that by choosing $N_{\text{Pulay}} = 1$ we get back simple linear mixing scheme.

The phenomenon called "charge sloshing"[9]—where a small variation in the input density (in the self-consistency loop) causes a large change in output density—is often observed in calculations for metallic systems. This results in very long repeats of the self-consistency loop if the calculation ever converges and it is one of the main obstacles against fast calculations on metallic systems. The solution to this problem exists from noting that charge sloshing is caused by long-range real-space changes (and hence short reciprocal-space changes) in the density dominating the variation of the Hartree potential. Hence by slightly modifying the mixing procedure and by filtering out the long range changes in density (the residuals) one can reduce the effect of charge sloshing and hence achieve much faster self-consistency convergence. This preconditioning method is named after its inventor Kerker[9], and has already been successfully applied to various ab initio electronic structure codes[11, 12]. An alternative method is by adding weight to the computation of the residual metric (A_{ij} in equation (2)) so that the short ranged variations in density are given more importance in the mixing procedure, and this method is referred to as *Wave-dependent metric* approach.

Working under reciprocal space, the Pulay mixing equation (1) becomes

$$\tilde{\rho}^m_{\text{in}} = \sum_{i=m-N_{\text{Pulay}}}^{m-1} \alpha_i \left(\tilde{\rho}^i_{\text{in}} + \lambda \tilde{R}[\tilde{\rho}^i_{\text{in}}] \right)$$

where $\tilde{\rho}^i_{\text{in}}(\mathbf{q})$ are the Fourier transform of input density histories; and $\tilde{R}([\tilde{\rho}^i_{\text{in}}], \mathbf{q}) = \tilde{\rho}^i_{\text{out}}(\mathbf{q}) - \tilde{\rho}^i_{\text{in}}(\mathbf{q})$. For Kerker preconditioning, we replace the scalar mixing parameter λ with a diagonal matrix in reciprocal space:

$$G = \lambda \frac{q^2}{q^2 + q_0^2}, \qquad q = \|\mathbf{q}\|$$

where q_0 is a user input parameter controlling the meaning of "long ranged" variation in density in the preconditioning procedure: for $q \gg q_0$ (short range in real space), $G \approx \lambda$ so we have normal Pulay mixing, but for $q \ll q_0$ (long range in real space), $G \approx 0$ so the portion of ρ^i_{in} do not take part in the mixing.

While the Kerker preconditioning method effectively adds a $\mathbf{q}$-dependent weight to the mixing parameter, the wave-dependent metric method adds a $\mathbf{q}$-dependent weight on how we compute the Pulay paramaters α_i. We modify the residual metric A_{ij} (given in equation (2)) to become

$$\tilde{A}_{ij} = \sum_{\mathbf{q}} \frac{q^2 + q_1^2}{q^2} \tilde{R}([\tilde{\rho}^i_{\text{in}}], \mathbf{q}) \tilde{R}([\tilde{\rho}^j_{\text{in}}], \mathbf{q}), \qquad q = \|\mathbf{q}\| \tag{3}$$

in the reciprocal space. q_1 is an user input parameter so that for $q \gg q_1$ (short range in real space) the weight for the metric $\to 1$ and hence its contribution (in terms of its inverse) to the Pulay parameters α_i will be as normal; and for $q \ll q_1$ (long range in real space), the weight $\to \infty$ and hence its contribution to the parameters α_i becomes negligible.

Special treatment is needed for the point $\mathbf{q} = 0$, since $\omega(\mathbf{q}) = \frac{q^2+q_1^2}{q^2}$ is undefined at this point. The easiest approach is to define the weight at $\mathbf{q} = 0$ as analytic continuation of the factor, and hence we define $\omega(\mathbf{q} = 0) \equiv \max_{\mathbf{q}} \omega(\mathbf{q})$.

2.1 Implementation

To implement both Kerker preconditioning and wave-dependent metric method into CONQUEST we note that both involves at least two Fourier transforms (to transform into reciprocal space and then back). It is therefore ideal if we can add the Kerker preconditioning and the wave-dependent weight at the same time while we are working inside the reciprocal space. Two additional arrays of the dimensions $N_{\text{Pulay}} \times$ No. of reciprocal space grid points were added, one for storing the preconditioned residuals on real space grid: $\mathfrak{F}^{-1}\left(G\tilde{R}[\rho^i_{\text{in}}]\right)(\mathbf{r})$ and one for storing the *covariant* residuals for the wave-dependent metric on real space grid: $\mathfrak{F}^{-1}\left(\frac{q^2+q_1^2}{q^2}\tilde{R}[\tilde{\rho}^i_{\text{in}}]\right)(\mathbf{r})$. Note that here $\mathfrak{F}^{-1}$ denotes the inverse Fourier transform.

It is possible to save the memory and calculate the preconditioned and the covariant residuals for every Pulay history and then accumulate in the Pulay mixing procedure (see equation (1)). However since CONQUEST is a code optimised for parallel calculation with many processors and Fourier transformation is a very communication intensive procedure, such approach would mean we have to do two Fourier transforms for every of the N_{Pulay} Pulay histories for every history summation loop in equation (1), and this would result a very inefficient code. By introducing two additional arrays to the code we minimise communications required for performing Fourier transforms. Note also that the original histories of the residual are still required because they are still needed in the calculation of the metric A_{ij}—acting as the *contravariant* part of the inner product.

Three subroutines were added to the original `hartree_module` in the CONQUEST source code. Subroutine `kerker` is for self-consistent calculation using Kerker preconditioning only, `wdmetric` is for using wave-dependent metric only and `kerker_and_wdmetric` is for using both.

Pseudocode algorithm 1 gives a rough schematic for subroutine `kerker_and_wdmetric`. The subroutines `kerker` and `wdmetric` are similar with the relevant parts omitted (for `kerker` the parts with wave-dependent metric will be omitted and vice versa for `wdmetric`). One of the subroutines is called depending on user input (whether to use Kerker preconditioning or wave-dependent metric or both) for

Algorithm 1 Subroutine `kerker_and_wdmetric`

Compute q_0^2 and q_1^2 from user input
Allocate temporary arrays $FR\_kerker(:)$ and $FR\_wdmetric(:)$
Fourier transform the newest residual and store in both $FR\_kerker(:)$ and $FR\_wdmetric(:)$
$facmax = 0$
for all reciprocal grid points **q do**
  if $q^2 > 0$ **then**
    Calculate preconditioning factor $fac = \frac{q^2}{q^2+q_0^2}$
    $FR\_kerker = fac * FR\_kerker$
    Calculate the weight for wave-dependent metric $fac2 = \frac{q^2+q_1^2}{q^2}$
    $FR\_wdmetric = fac2 * FR\_wdmetric$
    $facmax = \max(facmax, fac2)$
  end if
end for
Use MPI calls to find the global maximum of $facmax$ across all processors
if $q^2 = 0$ **then**
  $FR\_kerker(i0) = 0$ ▷ $i0$ corresponds the grid index for $\mathbf{q} = 0$
  $FR\_wdmetric(i0) = facmax * FR\_wdmetric(i0)$
end if
Inverse Fourier transform FR_kerker and $FR\_wdmetric$, store in respective arrays
Deallocate FR_kerker and $FR\_wdmetric$

every new residual, and the results from the temporary arrays FR_kerker and $FR\_wdmetric$ (see algorithm 1) are stored in the correct Pulay history slots in the $\mathfrak{F}^{-1}\left(G\tilde{R}[\rho^i_{\text{in}}]\right)(\mathbf{r})$ and $\mathfrak{F}^{-1}\left(\frac{q^2+q_1^2}{q^2}\tilde{R}[\tilde{\rho}^i_{\text{in}}]\right)(\mathbf{r})$ arrays respectively. The preconditioned and covariant residuals are then used in the normal Pulay mixing routine already implemented in CONQUEST, replacing the appropriate Pulay history residuals.

2.2 Test Results

Due to symmetry of the aluminium bulk charge sloshing will not take place. Any updates in density will follow the same symmetry as one cannot distinguish one lattice direction from another, resulting an output density which obeys the same symmetry as input density, which in turn preserves the symmetry of the updated Hartree potential $\hat{V}_{\text{H}}$. Indeed for bulk calculations with either 32 atoms or 108 atoms unit cell even with no Kerker preconditioning or wave-dependent metric the calculation reached self-consistency within 12 Pulay mixing steps (with mixing parameter set at $\lambda = 0.5$).

To break the symmetry a vacancy is introduced to the aluminium bulk by removing one atom at $(0, 0, 0)$. The standard linear mixing method failed to reach self-consistency with mixing parameter λ ranging from 0.1 to 0.8, however adding Kerker Preconditioning significantly improved the self-consistency convergence properties. Figure 1 shows charge sloshing behaviour in aluminium bulk system with 32 atoms unit cell and a defect (vacancy) at $(0, 0, 0)$ while using simple linear mixing ($q_0 = 0$ case). The effect of charge sloshing is demonstrated by the oscillations in the calculated ground-state energy. We can also see that the effect of charge sloshing is damped out with increasing q_0 for Kerker preconditioning.

Figure 2 shows the convergence properties of calculations with different Kerker pre-conditioning parameters. All calculations were done using simple linear mixing with mixing parameter of 0.5. and smearing temperature of 300K ($\approx$ 0.001Ha). Calculations with $q_0 < 0.4$ did not converge (no self-consistent solutions were found due to charge-sloshing). We found that the choice of $q_0 = 1.0{*}2\pi\text{bohr}^{-1} = 1.188\text{Å}^{-1}$ is optimum for aluminium bulk with the chosen vacancy. This is comparable with the results of [11].

Convergence performances on the 32 atoms aluminium bulk with vacancy at (0, 0, 0) using a wave-dependent metric method were also tested. However from the test it appears that Pulay mixing alone (with 5 Pulay history steps) is enough for reaching self-consistency. And switching on/off either wave-dependent metric or Kerker-preconditioning had no noticeable effect on self-consistency. This may be because the 32 Al atom with defect system is still not difficult enough for Pulay mixing to fail. More tests

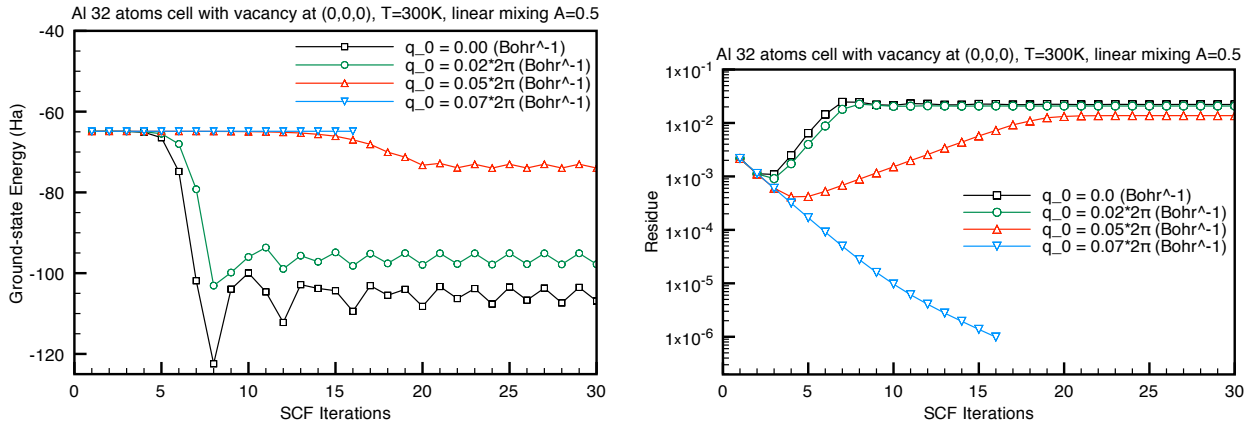

Figure 1: Left: Total ground-state energy vs. number of self-consistency iterations. Right: Residual in electron density vs. number of self-consistency iterations.

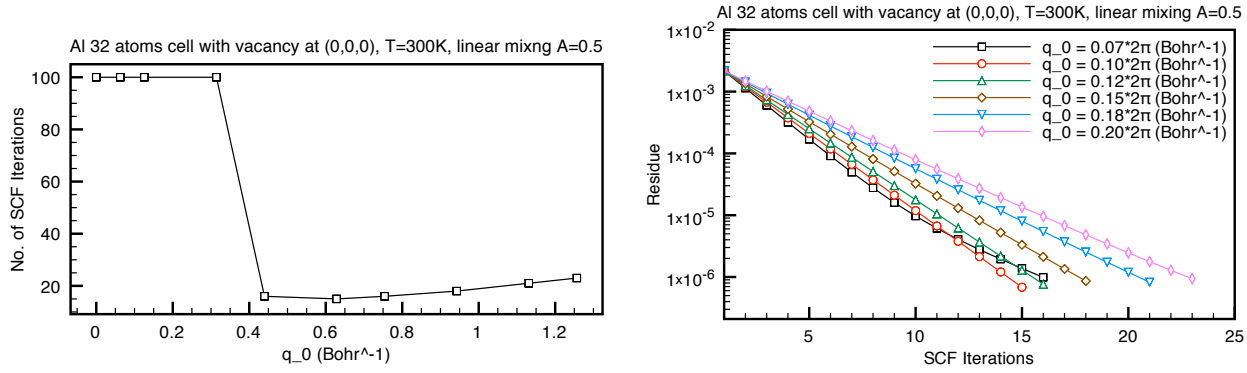

Figure 2: Left: Number of iterations required to reach self-consistency vs. q_0 for Kerker preconditioning. Right: Convergence properties with different q_0 parameters.

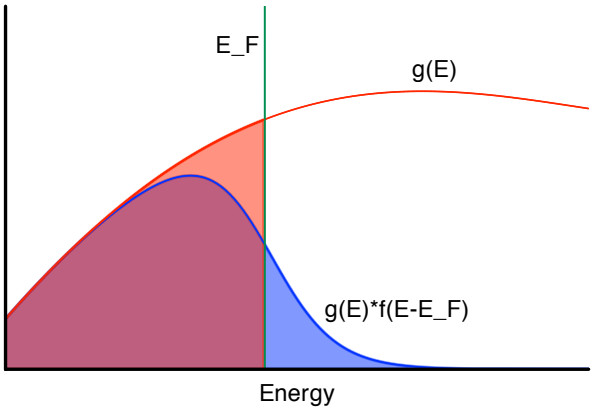

Figure 3: Illustration of the difference between $\int \mathrm{d}E\, g(E) f(E-E_F,T)$ and $\int \mathrm{d}E\, g(E)\theta(E-E_F,T)$. The blue area corresponds to the former and the red area corresponds to the latter. Notice that the patch of blue area above E_F is not the same size as the (uncovered) red area just below E_F.

may be required in future with larger systems with more complex structures may be needed to fully test the potential of Kerker preconditioning and wave-dependent metric on improving the self-consistency process.

3 Methfessel-Paxton Approximation to Step Function

Apart from the effects of charge-sloshing, another main problem associated with calculations involving metallic systems is that one must integrate a discontinuous function over the Brillouin zone due to the partial filling of the band. For numerical integration methods this requires a very fine reciprocal space mesh (**k**-mesh) inside the Brillouin zone. One way to rectify this problem is to approximate the step-function occupation function wit a Fermi-Dirac function of finite temperature T. This makes the integrand a differentiable function everywhere in the Brillouin zone and thus improves **k**-mesh convergence though one has to live with the fact that the result obtained is actually for a slightly different problem of system under finite temperatures. In fact one can show that by simply approximating the step function with a Fermi-Dirac distribution

$$g(E)f(E-E_F,T) \approx g(E)\theta(E-E_F)$$

where

$$f(E-E_F,T) \equiv \frac{1}{e^{(E-E_F)/k_BT}+1}$$

we in general cannot get the integral $\int \mathrm{d}E\, g(E) f(E-E_F,T)$ to equal to $\int \mathrm{d}E\, g(E)\theta(E-E_F,T)$ exactly, this can be seen clearly in figure 3

A more sophisticated approximation to the occupation function can be done using Methfessel and Paxton method[14]. This method approximates the step-function by starting with approximating the delta-function using expansion in a set of orthogonal Hermite polynomials. And then the N-th order approximation $S_N(x)$ is obtained by integrating the delta function. The method guarantees that

$$\int \mathrm{d}E\, g(E) S_N\left(\frac{E-E_F}{k_BT}\right) = \int \mathrm{d}E\, g(E)\theta\left(\frac{E-E_F}{k_BT}\right)$$

for any $g(E)$ that is an polynomial of N-th order or less. The expansion in Hermite polynomials are

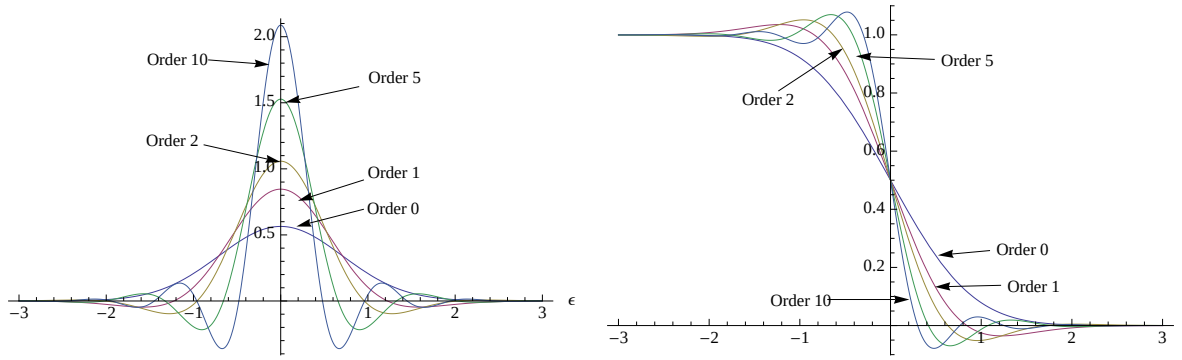

Figure 4: Left: Methfessel-Paxton approximation to delta function. Right: Methfessel approximation to step function.

given as

$$S_0(x) = \frac{1}{2}(1 - \mathrm{erf}(x)) \tag{4}$$

$$S_N(x) = S_0(x) + \sum_{n=1}^{N} A_n H_{2n-1}(x) e^{-x^2} \tag{5}$$

where

$$A_n = \frac{(-1)^n}{n! 4^n \sqrt{\pi}} \tag{6}$$

and Hermite polynomials can be generated by the recurrence relation

$$H_0(x) = 1 \tag{7}$$

$$H_1(x) = 2x \tag{8}$$

$$H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x) \tag{9}$$

Note that at 0-th order, the Methfessel-Paxton method corresponds to a simple Fermi-Dirac like smearing.

The ground-state energy when calculated using Methfessel-Paxton smearing is no-longer variational. Instead the ground-state density minimises the free energy[11]

$$F = E - \sum_n k_B T \cdot \frac{1}{2} A_n H_{2n} \left(\frac{E_n - E_F}{k_B T} \right) e^{-\left(\frac{E_n - E_F}{k_B T}\right)^2} \tag{10}$$

Difference between F and E are small even for large $k_B T$. This makes Methfessel-Paxton method desirable for calculations with metallic systems, one will be able to use a much corser reciprocal grid with a relatively high smearing factor $k_B T$ without introducing too much difference between F and E. If $F - E$ is large then it means the calculated ground-state energy deviates from the true zero-temperature energy—which we are trying to approximate—greatly and therefore the result is unreliable.

However Methfessel-Paxton method also introduces a numerical artefact that Fermi-energy may no-longer be unique. To see this we notice that by expanding the delta function in terms Hermite polynomials, we introduce a number of roots to the function, and this translates into regions of negative occupancies in S_N. This is illustrated in figure 4. As a result, the electron number as a function of Fermi energy:

$$N_e(E_F) = 2 * \sum_n \int \mathrm{d}^3\mathbf{k} w(\mathbf{k}) f\left(\frac{E_n(\mathbf{k}) - E_F}{k_B T} \right) \tag{11}$$

(where $w(\mathbf{k})$ is weight, and $f(x)$ is the occupation function, which is given as $S_N(x)$ for order N Methfessel-Paxton approximation) is no-longer monotonic. This behaviour can be illustrated by a simple toy model where we imaging have a system with the density of states represented as two gaussian

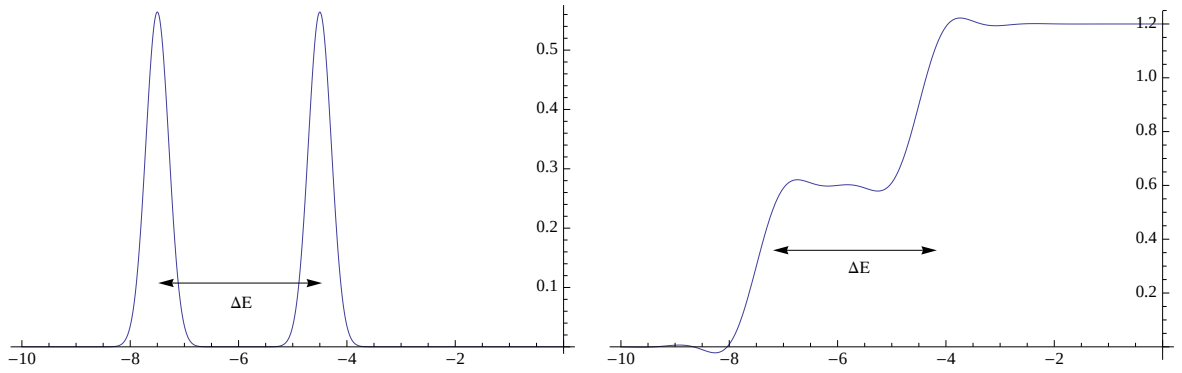

Figure 5: Left: Density of State of a toy model system. Right: The corresponding number of electrons with respect to Fermi Energy

function separated by some energy ΔE, then $N_e(E_f)$ can be calculated easily and are plotted in figure 5. As we can see, there is a possibility that for a given number of electrons there are more than one Fermi energies that would give the correct answer.

While given a lower bound and upper bound in function $N_e(E_F)$ the bisection method always returns a solution E_F that gives the desired electron numbers, it is important that we are consistent on which E_F—if there are more than one—we actually find. Note that the properties of non-unique Fermi energy is not physical, but is an artefact of Methfessel-Paxton approximation. Never the less, to be consistent in our calculations we define the Fermi energy to be always the lowest energy that give the correct electron number.

To make sure we always find the lowest energy solution, we need to find the lower and upper energy bounds very carefully. To get a lower bound, we could always start from the lowest energy. However to be more efficient, this energy is probably too low, and we define a parameter N_0 such that we fill up the lower bands (and assuming occupation of 1.0) until we have $N_e - N_0$ electrons. And we use the highest filled energy as the lower bound. To find the upper bound, we increase the lower band by energy steps of $\delta\epsilon$ until $N_e(E_F)$ is greater than the desired number of electrons. To find $\delta\epsilon$, we have to be careful the that it is not too big and miss one possible solution and at the same time not too small so that calculation is still efficient. We note that the width of the Methfessel-Paxton approximation to delta function is controlled by the gaussian weight e^{-x^2}, where $x = \frac{E_n - E_F}{k_B T}$. Hence we may define the width W of the Methfessel-Paxton approximation to delta function as

$$W = 2\sqrt{-\ln(g)}k_B T \tag{12}$$

where $g > 0$ is a user definable parameter. Then we prove that—see theorem A.1—for a Methfessel-Paxton approximation of order N, there are *exactly* $2N$ roots for $D_N(x)$—Methfessel-Paxton approximation to $\delta(x)$. This implies we can choose the energy step to be

$$\boxed{\delta\epsilon = \frac{W}{\eta 2N} = \frac{\sqrt{-\ln(g)}k_B T}{\eta N}} \tag{13}$$

where η is a user definable parameter controlling finess of the step. If $\eta \geq 1.0$, η should be small enough to only go over one stationary point in S_N at a time—remember $D_N(x)$ is the derivative of $S_N(x)$. And hence if starting from an absolute lower bound, the next upper bound we will find for $N_e(E_F)$ will guaranteed to bracket only the lowest Fermi Energy. We can then use the bisection method to find E_F.

3.1 Test Results

Figure 6 shows the comparison between calculations on a 4 atom unit cell aluminium bulk using Fermi and Methfessel-Paxton smearing methods. We only did non-self-consistency calculations. We can see that Fermi smearing leads to different energy results as we increase smearing factor $k_B T$. We can see that

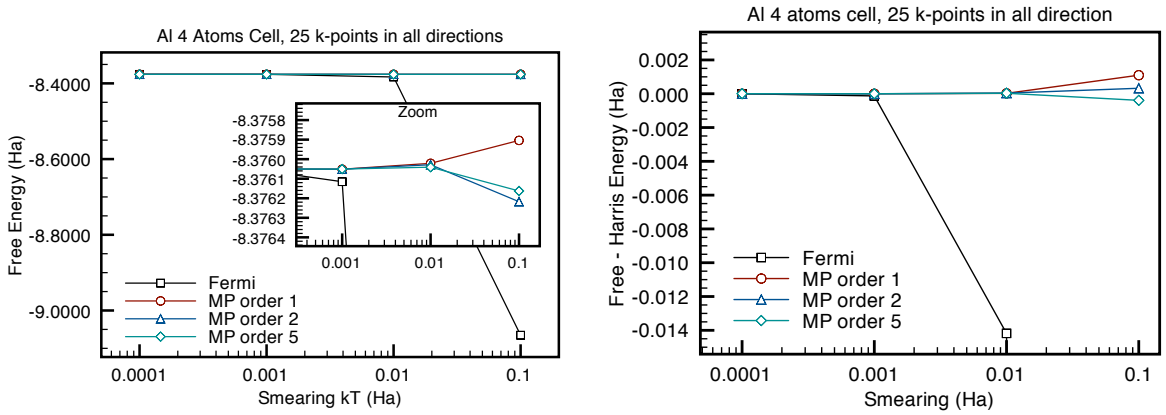

Figure 6: Left: Free energy vs. smearing factor k_BT. The inset shows a zoomed in part of the graph, comparing different values of free energy for different Methfessel-Paxton order of approximations. Right: The difference between free energy and the calculated ground-state energy (called Harris energy here) vs. smearing factor k_BT.

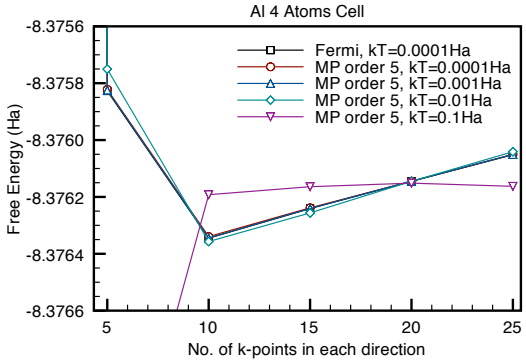

Figure 7: **k**-point convergence for aluminium bulk, 4 atoms cell, with order 5 Methfessel-Paxton smearing.

the difference between Free energy and the calculated ground-state energy (we called it Harris energy) also increased dramatically for Fermi smearing. This is expected as the smearing does correspond to the physical temperature and as k_BT increases we depart from the zero-temperature regime. On the other hand, the smearing in Methfessel-Paxton method is just a parameter used in the approximation. And we can see that increasing k_BT has small effect on the energies calculated using Methfessel-Paxton smearing and results also improve as the order of Hermite polynomials used increases. This means a relatively large smearing factor can be used under Methfessel-Paxton approximation, which allows fewer **k**-points. To be more specific, figure 7 shows that for smearing factor $k_BT = 0.1\text{Ha} = 2.72\text{eV}$, (corresponding to smearing temperature of about 31565.51 K), we reach **k**-point convergence at about 10 Bloch space (**k**) points per each reciprocal lattice direction, while the total energy will be within 10^{-5}Ha of the ground state energies calculated using low smearing and large number of **k**-points. This is compared with the requirement of 25 **k**-points for convergence with a small smearing factor, see figure 8.

4 ScaLAPACK Performance Profiling

The profiling of CONQUEST is done using CrayPAT on HECToR XT5h (compiled with Cray LibSci 10.5.0). The test is based on the calculation for bulk aluminium with 32 atoms unit cell and BLACS processor grid given by 1×4. This processor grid is used because it is the recommended by BLACS for small size matrices and that the calculation failed at diagonalisation for process-grid of 2×2. Table

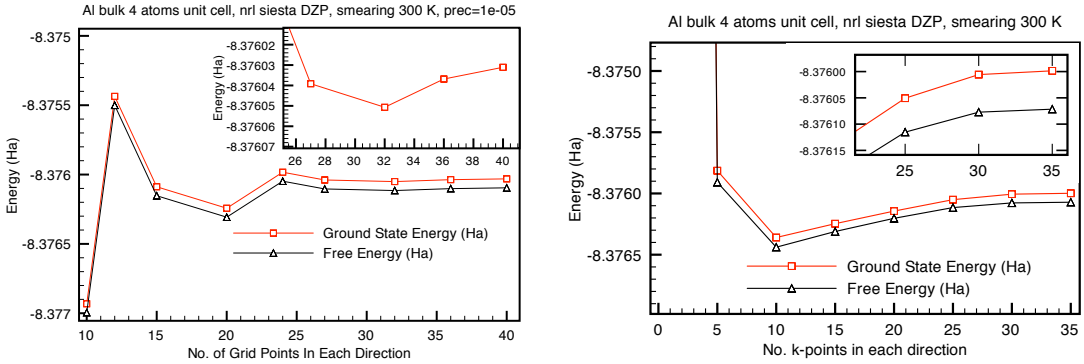

Figure 8: LEFT: Convergence of the ground-state energy and free energy with increased real space grid finess. RIGHT: Convergence of the ground-state energy and free energy with increased number of **k**-points in all directions.

below shows the performance comparisons between different ScaLAPACK block dimensions.

ScaLAPACK Block	CONQUEST Wall Time (s)	CrayPAT Wall Time (s)
13×13	8062.169	7981.555
26×26	7673.137	7626.500
52×52	8197.790	8163.051
104×104	8477.819	8451.286

Results for non-square blocks are omitted because for these cases the calculation again failed at diagonalisation. As one can see the optimum block dimension of the calculation is 26×26, this marks roughly 10% improvement over the CONQUEST default input value which is 104×104. The main bottleneck and the largest load imbalance in the calculations are found (after specifying trace group MPI in CrayPAT) to be the `MPI_recv` calls within the ScaLAPACK subroutine `pzhegvx` used for diagonalisation. Table below lists the largest load imbalances in terms of percentages in the calculation for different block sizes. For calculations with larger block sizes however, the large load imbalances in the `MPI_recv` calls are partially off set by the relatively less (but still significant) number of calls, and we see a shift from `MPI_recv` to `MPI_bcast` being the main bottleneck.

Scalapack Block	Largest Load Imbalance (`MPI_recv`) %	% Time	Largest Load Imbalance (`MPI_Bcast`) %	% Time
13×13	18.7806	29.4	3.0170	16.7
26×26	10.1233	32.6	7.2327	15.2
52×52	38.3164	48.0	13.1732	18.0
104×104	50.6869	35.8	30.7551	27.5

This indicates clearly that the choice of ScaLAPACK block sizes is a determine factor on controlling the efficiency of the ScaLAPACK routines. For the 32 atom bulk aluminium calculation the optimal value seems to be 26×26.

We also compared the efficiency of the ScaLAPACK subroutine `pzhegvx` with the LAPACK equivalent `zhegvx` for a *single* processor case. These calculations were performed on a local Sun workstation 2 × AMD Opteron 2214 (2.2 GHz 2 cores). The table below shows the results of comparison

Total Wall Time (s)	
LAPACK	6590.180
ScaLAPACK	8767.027
Time Spent in Calculating Density Matrix (s)	
LAPACK	5803.934
ScaLAPACK	7973.105

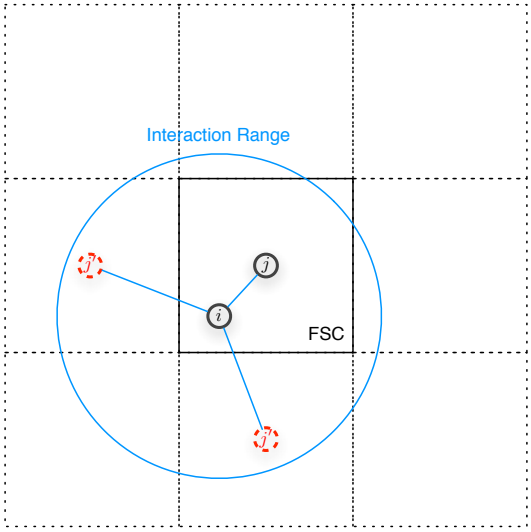

Figure 9: Interaction range of primary set atom i with its neighbour j and the corresponding periodic images j'. All j and j' are in he neighbour list of i.

So it appears that the LAPACK subroutine is inherently more efficient. This may have a lot to do with the different quality in the libraries used. The LAPACK implementation is supplied by the AMD Core Math Library (ACML) as part of the standard set of libraries already on the machine, whereas the ScaLAPACK implementation is a locally compiled version.

5 k-point Parallelisation

The CONQUEST implementation of matrices do not regard any system (bulk or otherwise) to have periodic boundary conditions, instead the code treats any location in real space as it is[5]. The cell from user input is regarded as the *Fundamental Simulation Cell* (FSC), and the FSC is repeated in all lattice directions so that all atoms taking part of interaction with that inside the FSC are taken account of (see figure 9). All quantum-mechanical operators are represented by matrices using support functions (which for the purpose of this report may be regarded as a set of basis functions upon which we have our matrix representation). For details on the meaning of support functions, how these are formed from the (actual) basis—which can either be Pseudo-Atomic Orbitals (PAOs) or B-Spline functions—and how are the quantum mechanical quantities represented by these support functions, please refer to [4, 5].

The CONQUEST method of storing matrices of the form

$$A_{i\alpha,n\beta} = \int \mathrm{d}^3\mathbf{r}\, \phi_{i\alpha}(\mathbf{r}-\mathbf{R}_i)\hat{A}\phi_{n\beta}(\mathbf{r}-\mathbf{R}_j) \tag{14}$$

is illustrated in figure 10. The atoms in the FSC are divided among the processors and the set of FSC atoms responsible by each processor is called a *primary set*. To give CONQUEST more flexibility in data transfer space is divided into many *partitions* and atoms contained in a partitioned space are regarded as to belong to the that particular partition. Matrix elements corresponding to the same partition are stored together in continuous piece of memory, and data transfer are normally done partition by partition. Given different matrices (operators) the interaction ranges between atoms differ, a *halo* atom of a given matrix (range) type is then defined to be any atom that is within the interaction range of the any atoms of the primary set. A set halo atoms corresponds to the union of the set of neighbours (*neighbour-lists*) of the primary set atoms. A *halo partition* of a given interaction range is then any partition that contains at least one halo atom. A *halo* is then defined to be the collection of all halo partitions corresponding to a given range. And finally a *covering set* of a given processor is defined as the minimum orthorhombic cell that contains the largest (one corresponding do the longest interaction range) halo. In $A_{i\alpha,n\beta}$ defined in equation (14), on each node i indexes the primary set atoms responsible by the processor, and n indexes

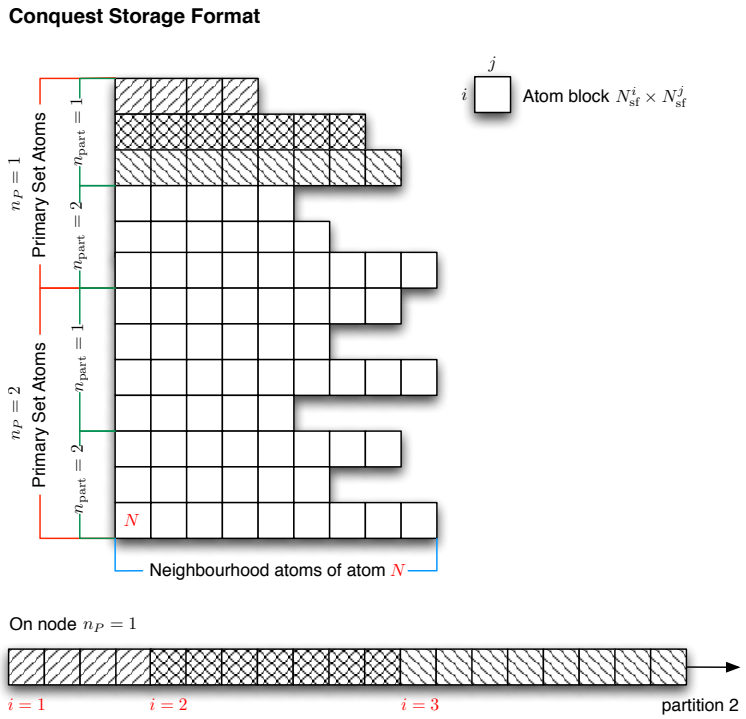

Figure 10: CONQUEST matrix storage format. Each block in the diagram corresponds to (i, j)th atom block with dimension $N^i_{\rm sf} \times N^j_{\rm sf}$, where $N^{i/j}_{\rm sf}$ is the number of support functions for atom i or j respectively. $n_{\rm part}$ is the partition index local to each processing-node.

the atoms in the covering set. Moreover, not all terms with a given n is stored, only those actually in the neighbour-list of each primary set atom i is stored piece-wise in a row-major format (see figure 10).

To update the electron density CONQUEST solves the generalised eigen-value problem

$$\sum_{j\beta} \tilde{H}^{\mathbf{k}}{}_{i\alpha,j\beta} c^{\mathbf{k}j\beta}{}_n = \epsilon_n(\mathbf{k}) \sum_{j\beta} \tilde{S}^{\mathbf{k}}{}_{i\alpha,j\beta} c^{\mathbf{k}j\beta}{}_n \tag{15}$$

where indices i, j index the atoms in the FSC, and α, β index the support functions associated to the corresponding atom. $\tilde{H}^{\mathbf{k}}{}_{i\alpha,j\beta}$ corresponds to the matrix representation of the Hamiltonian operator under periodic boundary conditions:

$$\tilde{H}^{\mathbf{k}}{}_{i\alpha,j\beta} = \int \mathrm{d}^3\mathbf{r}\, \tilde{\phi}^{\mathbf{k}}_{i\alpha}(\mathbf{r} - \mathbf{R}_i) \hat{H} \tilde{\phi}^{\mathbf{k}}_{j\beta}(\mathbf{r} - \mathbf{R}_j) \tag{16}$$

where the Bloch sums of support functions are defined as

$$\tilde{\phi}^{\mathbf{k}}_{i\alpha} = \frac{1}{\sqrt{N}} \sum_{\mathbf{a}} e^{i\mathbf{k}\cdot(\mathbf{R}_i + \mathbf{a})} \phi_{i\alpha}(\mathbf{r} - \mathbf{R}_i)$$

and N is normalisation factor, $\mathbf{a}$ is any linear combination of integer multiples of the lattice constants[1], and $\phi_{i\alpha}(\mathbf{r} - \mathbf{R}_i)$ is the α-th CONQUEST support function corresponding to atom i. All atoms which are periodic images of i share the same support function—i.e. $\phi_{i\alpha}(\mathbf{r} - \mathbf{R}_i) = \phi_{i'\alpha}(\mathbf{r} - \mathbf{R}_{i'})$ if $\mathbf{R}_{i'} = \mathbf{R}_i + \mathbf{a}$. $\tilde{S}^{\mathbf{k}}{}_{i\alpha,j\beta}$ denotes the overlap matrix under periodic boundary conditions

$$\tilde{S}^{\mathbf{k}}{}_{i\alpha,j\beta} = \int \mathrm{d}^3\mathbf{r}\, \tilde{\phi}^{\mathbf{k}}_{i\alpha}(\mathbf{r} - \mathbf{R}_i) \tilde{\phi}^{\mathbf{k}}_{j\beta}(\mathbf{r} - \mathbf{R}_j) \tag{17}$$

[1] In other words, sum over $\mathbf{a}$ corresponds to some over all periodic images of atom at $\mathbf{R}_i$.

And finally ${c^{\mathbf{k}i\alpha}}_n$ are the coefficients of the eigenvectors spanned by the Bloch sums of support functions

$$\psi_n^{\mathbf{k}}(\mathbf{r}) = \sum_{i\alpha} {c^{\mathbf{k}i\alpha}}_n \tilde{\phi}_{i\alpha}^{\mathbf{k}}(\mathbf{r} - \mathbf{R}_i)$$

It can be shown[3] that the hermitian matrices ${\tilde{A}^{\mathbf{k}}}_{i\alpha,j\beta}$ such as ${\tilde{H}^{\mathbf{k}}}_{i\alpha,j\beta}$ and ${\tilde{S}^{\mathbf{k}}}_{i\alpha,j\beta}$ defined in equation (16) and (17) can be calculated from the native CONQUEST matrices using the relationship

$${\tilde{A}^{\mathbf{k}}}_{i\alpha,j\beta} = \sum_{j'} e^{i\mathbf{k}\cdot(\mathbf{R}_{j'} - \mathbf{R}_i)} H_{i\alpha,j'\beta} \tag{18}$$

where j' indexes the atoms in the covering set which is a periodic image of primary set atom j, that is $\mathbf{R}_{j'} = \mathbf{R}_j + \mathbf{a}$ for any $\mathbf{a}$ being a sum of integer multiples of the lattice vectors.

A call to ScaLAPACK subroutine `pzhegvx` is made to obtain the set of eigenvalues (the band structure) $\epsilon_n(\mathbf{k})$ which are used to calculate the Fermi-energy and occupation function (this is discussed briefly in section 3). Once this is done another call to `pzhegvx` is made to get the eigenvectors for each $\mathbf{k}$ point, the new electronic density can than be calculated using formula[3]

$$K^{i\alpha,j'\beta} = \sum_{\mathbf{k}} \sum_n f_n {c^{\mathbf{k}i\alpha}}_n {c^{\mathbf{k}j\beta}}_n e^{i\mathbf{k}\cdot(\mathbf{R}_{j'} - \mathbf{R}_i)} \tag{19}$$

where again, atoms j' corresponds to the periodic images of the primary set atom j and $\mathbf{R}_{j'} = \mathbf{R}_j + \mathbf{a}$. So $K^{i\alpha,j'\beta}$ again is a matrix with j' extending over the entire covering set, and is a matrix that can be stored in native CONQUEST format. Note that there are two calls to `pzhegvx` because we cannot calculate the density without knowing the occupation function first, but on the other hand since the band structure needs eigenvalues calculated for all $\mathbf{k}$ if we are going to make only one call to `pzhegvx` all eigenvectors are then need to be stored. By calling `pzhegvx` twice we can save significant memory by simply accumulating the eigenvectors into the density matrix. It was found[2] also that the call to `pzhegvx` for only calculating eigenvalues is only about 10% the cost of the full call that also calculates the eigenvectors.

The original CONQUEST implementation solves equation (15) one $\mathbf{k}$ point at a time. And the matrices $H_{i\alpha,n\beta}$ and $S_{i\alpha,n\beta}$ are then mapped onto new matrices ${\tilde{H}^{\mathbf{k}}}_{i\alpha,j\beta}$ and ${\tilde{S}^{\mathbf{k}}}_{i\alpha,j\beta}$ distributed across all available processors arranged in a BLACS processor grid according to ScaLAPACK cyclic block format. The calculated eigenvectors from ScaLAPACK for each $\mathbf{k}$ are then transfered from ScaLAPACK data format and accumulated into $K^{i\alpha,j'\beta}$ stored across the processors in CONQUEST format, and the self-consistent calculation carries on from there.

We note that calculations involved for solving eigenvectors for different $\mathbf{k}$ are independent from each other. If we could add a degree of freedom of allowing subgroups of processors working on different $\mathbf{k}$ then it would allow us to choose better optimised parameters for the ScaLAPACK calculations. For matrices of a given size there is an optimal number of processors that should be allowed to work on it, and too many processors means inefficient communications taking over. Hence parallelising calculation in $\mathbf{k}$ would in theory allow one to use more processors more efficiently by having groups of optimal number of processors working for each ScaLAPACK subroutine call. Since for metallic calculations, te number of $\mathbf{k}$ points required are in the order of 1000s, this is a real degree of freedom we can exploit, especially for calculations running on HPC systems such as HECToR.

5.1 Implementation

We introduced a new user definable parameter N_G, denoting the number of processor groups each responsible for different set of $\mathbf{k}$ points. We want to allocate a set of $N_{\mathbf{k}}^G(n_G)$ points to each process group n_G ($n_G = 1, \ldots, N_G$), and then the $N_P^G(n_G)$ processing-nodes inside each group will work on the $N_{\mathbf{k}}^G(n_G)$ points. The cyclic allocation technique is the most efficient in terms of evenly distributing the work-load. So that the process-group indices n_G relates to processing-node indices n_P and $\mathbf{k}$-point indices as

$$n_G = \mathrm{mod}((n_p - 1), N_G) + 1 \tag{20}$$

[2]From private discussion with the main CONQUEST developer Dr. David Bowler.

and

$$n_G = \mathrm{mod}((n_{\mathbf{k}} - 1), N_G) + 1$$

To work out N_P^G we have

$$N_P^G(n_G) = \begin{cases} \mathrm{aint}(N_P/N_G) + 1 & (n_G \leq \mathrm{mod}(N_P, N_G)) \\ \mathrm{aint}(N_P/N_G) & (n_G > \mathrm{mod}(N_P, N_G)) \end{cases}$$

and similarly

$$N_{\mathbf{k}}^G(n_G) = \begin{cases} \mathrm{aint}(N_{\mathbf{k}}/N_G) + 1 & (n_G \leq \mathrm{mod}(N_{\mathbf{k}}, N_G)) \\ \mathrm{aint}(N_{\mathbf{k}}/N_G) & (n_G > \mathrm{mod}(N_{\mathbf{k}}, N_G)) \end{cases}$$

We introduced arrays $pg\_procs(1 : N_G, N_{P\max}^G)$ and $pg\_kpoints(1 : N_G, N_{\mathbf{k}\max}^G)$ to keep track of which processors and **k** points are allocated to which processor groups. These arrays can be worked out following the steps given in algorithm 2. Note however that the algorithm is order N_P^2 (where N_P is total number of processors), because the same algorithm is repeated on every processor. However the calculations are simple and one only has to do it once at the beginning of the calculation; further more no communication is needed between the processors.

Algorithm 2 Calculation of $pg\_procs(:,:)$ and $pg\_kpoints(:,:)$

```
for all Processor groups n_G do
    i = 1
    for all Processors, globally indexed by n_P do
        if n_G == mod((n_P − 1), N_G) + 1 then
            pg_procs(n_G, i) = n_P
        end if
        i = i + 1
    end for
    i = 1
    for all k points, indexed globally by n_k do
        if n_G == mod((n_k − 1), N_G) + 1 then
            pg_kpoints(n_G, i) = n_k
        end if
        i = i + 1
    end for
end for
```

The goal is to define a mapping between the ScaLAPACK block cyclic format distributed over a processor grid consisting of only a subset of processors for a given **k** point and the native CONQUEST format that distributes the rows of the matrices into primary set atoms and partitions across the entire set of processors. In the original CONQUEST implementation, to aid the mapping a reference matrix format is introduced. This is the format for the matrices $\tilde{H}^{\mathbf{k}}{}_{i\alpha,j\beta}$ and $\tilde{S}^{\mathbf{k}}{}_{i\alpha,j\beta}$ which has the same form as if one writes them down on for calculations on paper. The reference format matrices are never physically stored. Also in the original implementation by noticing the fact that the order of atoms in a calculation can be arbitrary, the atomic ordering of the rows of the ScaLAPACK format is *chosen* to be the same as that of the native CONQUEST format. In other words the atomic ordering (index i) of the ScaLAPACK matrices $\tilde{H}^{\mathbf{k}}{}_{i\alpha,j\beta}$ and $\tilde{S}^{\mathbf{k}}{}_{i\alpha,j\beta}$ is chosen to be the same as the ordering of i in matrices $H_{i\alpha,n\beta}$ and $S_{i\alpha,n\beta}$ stored in CONQUEST format. We have kept this requirement for the implementation **k** point parallelisation.

Figure 11 shows the possible mapping of matrices if there are 8 processors in total and the user have chosen 2 processor groups. The BLACS process grid is assumed to be 2×2 and the ScaLAPACK blocks are shown in purple boxes. Due to cyclic allocation, the first group will have processors 1, 3, 5, 7 (note we are indexing the processors from 1) and the second group have processors 2, 4, 6, 8. The ScaLAPACK matrix for each group have identical format, and the matrix elements differ only by the way different **k** point is used to generate $\tilde{H}^{\mathbf{k}}{}_{i\alpha,j\beta}$ or $\tilde{S}^{\mathbf{k}}{}_{i\alpha,j\beta}$ matrices from the CONQUEST matrices $H_{i\alpha,n\beta}$ or $S_{i\alpha,n\beta}$. The coloured squares in the CONQUEST matrix shows the periodic images of atoms all belong

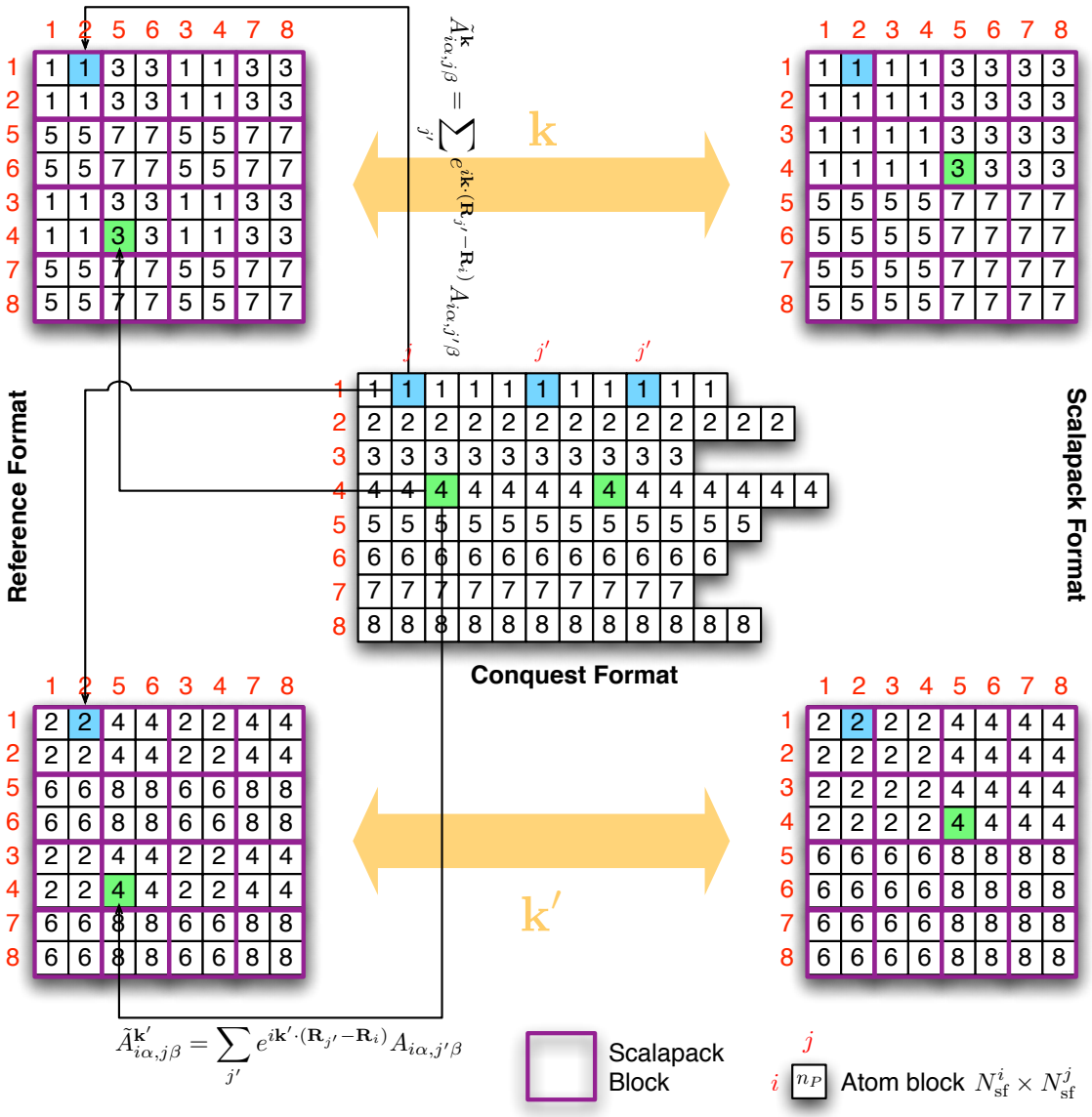

Figure 11: Schematic showing the mapping between native Conquest format matrices and the corresponding square matrices in ScaLAPACK and reference format.

to the neighbourhood of a given primary set atom. And their contributions are summed up according to equation (18) and stored in the corresponding coloured location in the ScaLAPACK matrix. The numbers inside the small boxes identify the processor responsible for the data block.

Since the atomic ordering of rows and columns of the ScaLAPACK is unchanged from the original implementation, subroutines `find_SC_row_atoms`, `find_ref_row_atoms` and `find_SC_col_atoms` of the `ScalapackFormat` module do not need to be modified. However the rest of the module as well as the data transfer modules in module `DiagModule` do need modification, to take into account that now the maps to BLACS processor grid is also dependent on the processor group we are in, and we have to send one copy of the matrices to each of the N_G processor groups.

To obtain the eigenvectors the ScaLAPACK subroutine `pzhegvx` call is made on every processor, with each processor group using a different set of $\tilde{H}^{\mathbf{k}}{}_{i\alpha,j\beta}$ and $\tilde{H}^{\mathbf{k}}{}_{i\alpha,j\beta}$ (corresponding to a different $\mathbf{k}$). Upon return from the ScaLAPACK call each processor group will have a different set of eigenvectors. To build the density matrix we first map the ScaLAPACK format eigenvectors into the native CONQUEST format, then accumulate the products of the eigenvectors one $\mathbf{k}$ point at a time (see equation (19)). This is preferred because we can then utilise the pre-existing code for generating density matrix, and only one storage array for eigenvectors in CONQUEST format is needed on each processor—as contributions from eigenvectors are accumulated into the density matrix. Also performance wise the main benefit of $\mathbf{k}$ point parallelisation comes from the freedom it offers for optimising the ScaLAPACK routine, and building the density matrix is simply a scalar multiplication and accumulation process hence it is perfectly efficient to build the density matrix using the parallel processing format provided by the native CONQUEST approach (please read [5] for details on how matrix operations are handled in CONQUEST).

Some attention needs to be paid on the mapping from the output eigenvectors distributed following the ScaLAPACK block-cyclic format and the BLACS processor grid, to the distribution format for matrices used by CONQUEST. First important fact to note is that the ScaLAPACK subroutine `pzhegvx` returns the eigenvectors in *columns* and in the same order as the corresponding eigenvalues (see Figure 12). On the other hand, in CONQUEST format for each $\mathbf{k}$ each processor is required to be in charge of a set of atoms in FSC, in other words, a set of indices $i\alpha$ for the eigenvectors $c^{\mathbf{k}i\alpha}{}_n$. And unlike the cases with conventional matrices in CONQUEST which are stored in row-major format (i.e. not strictly a matrix in FORTRAN 90 sense), the eigenvectors *are* stored as a two-dimensional array. We require the $c^{\mathbf{k}i\alpha}{}_n$ data associated to each $i\alpha$ index to stay in one continuous piece of memory, this means a column-major storage for the eigenvector. In other words in CONQUEST format the eigenvectors are stored as $c^{\mathbf{k}}{}_n{}^{i\alpha}$. That is, the eigenvectors are stored in *rows*. The order of eigenvalues also changes from the ScaLAPACK format, because the CONQUEST format ordering should be the same as that of the reference format.

The steps taken to generate the new density is illustrated in pseudocode algorithm 3. One has to be careful that we do not step over the limits since it is highly likely that number of $\mathbf{k}$ points in each processor group is different, and hence an `if` statement is required to ensure this.

Algorithm 3 Steps for getting eigenvectors and updating density matrix

1: Build data transfer maps (contained in `ScalapackFormat` module)
2: **for all** $\mathbf{k}$ point indices *pgk* in processor groups **do**
3: Distribute $\mathbf{H}$ and $\mathbf{S}$ from CONQUEST format to ScaLAPACK format $\tilde{\mathbf{H}}^{\mathbf{k}}$ and $\tilde{\mathbf{S}}^{\mathbf{k}}$
4: Call `pzhegvx`, getting eigenvalues (band structure) only
5: **end for**
6: Get Fermi energy and occupation function
7: **for all** $\mathbf{k}$ point indices *pgk* in processor groups **do**
8: Distribute $\mathbf{H}$ and $\mathbf{S}$ from CONQUEST format to ScaLAPACK format $\tilde{\mathbf{H}}^{\mathbf{k}}$ and $\tilde{\mathbf{S}}^{\mathbf{k}}$
9: Call `pzhegvx`
10: **for all** processor groups n_G **do** ▷ We do this one k at a time involving all processors
11: Distribute $c^{\mathbf{k}i\alpha}{}_n$ from group n_G to all processors from ScaLAPACK format to CONQUEST format
12: Accumulate density matrix
13: **end for**
14: **end for**

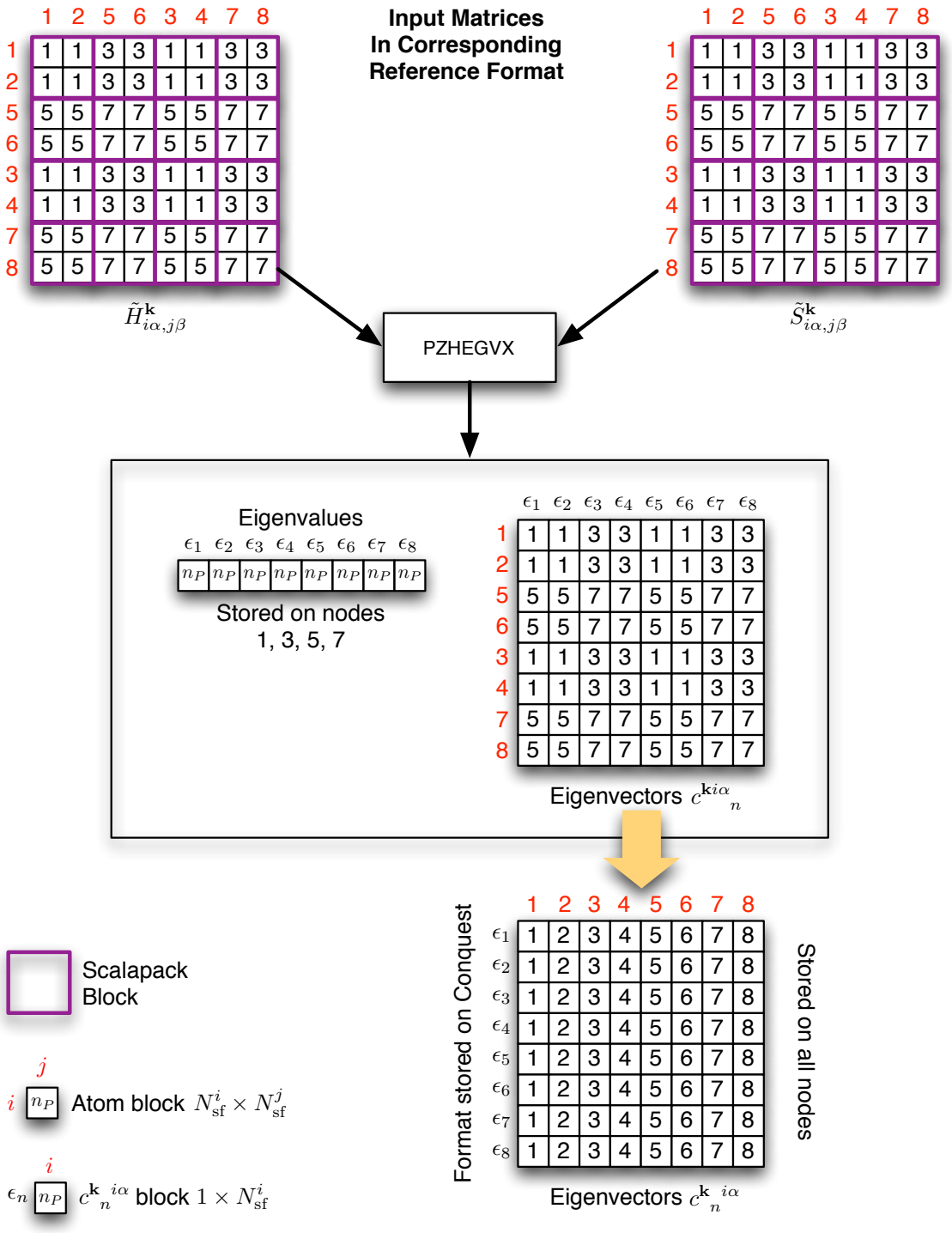

Figure 12: Matrix formats before and after `pzhegvx` call

5.2 Test Results

Preliminary test calculations were done on both HECToR XT5h (with Cray LibSci 10.5.0) and a local Sun Workstation with 2 × AMD Opteron 2214 (2 Core 2.2GHz) (with ACML for LAPACK and local compilation for ScaLAPACK). We used aluminium bulk with 32 atoms unit cell, with a $13 \times 13 \times 13$ **k** point mesh, Fermi-Dirac smearing with temperature of 0.001 Ha. In all cases we did a non-self-consistent calculation on 4 nodes, results are shown in the table below

	N_G	Processor Grid	ScaLAPACK Block	Wall Time
HECToR XT5h	1	1×4	26×26	2318.599
	2	1×2	26×26	1808.664
	4	1×1	26×26	1821.493
Sun Workstation 2 × AMD Opteron 2214	1	1×4	26×26	8794.051
	2	1×2	26×26	6375.609
	4	1×1	26×26	3331.723

As the results clearly shows that **k** point parallelisation has a significant improvement on calculation speed given the same amount of resources compared to the original implementation. This improvement is more apparent for platforms where the ScaLAPACK libraries are not highly optimised.

6 Conclusion

All of the proposed changes to CONQUEST code have been successfully implemented and tested. The Kerker preconditioning method shows clear advantage over linear mixing in allowing calculations to converge under charge-sloshing conditions. However it is not clear from the examples we have tested how much improvement Kerker and wave-dependent metric preconditioning have over Pulay mixing, as the aluminium bulk with defect system also reached self-consistency relatively fast. It is particular difficult to test the effectiveness of wave-dependent metric preconditioning in this case because it has to be used together with Pulay mixing. Therefore further testing may be required to realise the true potential of Kerker preconditioning and wave-dependent metric implementations. We may have to test on a larger system with more complicated defects.

A technical complexity that could lead to the potential problem of non-unique Fermi energy is discovered in the Methfessel-Paxton method for approximating the step function. This is an intrinsic problem originating from the form of the Hermite polynomials. The standard search methods still always find a Fermi energy, but in the rare case of the existence of more than one possible solution, the method can only pick a random one. This will cause problems later on in the calculations especially if one wants to calculate forces. We have developed a search method that ensures always the lowest Fermi energy state is found. And the implementation in CONQUEST is tested to be working as expected, with the Methfessel-Paxton approximation allowing much higher smearing temperatures while giving more accurate ground-state energies than Fermi-Dirac smearing. As demonstrated this allows one to reduce the number of **k** points required for a calculation significantly and hence reduces the computational cost.

The bottleneck of the calculations was found to be the diagonalisation process, as expected. And it seems the main bottleneck within the diagonalisation process comes from the imbalances in MPI communications initiated by ScaLAPACK. Changing the ScaLAPACK block sizes will give a significant change in the performance of the code, the smaller the block sizes the less load imbalances but at the same time more communications. Further study is required for testing a wide range of system sizes and ScaLAPACK parameters which will allow us to develop a better automatic parameterisation scheme for CONQUEST, and make it more user friendly. There is an unsolved issue on why the calculation with 2×2 processor grid and non-square block sizes fails. There may yet be a bug in the code waiting to be resolved, and more work needs to be done to solve this issue.

The modification of CONQUEST for **k** point parallelisation has been successful. There is currently a limiting requirement that each processor group must hold the same number of processors and no-redundant processors are allowed. This means N_G must be chosen to be a factor of the total number of processors. We have shown that by dividing processors into subgroups each working on a **k** point offers more flexibility and in most cases improves on the efficiency of the code. This is especially true if running CONQUEST on a machine that do not have a highly optimised linear algebra library. One

limitation of the current implication is that **k** point parallelisation only applies to the processes involved in diagonalisation, and for other processes the matrices are still shared between all available processors. This prevents one from using more processors than allowed on calculation with more **k** points than number of atoms. There are only a certain number of processors allowed in a given calculation because no processor is allowed to have zero atoms, but atoms are distributed to all processors. For example it is not possible to calculate the 32 atom cell bulk aluminium system with 2 processor-groups each having a 1×4 processor grid, because this requires 8 processors, but for 32 atoms case some processors will not be allocated with atoms. More tests are needed to show the potential **k** point parallelisation have on much larger calculations on HECToR.

The implementations in CONQUEST will be submitted to the code repository after further testing and will be available in the future (beta) release of the code obtainable from `http://hamlin.phys.ucl.ac.uk/NewCQWeb/bin/view`.

7 Acknowledgement

This project was funded under the HECToR Distributed Computational Science and Engineering (CSE) Service operated by NAG Ltd. HECToR—A Research Councils UK High End Computing Service—is the UK's national supercomputing service, managed by EPSRC on behalf of the participating Research Councils. Its mission is to support capability science and engineering in UK academia. The HECToR supercomputers are managed by UoE HPCx Ltd and the CSE Support Service is provided by NAG Ltd. `http://www.hector.ac.uk`

A Mathematical Result Used For Implementation of Methfessel-Paxton Approximation

Theorem A.1 *If*

$$D_N(x) = \sum_{n=0}^{N} A_n H_{2n}(x) e^{-x^2}, \qquad An \equiv \frac{(-1)^n}{n! 4^n \sqrt{\pi}} \tag{21}$$

and $H_n(x)$ are Hermite polynomials of order n, then $D_N(x)$ has exactly $2N$ real and distinct roots.

PROOF We first show that $AnH_{2n}(x=0) > 0, \forall n$. Using the definition of Hermite polynomials we have

$$H_n(x=0) = \begin{cases} 0 & n \text{ is odd} \\ \frac{(-1)^{n/2} n!}{(n/2)!} & n \text{ is even} \end{cases}$$

Hence $AnH_{2n}(x=0) = \frac{(-1)^n}{n! 4^n \sqrt{\pi}} \cdot \frac{(-1)^{n/2} n!}{(n/2)!} = \frac{(2n)!}{(n!)^2 4^n \sqrt{\pi}} > 0$.

Next we show $D_N(x)$ is even. This is trivial, since $H_{2n}(x)$ and e^{-x^2} are even and sum of even functions is even.

By definition of $D_N(x)$ it is a polynomial of degree $2N$. Suppose $D_N(x)$ has $2k$ real and distinct roots, by fundamental theorem of algebra $k \leq N$. By fundamental theorem of algebra again any polynomial could be uniquely defined by its full set of roots up to a constant. In our case let us define a degree $2(k+1)$ polynomial $S(x)$ which shares $2k$ roots with $D_N(x)$, but with one extra root at $x = 0$.

$$S(x) = \alpha(x - x_1)(x + x_1) \cdots (x - x_k)(x + x_k)x^2$$

where $x_1, \ldots, x_k$ are roots of $D_N(x)$, and the scalar factor $\alpha = \pm 1$ is defined so that $S(x) \geq 0$ for $x \in [-x_1, x_1]$. Note that since functions change sign at roots, $D_N(x) > 0$ for $x \in [-x_1, x_1]$, if we define $\pm x_1$ to be the closest pair of roots of $D_N(x)$ near $x = 0$.

Suppose (for proof by contradiction) $k < N$. It is clear from construction that $D_N(x)S(x) \geq 0$ for $\forall x \in (-\infty, +\infty)$ and also $e^{-x^2} > 0, \forall x$, hence we have (note that $S(x) \neq 0$)

$$\int \mathrm{d}x \, D_N(x) S(x) e^{-x^2} > 0$$

On the other hand $D_N(x)$ is the $(2N+1)$-th order expansion of $\delta(x)$. Hence for any polynomial of degree $n, n \leq N+1$

$$\int \mathrm{d}x\, P_n(x) D_N(x) = \int \mathrm{d}x\, P_n(x)\delta(x) = P_n(x=0)$$

As $k < N$, which implies $2(k+1) < 2(N+1)$, we have

$$\int \mathrm{d}x\, S(x) e^{-x^2} D_N(x) = S(x=0)e^0 = 0$$

We hence have a contradiction. ∎

References

[1] http://opium.sourceforge.net/.

[2] http://www.icmab.es/siesta/.

[3] D. R. Bowler. Implementation of diagonalisation within conquest. Technical report, University College London, Oct. 2008.

[4] D. R. Bowler, I. J. Bush, and M. J. Gillan. Practical methods for ab initio calculations on thousands of atoms. *Int. J. Quantum Chem.*, 77(5):831–842, 2000.

[5] D. R. Bowler, T. Miyazaki, and M. J. Gillan. Parallel sparse matrix multiplication for linear scaling electronic structure calculations. *Comput. Phys. Commun.*, 137(2):255–273, June 2001.

[6] R. Gaudoin and W. M. C. Foulkes. Ab initio calculations of bulk moduli and comparison with experiment. *Phys. Rev. B*, 66(5):052104, Aug 2002.

[7] R. Gaudoin, W. M. C. Foulkes, and G. Rajagopal. Ab initio calculations of the cohesive energy and the bulk modulus of aluminium. *J. Phys.: Condens. Matter*, 14(38):8787–8793, 2002.

[8] P. Hohenberg and W. Kohn. Inhomogeneous electron gas. *Phys. Rev.*, 136(3B):B864, 1964.

[9] G. P. Kerker. Efficient iteration scheme for self-consistent pseudopotential calculations. *Phys. Rev. B*, 23(6):3082–3084, Mar 1981.

[10] W. Kohn and L. J. Sham. Self-consistent equations including exchange and correlation effects. *Phys. Rev.*, 140(4A):A1133, 1965.

[11] G. Kresse and J. Furthmüller. Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set. *Computational Materials Science*, 6(1):15–50, July 1996.

[12] G. Kresse and J. Furthmüller. Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Phys. Rev. B*, 54(16):11169–11186, Oct 1996.

[13] R. M. Martin. *Electronic Structure: Basic Theory and Practical Methods.* Cambridge University Press, 2004.

[14] M. Methfessel and A. T. Paxton. High-precision sampling for brillouin-zone integration in metals. *Phys. Rev. B*, 40(6):3616–3621, Aug 1989.

[15] H. J. Monkhorst and J. D. Pack. Special points for brillouin-zone integrations. *Phys. Rev. B*, 13(12):5188–5192, Jun 1976.

[16] P. Pulay. Convergence acceleration of iterative sequences. the case of scf iteration. *Chem. Phys. Lett.*, 73(2):393 – 398, 1980.

[17] A. M. Rappe, K. M. Rabe, E. Kaxiras, and J. D. Joannopoulos. Optimized pseudopotentials. *Phys. Rev. B*, 41(2):1227–1230, Jan 1990.

[18] A. M. Rappe, K. M. Rabe, E. Kaxiras, and J. D. Joannopoulos. Erratum: Optimized pseudopotentials [phys. rev. b 41, 1227 (1990)]. *Phys. Rev. B*, 44(23):13175–13176, Dec 1991.

[19] N. Troullier and J. L. Martins. Efficient pseudopotentials for plane-wave calculations. *Phys. Rev. B*, 43(3):1993–2006, Jan 1991.