Solving large nonlinear generalized eigenvalue problems from Density Functional Theory calculations in parallel

Claus Bendtsen [a] Ole H. Nielsen [b] and Lars B. Hansen [b]

[a] *UNI•C, DTU Bldg. 304, DK-2800 Lyngby, Denmark*

[b] *Dept. of Physics, DTU Bldg. 307, DK-2800 Lyngby, Denmark*

Abstract

The quantum mechanical ground state of electrons is described by Density Functional Theory, which leads to large minimization problems. An efficient minimization method uses a selfconsistent field (SCF) solution of large eigenvalue problems. The iterative Davidson algorithm is often used, and we propose a new algorithm of this kind which is well suited for the SCF method, since the accuracy of the eigensolution is gradually improved along with the outer SCF-iterations. Best efficiency is obtained for small-block-size iterations, and the algorithm is highly memory efficient. The implementation works well on both serial and parallel computers, and good scalability of the algorithm is obtained.

1 Introduction

Within recent years it has become possible to perform quantum mechanical calculations from first principles using the Density Functional Theory (DFT) of Hohenberg, Kohn and Sham (see e.g. [17]). Realistic calculations of the electrons' ground-state can be carried out for large systems consisting of tens to hundreds of atoms owing to improvements in physical methods, faster computers, and more efficient algorithms. In this way modeling of a vast number of physical and chemical properties can be carried out, often taking large computer systems to their limits.

The Density Functional Theory specifies that the ground-state total energy E_0 can be obtained as the global minimum of the energy functional $E[\rho]$,

$$E[\rho] = T[\rho] + V_{\mathrm{ee}}[\rho] + \int \rho(\mathbf{r}) v_{\mathrm{ext}}(\mathbf{r}) d\mathbf{r} + \gamma_{\mathrm{Ewald}} \tag{1}$$

Preprint submitted to Elsevier Preprint *28 June 1999*

Here the position vector in space is denoted as $\mathbf{r}$, $\rho(\mathbf{r})$ (or ρ for short) is the charge density of electrons satisfying the constraint $\int \rho(\mathbf{r})d\mathbf{r} = N$, where N is the integral number of electrons in the system. The $v_{\text{ext}}(\mathbf{r})$ is the "external" potential function acting on the electrons and originating from the atomic nuclei at fixed positions in the system. The kinetic energy $T[\rho]$ as well as the electron-electron interaction energy $V_{\text{ee}}[\rho]$ are functionals of the density ρ, only. Finally, an electrostatic repulsion energy between the nuclei, denoted as γ_{Ewald}, is added. The temperature is taken to be at the absolute zero so that entropy terms are left out in the equations below.

Kohn and Sham[17] proved that the charge density $\rho(\mathbf{r})$ can be represented by a system of N *non-interacting* electrons, whose complex-valued quantum-mechanical wavefunctions $\psi_i(\mathbf{r})$ for $i = 1, \ldots, N$ yield the charge density as

$$\rho(\mathbf{r}) = \sum_{i=1}^{N} |\psi_i(\mathbf{r})|^2 \tag{2}$$

(for simplicity, we only consider integral occupation numbers of 0 or 1) and the non-interacting kinetic energy functional T_s as

$$T_s[\rho] = -\tfrac{1}{2}\sum_{i=1}^{N} \int \bar{\psi}_i(\mathbf{r})\nabla^2\psi_i(\mathbf{r})d\mathbf{r} \tag{3}$$

($\bar{\psi}_i(\mathbf{r})$ denotes the complex conjugate of $\psi_i(\mathbf{r})$). The energy functional Eq. 1 can now be rewritten as

$$E[\rho] = T_s[\rho] + E_{\text{Hartree}}[\rho] + E_{\text{xc}}[\rho] + \int \rho(\mathbf{r})v_{\text{ext}}(\mathbf{r})d\mathbf{r} + \gamma_{\text{Ewald}} \tag{4}$$

where the $E_{\text{Hartree}}[\rho]$ is the classical electrostatic energy of the electrons, and $E_{\text{xc}}[\rho]$ is an (unknown) *exchange-correlation energy* functional.

The wavefunctions $\psi_i(\mathbf{r})$ are the $i = 1, \ldots, N$ lowest eigensolutions of the *Kohn-Sham equation*

$$\hat{H}\psi_i(\mathbf{r}) \equiv [-\tfrac{1}{2}\nabla^2 + v_{\text{eff}}(\mathbf{r})]\psi_i(\mathbf{r}) = \epsilon_i\psi_i(\mathbf{r}) \tag{5}$$

where $\hat{H}$ is known as the Hamiltonian operator. The *effective potential* $v_{\text{eff}}(\mathbf{r})$ is the sum of the external, the *Hartree* (electrostatic), and the *exchange-correlation* potentials:

$$v_{\text{eff}}(\mathbf{r}) = v_{\text{ext}}(\mathbf{r}) + \frac{\delta E_{\text{Hartree}}[\rho]}{\delta\rho(\mathbf{r})} + \frac{\delta E_{\text{xc}}[\rho]}{\delta\rho(\mathbf{r})} = v_{\text{ext}}(\mathbf{r}) + \int \frac{\rho(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}d\mathbf{r}' + v_{\text{xc}}(\mathbf{r}) \tag{6}$$

It is seen from Eq. 6 that the potential $v_{\text{eff}}(\mathbf{r})$ depends on the charge density $\rho(\mathbf{r})$, which itself is given by Eq. 2 as a sum of squared wavefunctions that are determined as eigensolutions of Eq. 5. Hence a *self-consistent* solution of these equations must be found starting from an initial guess of $\rho(\mathbf{r})$, from which the $v_{\text{eff}}(\mathbf{r})$ is constructed; solving the Kohn-Sham eigenvalue equation yields through Eq. 2 a renewed charge density. A minimization algorithm for $E[\rho]$ must be used to obtain the ground-state total energy E_0 from such iterations.

An additional complication in some of the current approaches is the use of *ultrasoft* pseudopotentials[26] that puts a lower requirement on the basis set size, but at the price of a more complicated and computationally more demanding Hamiltonian $\hat{H}$. This leads to replacing the Kohn-Sham equation Eq. 5 by a *generalized eigenvalue equation*,

$$\hat{H}\psi_i(\mathbf{r}) = \epsilon_i \hat{S}\psi_i(\mathbf{r}) \tag{7}$$

where the $\hat{S}$ denotes the overlap projection operator[26].

In most numerical calculations the wavefunctions $\psi_i(\mathbf{r})$ are expanded on a suitable, finite set of *basis functions*. The present work employs the widely used *pseudopotential approximation* together with *plane-wave* (or Fourier expansion) basis sets, and therefore full, complex Hermitian matrices of sizes $10^3 - 10^5$ are usually encountered, whereas the number of electrons N may typically be two orders of magnitude smaller, $10^1 - 10^3$. However, the algorithms described below remain valid for very general classes of problems using other basis function sets.

The discretized problem originating from Eq. 7, which essentially has to be solved for the smallest N eigenvalues, is the non-linear matrix eigenvalue problem,

$$\mathbf{H}(\underline{\rho})\underline{\psi_i} = \epsilon_i \mathbf{S}\underline{\psi_i}, \quad i = 1, \ldots, N \tag{8}$$

$$\underline{\rho} = \sum_{i=1}^{N} \underline{\psi_i}^H \mathbf{S}\underline{\psi_i}, \tag{9}$$

where the overlap matrix $\mathbf{S}$ is a positive definite matrix when *ultrasoft* pseudopotentials[26] are used (and the identity matrix otherwise). Given an initial charge density vector $\underline{\rho}$ (reasonably good initial values for ρ can usually be constructed using the densities of the constituent free atoms), Eq. 8 can be solved for the wave functions and the resulting charge density can subsequently be computed using Eq. 9. A new input charge density can finally be formed, and the cycle can be iterated until convergence. The overall procedure is commonly termed Self Consistent Field (SCF).

Alternative approaches to the SCF method exist, notably direct minimization of the energy as a functional of the wavefunctions $\psi_i(\mathbf{r})$ instead of the charge density ρ, see for example [25]. It has been shown[9] that this method has about the same efficiency as the traditional SCF method.

2 Solving the Non-linear Eigenvalue Problem

Using a quasi newton method for the iterations of Eq. 8 and Eq. 9 generally results in convergence within a small number of iterations. The iterations are usually carried out (see e.g. [9]) as proposed by Pulay[18], but a more robust and slightly more efficient approach is the quasi newton method described in Appendix A. Ongoing investigations of the limited memory BFGS method[11] indicates that is has a similar efficiency but may be less robust.

Solving the eigenvalue problem Eq. 8 is computationally by far the most expensive part of the algorithm because of the cost of the $\mathbf{H}(\underline{\rho})\underline{\psi_i}$ product. The method of Davidson[6] has been especially successful for electronic structure computations, and has been improved by iterating on several eigenpairs simultaneously (see e.g. [7]) and using better preconditioners[13,15,20]. Lately, better theoretical understanding of the methods has contributed to further generalizations such as the Jacobi-Davidson method[23,22] and restarting techniques[3,5,24] along with other iterative methods for large eigenvalue problems based on Rayleigh quotient iterations[8], inverse iteration[10] or the Lanczos method[14,4]. A currently advocated method in this field is the residual minimization method, RMM-DIIS[9] but as this method essentially just computes eigenpairs closest to the initial ones and therefore easily can result in eigenpairs being missed, we do not recommend this approach.

Unfortunately, a major drawback—in the present context—of the above methods is that they all (except for RMM-DIIS) focus on solving the eigenvalue problem to a fairly high accuracy. However, this turns out not to be necessary during every step of the selfconsistency cycle; in fact, our calculations have shown that solving the eigenvalue problem to a high accuracy generally does *not* decrease the number of overall steps in the quasi newton/fixpoint iteration noticeably, and in certain cases even increases the number of steps.

Thus we seek an eigenproblem solver which, given an initial estimate of N eigenpairs, is able to improve this estimate (i.e. perform a relatively small number of iterations) for *each* of the eigenpairs. In addition, an important aspect of large-scale applications is that the memory requirement must be minimized. In order to achieve these objectives, we investigated a number of algorithms for improving eigenpairs so that residuals are improved only until a certain, adjustable limit.

Using deflation techniques[21,16] we experienced convergence problems when deflating eigenpairs which were not well converged, and therefore the overall eigenvalue problem had to be solved to a much higher accuracy (and at a higher computational cost) than actually required. Similar experiences have been reported in [20], where the convergence tolerance was loosened as more and more eigenpairs converged.

We instead propose the following algorithm for solving the generalized eigenvalue problem Eq. 8, which essentially is a blocked Davidson-like algorithm using a non-orthogonal basis, but where a maximum number of expansions on each eigenpair can be imposed.

(1) Given an estimate $\underline{\psi}_1, \ldots, \underline{\psi}_N$ for the N eigenvectors, initialize the subspace basis $\mathbf{B} \equiv [\underline{b}_1, \ldots, \underline{b}_N] = [\underline{\psi}_1, \ldots, \underline{\psi}_N]$ and solve the small, projected subspace eigenvalue problem of dimension N:

$$\mathbf{B}^H \mathbf{H} \mathbf{B} \underline{\phi}_i = \lambda_i \mathbf{B}^H \mathbf{S} \mathbf{B} \underline{\phi}_i, \quad i = 1, \ldots, N, \text{ where } \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_N.$$

Select an iteration block-size n_b and a maximum number of basis vectors $n_{\max}$ and a maximum number of iterations on each eigenpair $k_{\max}$.

(2) Initialize the set $\mathcal{K} = \emptyset$.

(3) Loop over all eigenpairs, $i = 1, \ldots, N$
   (a) If the i'th eigenpair $\lambda_i, \underline{\phi}_i$ has less than $k_{\max}$ iterations then compute the corresponding residual $\underline{r}_i = \mathbf{H}\mathbf{B}\underline{\phi}_i - \lambda_i \mathbf{S}\mathbf{B}\underline{\phi}_i$. If $\|\underline{r}_i\|$ is larger than some tolerance add i to $\mathcal{K}$.
   (b) If $\mathcal{K}$ has n_b elements skip to 4a

(4) Do an iteration on eigenpairs given by indexes in $\mathcal{K}$:
   (a) "Precondition" the residuals corresponding to unconverged eigenpairs, i.e. solve approximately for $\forall i \in \mathcal{K} : (\mathbf{H} - \lambda_i \mathbf{S}) \underline{x}_i = -\underline{r}_i$. The usual physical approximation in [25] is used here.
   (b) Extend the basis, $\forall i \in \mathcal{K} : \mathbf{B} = [\mathbf{B}, \underline{x}_i / \|\underline{x}_i\|]$.
   (c) Solve the updated subspace eigenproblem for the N smallest eigenvalues,

$$\mathbf{B}^H \mathbf{H} \mathbf{B} \underline{\phi}_i = \lambda_i \mathbf{B}^H \mathbf{S} \mathbf{B} \underline{\phi}_i, \quad i = 1, \ldots, N, \text{ where } \lambda_1 \leq \ldots \leq \lambda_N.$$

(5) If the number of basis vectors in $\mathbf{B}$ is too large (i.e. larger than $n_{\max} - n_b$) then collapse the subspace (restart):

$$\mathbf{B} = \mathbf{B}[\underline{\phi}_1, \ldots, \underline{\phi}_N].$$

(6) Repeat from 2 until $\mathcal{K} = \emptyset$.

(7) Update $[\underline{\psi}_1, \ldots, \underline{\psi}_N] = \mathbf{B}[\underline{\phi}_1, \ldots, \underline{\phi}_N]$

The reason for choosing a non-orthogonal basis, $\mathbf{B}$ (and thus obtaining a generalized eigenvalue problem in the projected space, step 1 and 4c) is, besides from reducing the computational cost compared to using an $\mathbf{S}$-orthonormal basis, that it reduces the number of synchronization points in the parallel implementation (even when compared to an orthogonalization algorithm implemented with delayed summation as in e.g. [20]). The drawback is that projected eigenvalue problem, step 1 and 4c could be poorly conditioned and thus slow down the convergence[16, Sec. 11.10]; this has however not been observed in practice.

Attempts were made to reduce the subspace dimension locally by disregarding eigendirections of eigenvalues far from the ones which were currently iterated on. An algorithm was devised which worked well in practice, provided that not too many expansions were carried out on each eigenpair (and thereby changing the corresponding eigenvector drastically), but it involved complicated heuristics and the benefit from a slightly lower overhead in the linear algebra did not seem worthwhile. Similar conclusions were made for an attempt on an algorithm which "locked" treated eigenvectors but did not remove them from the set of basis vectors. The problem for the latter algorithm was the heuristics needed for treating eigenvalues with higher multiplicity (which often occur due to symmetries in the systems).

Since the time consuming part of the eigenvalue solver is computing the products of $\mathbf{H}$ and $\mathbf{S}$ by vectors in the steps 1, 3a and 4c these vectors are stored and reused whenever possible. Thus the memory requirement of the algorithm is $\mathcal{O}(3nn_{\max})$, where n is the vector length given by the dimension of Eq. 8.

Typical problem sizes might be $N \approx 10^1 - 10^3$, and as it will be shown in Sec. 4 the parameters should typically be chosen so $k_{\max} \approx 1$ to 2, $n_b \approx 10$ and $n_{\max} \approx N + 5 * n_b$, so the memory requirement is very close to the absolute minimum, which is the $3N$ vectors $\underline{\psi_i}$, $\mathbf{H}\underline{\psi_i}$ and $\mathbf{S}\underline{\psi_i}$ for $i = 1, ..., N$. Lower memory consumption could be achieved by eliminitating the storage of $\mathbf{H}\underline{\psi_i}$ and $\mathbf{S}\underline{\psi_i}$ and recomputing these products as required in steps 3a and 4c, however with the tradeoff of requiring about twice as many multiplications of $\mathbf{H}$ and $\mathbf{S}$ by vectors.

Most floating-point operations stem form the very complicated products of $\mathbf{H}$ and $\mathbf{S}$ by vectors (the number of such operations should be minimized by any algorithm). In addition, the present algorithm contains significant amounts of linear algebra (approximately $8nN$ flops for each $\mathbf{H}$ product) which however can be carried out efficiently using BLAS level 3 operations.

3 Parallel Implementation

Since the $\mathbf{H}$ times vector products parallelize particularly poorly (they involve 3D FFTs of fairly small size in each dimension), the parallelization for a distributed memory computer is not viable solely across the plane waves (vectors of length n). By using a block size n_b equal to the number of available processors, it is however possible to perform n_b $\mathbf{H}$-products in parallel – each on a different processor. This approach is used in the implementation for the initial calculation of the $\mathbf{H}$ and $\mathbf{S}$ products in step 1 and for step 4a, where the n_b residual vectors each are gathered on different processors before preconditioning them. After the preconditioning, $\mathbf{H}$ and $\mathbf{S}$ products are formed for the preconditioned vector, and the results are finally redistributed across the processors. This requirement on the block size n_b can be lifted on computers that perform small-size parallel 3D FFTs with high efficiency.

For the remaining part of the code the parallelization is performed across the plane waves (i.e. each basis-vector is partitioned among the processors). This requires a fairly minimal amount of communication (essentially only a small number of reduction operations).

The parallel implementation was done using MPI[12], and some typical results indicating the scaling for runs carried at on the IBM SP [1] are given in Table 1 [2].

Table 1
Parallel runs of the algorithm for a Pt problem.

# of processors	execution time	speed-up
1	9544 (10932)	-
2	4208 (4671)	2.3
4	1860 (2192)	5.1
6	1152 (1421)	8.3
8	901 (1197)	10.6
12	634 (873)	15.1

It is seen that the speed-up of the eigenvalue solver is very satisfactory for these problem sizes, and larger problems are expected to scale up to a correspondingly higher number of processors. Notice that the superlinear speed-up is partly due to the fact that the blocksize and maximum subspace dimension varies with the number of processors which implies that the 1 processor run did not benefit from BLAS level 3 operations. A sequential run with the same parameters as the 4 processor parallel run led to a speed-up of 3.0 on 4 processors.

[1] Located at UNI•C in Lyngby, Denmark.
[2] The parameters used for the run were: vector length $n = 16409$, $N = 210$, $n_b =$ # of processors, $n_{\max} = N + 8n_b$, number of selfconsistent iterations = 4. The execution time is given for the eigensolver but with time for the complete runs including initialization (not yet fully parallelized) given in parenthesis.

These runs were made using only a single "**k**-point" (see e.g. [9]) subproblem, but an additional parallel dimension is actually available in the code for cases of more than one **k**-point and/or electron spin. Such parallelism should always be exploited where possible, since it will be optimally efficient and can easily be combined with the parallelization described above, leading to the possibility of utilizing large numbers of processors efficiently for many typical production runs.

4 Numerical Examples

In order to investigate the stability of the proposed algorithm and its sensitivity to choice of parameters, a large number of different examples have been tested. Some representative examples are shown in Fig. 1, 2 and 3.

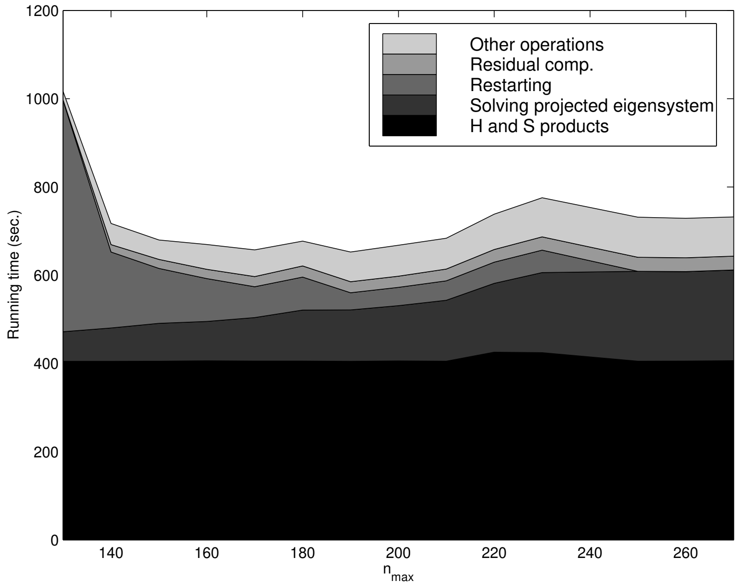

Fig. 1. Time spent in different parts of the code for an Au problem, using 6 processors ($n_b = 6$), $k_{\max} = 1$ and varying the maximum number of basis vectors, $n_{\max}$.

The experiments indicate that the size of n_b is not critical as long as it is chosen large enough for the solution of the projected eigensystem not to be unnecessarily large. The maximum subspace dimension, $n_{\max}$ is however important – if $n_{\max}$ is too small the overhead in restarting becomes dominant, on the other hand if $n_{\max}$ becomes unnecessarily large the memory consumption and the cost of solving the projected eigenvalue problem becomes large. From Fig. 3 it is seen that the overall work grows linearly when increasing the precision to which we solve the eigenvalue problem, indicating that it is not worthwhile to solve the eigenvalue problem to a high accuracy (for $k_{\max} = 1, 2, 3$ a total of 13 SCF iterations were needed – this was reduced to 12 for $k_{\max} = 5$).

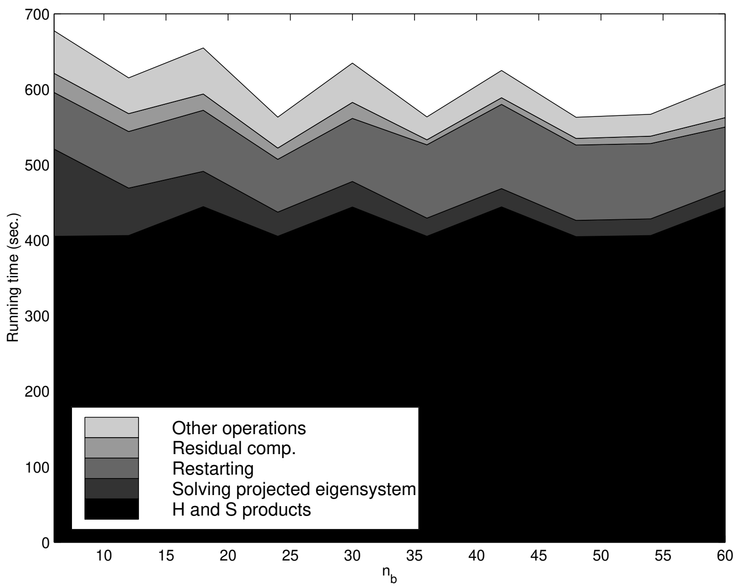

Fig. 2. Time spent in different parts of the code for an Au problem, using 6 processors, $k_{\max} = 1$, $n_{\max} = 180$ and varying the blocksize, n_b.

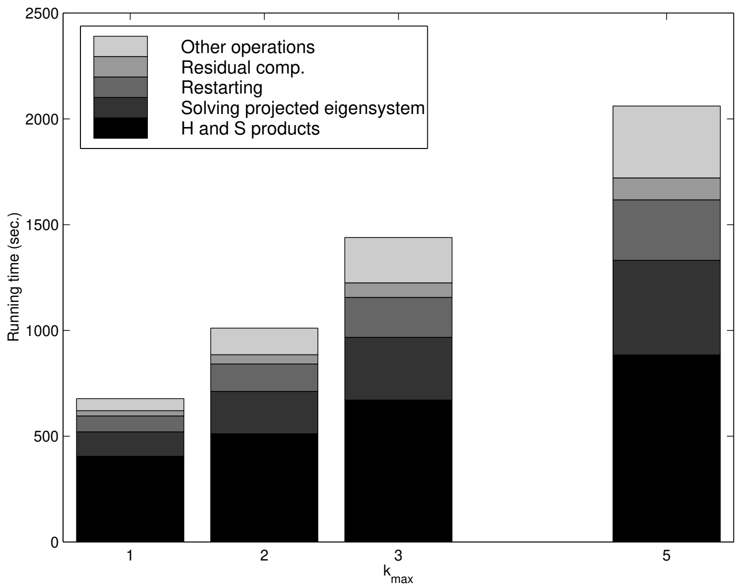

Fig. 3. Time spent in different parts of the code for an Au problem, using 6 processors ($n_b = 6$), $n_{\max} = 180$ and varying the maximum number of expansions on each eigenpair, $k_{\max}$.

A The Projected Newton Method

Given a series of input charge densities, $\underline{\rho}_{\text{in}}^{k-m}, \ldots, \underline{\rho}_{\text{in}}^{k}$ ($k \geq m \geq 1$) for Eq. 8, the corresponding output charge densities defined by Eq. 9 will be denoted as

$\underline{\rho}_{\text{out}}^{k-m}, \ldots, \underline{\rho}_{\text{out}}^{k}$. The residual of the charge density iteration is then defined by

$$\underline{r}^k = \underline{\rho}_{\text{out}}^k - \underline{\rho}_{\text{in}}^k$$

of which the Jacobian can be approximated by $\mathbf{T}_k \mathbf{G}_k^\dagger$, where $\mathbf{G}^\dagger$ denotes a pseudo inverse of $\mathbf{G}$, and we have defined

$$\begin{aligned} \mathbf{G}_k &= (\underline{\Delta\rho}^{k-m+1}, \ldots, \underline{\Delta\rho}^k), & \underline{\Delta\rho}^k &= \underline{\rho}_{\text{in}}^k - \underline{\rho}_{\text{in}}^{k-1} \\ \mathbf{T}_k &= (\underline{\Delta r}^{k-m+1}, \ldots, \underline{\Delta r}^k), & \underline{\Delta r}^k &= \underline{r}^k - \underline{r}^{k-1}. \end{aligned}$$

Applying Newton's method gives

$$\underline{\rho}_{\text{in}}^{k+1} = \underline{\rho}_{\text{in}}^k - \mathbf{G}_k \mathbf{T}_k^\dagger \underline{r}^k \tag{A.1}$$

where $\underline{\alpha}_k \equiv \mathbf{T}_k^\dagger \underline{r}^k$ is obtained as the least squares solution of

$$\mathbf{T}_k \underline{\alpha}_k = \underline{r}^k. \tag{A.2}$$

It is seen that the fixpoint solution $\underline{\rho}^* \equiv \underline{\rho}^k$ corresponding to $\underline{r}^k = \underline{0}$ will never be found unless $\underline{\rho}^* - \underline{\rho}^i$ (where i is equal to the initial k used) belongs to the span of $\mathbf{G}_i$ (i.e., the span of $\mathbf{G}_k$ remains constant). Therefore, a term $\beta(\underline{r}^k - \mathbf{T}_k \underline{\alpha}_k)$ is added to the right hand side of Eq. A.1 for some arbitrarily chosen full rank, diagonal matrix β as proposed in [1] (notice that $\underline{r}^k = \mathbf{T}_k \underline{\alpha}_k$ if $\mathbf{T}_k$ has full rank).

The least squares system Eq. A.2 is solved via the thin QR factorization on the augmented matrix $[\mathbf{T}_k \underline{\alpha}_k]$ as described in e.g. [2], and the factorization is updated rather than recomputed in the following iteration using the routines from [19].

As the above algorithm can not be used for the first iteration, the value $\underline{\rho}_{\text{in}}^1 = \underline{\rho}_{\text{in}}^0 + \beta \underline{r}^0$ is used instead. For the subsequent iterations the above procedure is applied with m incremented by one for each iteration (starting with $m = 1$) until a maximum value m_{max} is reached. Choosing m_{max} in the range of 5 to 10 generally works well for the present DFT problems.

References

[1] Donald G. Anderson, *Iterative Procedures for Nonlinear Integral Equations*, Journal of the ACM **12** (1965), no. 4, 547–560.

[2] Å. Björk, *Numerical Methods for Least Squares Problems*, SIAM, 1996.

[3] James H. Bramble, Jospeh E. Pasciak, and Andrew V. Knyazev, *A subspace preconditioning algorithm for eigenvector/eigenvalue computation*, Adv. Comp. Math. **6** (1996), 159–189.

[4] D. Calvetti, L. Reichel, and D. C. Sorensen, *An implicitly restarted lanczos method for large symmetric eigenvalue problems*, Electronic Transactions on Numerical Analysis **2** (1994), 1–21.

[5] M. Crouzeiz, B. Philippe, and M. Sadkane, *The Davidson Method*, SIAM J. Comput. Sci. **15** (1994), no. 1, 62–76.

[6] Ernest R. Davidson, *The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices*, J. Comp. Phys. **17** (1975), 87–94.

[7] ______, *Monster Matrices: Their Eigenvalues and Eigenvectors*, Comp. Phys. **7** (1993), no. 5, 519–522.

[8] J. Descloux, J.-L. Fattebert, and F. Gygi, *Rayleigh Quotient Iteration, an old Recipe for Solving Modern Large-Scale Eigenvalue Problems*, Computers in Physics **12** (1998), no. 1, 22–27.

[9] G. Kresse and J. Furthmüller, *Efficiency iterative schemes for ab initio total-energy calculations using a plane-wave basis set*, Phys. Rev. B **54** (1996), no. 16, 11169–11186.

[10] Yu-Ling Lai, Kun-Yi Lin, and Wen-Wei Lin, *An Inexact Inverse Iteration for Large Sparse Eigenvalue Problems*, Numerical Linear Algebra with Applications **4** (1997), no. 5, 425–437.

[11] D. C. Liu and J. Nocedal, *On the limited memory BFGS method for large scale optimization*, Math. Programming **45** (1989), 503–528.

[12] Message Passing Interface Forum, Univ. of Tennessee, Knowville, Tennessee, *MPI: A Message-Passing Interface Standard*, 1995.

[13] Ronald B. Morgan and David S. Scott, *Generalizations of Davidson's Method for Computing Eigenvalues of Sparse Symmetric Matrices*, SIAM J. Sci. Stat. Comput. **7** (1986), no. 3, 817–825.

[14] ______, *Preconditioning the Lanczos Algorithm for Sparse Symmetric Eigenvalue Problems*, SIAM J. Sci. Comput. **14** (1993), no. 3, 585–593.

[15] Jeppe Olsen, Poul Jørgensen, and Jack Simons, *Passing the one-billion limit in full configuration-interaction (FCI) calculations*, Chemical Physics Letters **169** (1990), no. 6, 463–472.

[16] B. N. Parlett, *The Symmetric Eigenvalue Problem*, SIAM, 1998.

[17] R. G. Parr and W. Yang, *Density-Functional Theory of Atoms and Molecules*, Oxford University Press, 1989.

[18] Péter Pulay, *Convergence Acceleration of Iterative Sequences. The Case of SCF iteration*, Chem. Phys. Lett. **73** (1980), no. 2, 393–398.

[19] L. Reichel and W. B. Gragg, *FORTRAN subroutines for updating the QR decomposition*, ACM Trans. Math. Software **16** (1990), 369–377.

[20] Y. Saad, A. Stathopoulos, J. Chelikowsky, K. Wu, and Öğüt, *Solution of Large Eigenvalue Problems in Electronic Structure Calculations*, BIT **36** (1996), no. 3, 563–578.

[21] Youcef Saad, *Numerical Methods for Large Eigenvalue Problems*, Manchester University Press, 1992.

[22] Gerard L. G. Sleijpen, Albert G. L. Booten, Diederik R. Fokkema, and Henk A. van der Vorst, *Jacobi-Davidson Type Methods for Generalized Eigenproblems and Polynomial Eigenproblems*, BIT **36** (1996), no. 3, 595–633.

[23] Gerard L. G. Sleijpen and Henk A. van der Vorst, *A Jacobi-Davidson Iteration Method for Linear Eigenvalue Problems*, SIAM J. Matrix Anal. Appl. **17** (1996), no. 2, 401–425.

[24] Andreas Stathopoulos, Yousef Saad, and Kesheng Wu, *Dynamic Thick Restarting of the Davidson and the Implicitly Restarted Arnoldi Methods*, SIAM J. Sci. Comput. **19** (1998), no. 1, 227–245.

[25] Michael P. Teter, Michael C. Payne, and Douglas C. Allan, *Solution of Schrödinger's equation for large systems*, Phys. Rev. B **40** (1989), no. 18, 12255–12263.

[26] David Vanderbilt, *Soft self-consistent pseudopotentials in a generalized eigenvalue formalism*, Phys. Rev. B **41** (1990), no. 11, 7892–7895.