Journal of Computational Physics 229 (2010) 441–452

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Accelerated Block Preconditioned Gradient method for large scale wave functions calculations in Density Functional Theory

J.-L. Fattebert

Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94551, United States

ARTICLE INFO

Article history:
Received 1 August 2008
Received in revised form 18 September 2009
Accepted 24 September 2009
Available online 4 October 2009

PACS:
71.15.−m
71.15.Dx

Keywords:
Block Preconditioned Gradient
Density Functional Theory
Kohn–Sham equations

ABSTRACT

An Accelerated Block Preconditioned Gradient (ABPG) method is proposed to solve electronic structure problems in Density Functional Theory. This iterative algorithm is designed to solve directly the non-linear Kohn–Sham equations for accurate discretization schemes involving a large number of degrees of freedom. It makes use of an acceleration scheme similar to what is known as RMM-DIIS in the electronic structure community. The method is illustrated with examples of convergence for large scale applications using a finite difference discretization and multigrid preconditioning.

1. Introduction

Fast iterative solvers for the Kohn–Sham (KS) equations are crucial to enable large-scale first-principles simulations. It is particularly important for Born–Oppenheimer molecular dynamics simulations where the electronic ground state needs to be calculated numerous times with a relatively high accuracy. Many different algorithms have been proposed in the physics and chemistry literature. New algorithms or algorithm adaptations however are always in need as larger and larger problems are being solved on modern parallel computers, using very accurate discretization schemes. This paper focuses on an iterative algorithm designed for solving directly the non-linear KS equations for accurate discretization schemes involving a large number of degrees of freedom. Such discretizations schemes include general numerical methods such as pseudo-spectral – usually referred to as Plane Wave (PW) in the electronic structure community – Finite Difference (FD), or Finite Element (FE) methods. For these numerical schemes the number of degrees of freedom to describe each electronic wave function is typically large compared to approaches based on smaller (physics-based) basis sets such as Linear Combinations of Atomic Orbitals (LCAO), but offers more flexibility, generality and a lower computational cost per degree of freedom. This paper features numerical examples based on a FD approach. The iterative algorithm is however directly applicable to other discretizations and will be presented in a general context. From a mathematical point of view, the discretization schemes mentioned above result in large sparse matrices – indirectly through Fast Fourier Transforms for the PW method – and lead to the use of matrix-free implementations of non-linear iterative solvers. As usual for this kind of approach, we assume

E-mail address: fattebert1@llnl.gov

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.09.035

atomic potentials are replaced by smooth non-local pseudopotential approximations. These implicitly include the core electrons not directly involved in chemical bonds. This makes many physical applications tractable using uniform meshes.

The first goal of this paper is to describe and analyze an iterative algorithm for solving the KS equations. But the aim is also to demonstrate how it can be efficiently used to solve very large scale problems which are becoming tractable with modern parallel computers and involve hundreds of eigenvalues. The algorithm can be described as an Accelerated Block Preconditioned Gradient (ABPG) method. It makes use of a simple preconditioner – geometric multigrid for a real-space discretization – to build correction vectors. The acceleration scheme introduced by Anderson [1] is used to enhance convergence of the gradient iteration. The basic algorithm was first introduced in the context of linear scaling methods with FD discretization [8]. An interesting aspect of the algorithm presented here is indeed its direct applicability in the linear scaling context, when electronic wave functions are confined to limited regions in space. It is so because no Rayleigh–Ritz procedure is required in ABPG. The method was also applied later in the context of a finite elements discretization [9], but limited to rather small problems. This paper illustrates how this algorithm can be rather efficient for large scale problems in the context of standard $O(N^3)$ approaches.

The idea of using an extrapolation scheme to accelerate convergence of an iterative scheme in electronic structure calculations goes back to Pulay [20] in 1980. He gave the name DIIS for *Direct Inversion of the Iterative Subspace* to his approach. Related algorithms are also found under the name "Pulay mixing". While Pulay described his original approach for extrapolating a Fock matrix based solution of a self-consistent calculation – represented in a space spanned by a set of local atomic orbitals – the name DIIS was later used in the community for various extrapolation algorithms.

In this paper, the focus is on using an extrapolation scheme to accelerate the calculation of the wave functions solution to the Kohn–Sham (KS) equations. For this purpose, Wood and Zunger [23] described an algorithm they call "RMM-DIIS", attributed to Bendt and Zunger. This algorithm uses the DIIS extrapolation scheme to minimize the residual of an eigenvalue problem associated to a targeted eigenpair, for a fixed Hamiltonian operator. Hutter et al. [13] proposed an adaptation of DIIS for the Kohn–Sham equations discretized in a Plane Waves basis set. They use the DIIS extrapolation scheme to directly solve the non-linear KS equations by minimizing preconditioned residuals. The RMM-DIIS idea was also used by Kresse and Furthmueller in the Plane Waves context [15]. They use RMM-DIIS as a solver for the surrogate problem associated with the linearized (frozen) Hamiltonian to be solved at each step of an iterative – self-consistent (SC) – process. The iterative solution of the full non-linear problem is then calculated using a mixing scheme (outer loop) to generate a better trial solution from a linear combination of the wave functions obtained at the most recent SC steps. The use of DIIS for calculating the wave functions solutions of the Hartree–Fock equations in a Gaussian basis sets is discussed in [10]. The possibility of formulating DIIS-type algorithms as accelerated inexact Newton schemes was pointed out by Harrison in [12]. Similar extrapolation schemes written under the "Anderson" form [1] were also used in the electronic structure community, for instance to accelerate self-consistent iterations by extrapolating potentials obtained at successive iterations [16]. The similarities between the Anderson extrapolation scheme and variants of Broyden's method in the context of electronic structure calculations were extensively discussed by Eyert [6].

While formulated in the form proposed by Anderson [1], the basic extrapolation idea used in our algorithm is closely related to the one used in the RMM-DIIS method (see Appendix A). Unlike [15] however, the method presented here attempts to solve *directly* the non-linear KS problem, without intermediate surrogate problems. That means our extrapolation scheme involves wave functions and residuals computed with the full non-linear Hamiltonian and not a surrogate Hamiltonian. The two algorithms also differ considerably for the case of multiple eigenvalues/eigenfunctions. The algorithm described here uses a *block* approach: all the electronic wave functions are updated simultaneously, using a single extrapolation step. Unlike [13] where extrapolation is performed on Ritz vectors, our approach is formulated for a general representation of the searched invariant subspace in a basis of non-orthogonal wave functions.

After introducing notations and formulating the Kohn–Sham (KS) equations in terms of non-orthogonal wave functions in Section 2, our iterative algorithm is described in Section 3. Details and practical implementation of the algorithm are presented in Section 4. The numerical examples in Section 5 illustrate how the algorithm can be applied very efficiently to solve large scale problems which require the computations of over a thousand eigenvalues.

2. Density Functional Theory

We consider the DFT energy functional written as a functional of N orthonormal electronic wave functions ψ_i (KS model)

$$E_{KS}\left[\{\psi_i\}_{i=1}^N\right] = \sum_{i=1}^N \int_\Omega \psi_i(r)(-\Delta\psi_i)(r)\,dr + \int_\Omega \int_\Omega \frac{\rho(r_1)\rho(r_2)}{|r_1 - r_2|}\,dr_1\,dr_2 + E_{XC}[\rho] + \sum_{i=1}^N \int_\Omega \psi_i(r)(V_{ext}\psi_i)(r)\,dr, \tag{1}$$

where ρ is the electronic density defined by

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

(see for example [3]). E_{KS} is made of the sum of the kinetic energy of the electrons, the Coulomb interaction between electrons, the exchange and correlation electronic energy, and the energy of interaction of the electrons with the potential

generated by all the atomic cores V_{ext}. Given an external potential V_{ext} – defined by the various atomic species present in the problem, their respective positions and pseudopotentials – the ground state of the physical system is obtained by minimizing the energy functional (1) under the orthonormality constraints

$$\int_\Omega \psi_i(r)\psi_j(r) = \delta_{ij}, \quad i,\ j = 1,\ldots,N. \tag{3}$$

To avoid mathematical difficulties irrelevant to the present study, let us assume from here on that we have to solve a problem in a finite dimensional space of dimension M resulting from the discretization of the above equations. To be concrete, suppose that we have a finite difference discretization on a uniform mesh with periodic boundary conditions, and thus the functions ψ_i are M-dimensional vectors with components corresponding to their values at the mesh points, $\psi_{i,k} = \psi_i(r_k)$. Let L_h denote the finite difference approximation of the Laplacian operator. Without loss of generality, wave functions are assumed to take real values only.

One can derive the Euler–Lagrange equations associated to the minimization problem (1) with N^2 Lagrange parameters corresponding to the orthonormality constraints (3). One obtains the so-called Kohn–Sham equations in their usual form for the particular choice of the functions $\{\psi_i\}_{i=1}^N$ which diagonalizes the matrix made of the Lagrange parameters,

$$\begin{aligned} &-L_h\psi_i + V_{KS}[\rho]\psi_i = \lambda_i\psi_i, \\ &\rho(r_k) = \sum_{i=1}^{N} |\psi_i(r_k)|^2, \\ &\sum_{k=1}^{M} \psi_i(r_k)\psi_j(r_k) = \delta_{ij}, \end{aligned} \tag{4}$$

where V_{KS} is a discretized non-linear effective potential operator (see *e.g.* [3]). In this approach, one has to find the N lowest eigenvalues $\lambda_i,\ i = 1,\ldots,N$ and the corresponding eigenfunctions. We assume here that $\lambda_{N+1} - \lambda_N > 0$.

Let V denote the set of matrices of M rows and N columns. We can represent the solution of the discretized problem as a matrix

$$\Psi = (\psi_1,\ldots,\psi_N) \in V. \tag{5}$$

Ψ represents the invariant subspace spanned by the eigenvectors associated to the N lowest eigenvalues. Using these notations, the KS equations are given by

$$-L_h\Psi + V_{KS}[\rho]\Psi = \Psi\Lambda \tag{6}$$

where Λ is an $N \times N$ diagonal matrix with diagonal entries $\Lambda_{ii} = \lambda_i,\ i = 1,\ldots,N$.

Let us assume that we know another representation of that same invariant subspace given by another set of N linearly independent vectors,

$$\Phi = (\phi_1,\ldots,\phi_N) \in V. \tag{7}$$

One can find an $N \times N$ matrix C such that

$$\Psi = \Phi C. \tag{8}$$

The matrix C satisfies

$$CC^T = S^{-1}, \tag{9}$$

where $S = \Phi^T\Phi$, the Gram matrix, is of rank N. Using relations (8) and (9), one can express the electronic density in terms of the matrix elements of Φ and S^{-1},

$$\rho_k = \sum_{i,j=1}^{N} (S^{-1})_{ij}\Phi_{ki}\Phi_{kj}, \tag{10}$$

where ρ_k denotes the value of the electronic density at the mesh point x_k. In this form we note that ρ is also independent of the particular representation chosen for the search invariant subspace. Also the KS equations for Φ can be rewritten as

$$-L_h\Phi + V_{KS}[\rho]\Phi = \Phi S^{-1}H_\Phi, \tag{11}$$

where $H_\Phi = \Phi^T(-L_h + V_{KS})\Phi$. For a trial solution Φ, the residual is given by

$$R(\Phi) = -L_h\Phi + V_{KS}[\rho]\Phi - \Phi S^{-1}H_\Phi. \tag{12}$$

It is easy to verify that

$$\Phi^T R(\Phi) = 0$$

and that

$$R(\Phi W) = R(\Phi)W$$

for any non-singular $N \times N$ matrix W.

In this formulation, unlike more traditional approaches which include an additional equation to enforce orthonormality as in Eq. (4), the columns of Φ constitute a general non-orthogonal basis of the trial invariant subspace. Some minimal caution needs to be taken to avoid cases where the columns of Φ become linear dependent. A general formulation based on non-orthogonal wave functions is convenient to implement extrapolation algorithms such as the one described later in this paper. It has also been used for conjugate gradient algorithm implementation in electronic structure calculations [22].

3. Accelerated Block Preconditioned Gradient for KS

The non-linear KS problem can be written as

$$R(\Phi) = -L_h\Phi + V_{KS}[\rho]\Phi - \Phi S^{-1}H_\Phi = 0. \tag{13}$$

To solve this equation, using our matrix notation, we write an inexact Newton iteration (see Appendix B) in the block form (at step k)

$$\Phi_{k+1} = \Phi_k - \bar{J}_k^{-1}R(\Phi_k). \tag{14}$$

Here $\bar{J}_k$ is an $M \times M$ matrix, a diagonal block of an approximate Jacobian written in a block diagonal form

$$\widetilde{J}_k = \begin{pmatrix} \bar{J}_k & 0 & \cdots & 0 \\ 0 & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & 0 \\ 0 & \cdots & 0 & \bar{J}_k \end{pmatrix}.$$

This "block" form considerably simplifies our approach as we assume one can find a good approximate Jacobian independent of the particular function ϕ_i (or column of Φ) we are correcting. That is also the key which lets us design an algorithm targeting the invariant searched subspace independently of its representation.

Let

$$P_k = -\bar{J}_k^{-1}R(\Phi_k), \tag{15}$$

denote the subspace correction computed at step k.

The simplest approximate Jacobian $\bar{J}_k$ one could use is the identity scaled by a factor

$$\bar{J}_k = \eta I.$$

In the language of eigensolvers, it leads to a block shifted power method (see *e.g.* [2, Section 11.3]). It is also called a gradient method for computing the smallest eigenvalue since R is collinear to the gradient of the Rayleigh quotient. Such an approach can be substantially improved if a good preconditioner T is available by choosing

$$\bar{J}_k = T.$$

Iteration (14) then becomes a block preconditioned gradient iteration. For a linear positive definite operator A, Neymeyr [17] demonstrated convergence to the invariant subspace spanned by the eigenvectors associated to the lowest N eigenvalues of A. The main assumptions are that $\lambda_{N+1} > \lambda_N$ and T is a good approximation of A in the sense that

$$\|I - TA\|_A \leqslant \gamma$$

for $\gamma \in (0,1)$. In practice, due to numerical rounding errors, this iteration converges for randomly generated initial trial subspaces. For an appropriate choice of the preconditioner, this algorithm can lead to a mesh independent convergence rate with a very low cost iteration. For KS equations discretized on a mesh or in Plane Waves, a good option is to use a preconditioner T which approximates $(-L_h)^{-1}$ in the high-frequency domain. For the numerical results presented in Section 5, we will use the multigrid preconditioner proposed in [7]. It is also easy to see that this block preconditioned gradient iteration leads to successive subspaces independent of the particular representation used for the initial subspace [7].

Newton's method is a one-step method which uses information only from the current step to improve a trial solution. This would be fast enough if we were considering an accurate solver for the Newton iteration. In the inexact Newton method case however, convergence rate is usually only linear, and acceleration can be achieved using information accumulated during the previous m steps, to build an improved trial solution. Such an approach is referred to as an Accelerated Inexact Newton (AIN) method [11]. Instead of simply adding the correction P_k to the current approximation Φ_k, the idea is to use the knowledge accumulated during the m previous inexact Newton steps to build a better update. Let

$$V_{k,m} := [\Phi_{k-m}, P_{k-m}, \ldots, \Phi_k, P_k] \tag{16}$$

be a search space. An improved update for Φ_k would be

$$\overline{\Phi}_k = V_{k,m} y, \tag{17}$$

where $y \in R^{(2m+2)N}$ is the solution of a projected problem in $V_{k,m}$.

Fokkema et al. [11] proposed various conditions (projected problems in appropriately chosen search subspace) to determine the vector of coefficients y: a Galerkin condition, a minimum residual condition, or a mix of both. In the Davidson–Liu algorithm [4] for instance, one would solve a Galerkin problem in the space

$$V_{k,DL} = [\Phi_k, P_k].$$

In LOBPCG [14], one would solve instead a Galerkin problem in the space

$$V_{k,LOBPCG} = [\Phi_{k-1}, \Phi_k, P_k]$$

to get the optimal solution within this subspace. However, an optimal algorithm has to take into account some of the features specific to DFT calculations: a non-linear operator and a large number of eigenpairs to compute. Yang et al. [24] proposed to generalize LOBPCG and solve iteratively a non-linear $3N \times 3N$ problem in $V_{k,LOBPCG}$. This is a very robust approach. However, building multiple times $3N \times 3N$ matrices, with elements made of dot products between pairs of M-dimensional vectors, becomes quite costly for large scale problems.

Here we use a much less computationally demanding method based on minimizing an extrapolated preconditioned residual. In [1], Anderson proposed an extrapolation based on the solution of a projected linear problem defined by the residuals computed at the m previous steps. The basic idea is to write an extrapolation scheme for Φ, using the approximations at the previous m steps,

$$\overline{\Phi}_k := \Phi_k + \sum_{j=1}^{m} \theta_j (\Phi_{k-j} - \Phi_k), \tag{18}$$

where the columns of Φ are assumed to be normalized at each step. Asymptotically and at first order, the same extrapolation scheme holds for the residual $R(\Phi)$, and thus for the preconditioned residual $P = -TR$. Thus we write

$$\overline{P}_k := P_k + \sum_{j=1}^{m} \theta_j (P_{k-j} - P_k). \tag{19}$$

Assuming that $\overline{P}_k$ is a good approximation of the error associated to a trial solution $\overline{\Phi}_k$ we define the real coefficients θ_j as those minimizing the norm of $\overline{P}_k$. For that we use the norm $\|\cdot\|_B$ corresponding to the dot product $(\cdot,\cdot)_B$ defined in the space V of $M \times N$ matrices as

$$(Q^T, Q)_B := \sum_{i=1}^{N} (B^{-1})_{ii} \mathbf{q}_i^T \mathbf{q}_i. \tag{20}$$

Here $\mathbf{q}_i$ denotes the column i of Q. B is a positive definite $N \times N$ matrix which we set to $B = \Phi_k^T \Phi_k$ at step k. The justification for using this norm is detailed in Appendix C. The coefficients θ_j can be then found by solving the $m \times m$ linear system defined by

$$\sum_{j=1}^{m} (P_k - P_{k-i}, P_k - P_{k-j})_B \theta_j = (P_k - P_{k-i}, P_k)_B, \quad i = 1, \ldots, m. \tag{21}$$

Finally, the new trial solution in $V_{k,m}$ is computed as

$$\Phi_{k+1} = \overline{\Phi}_k + \beta \overline{P}_k,$$

where $0 < \beta \leqslant 1$ is an additional parameter. In the examples presented in Section 5, we always use $\beta = 1$ as recommended in [1]. Smaller values could be considered if necessary for applications with extremely strong non-linearity.

Note that according to Appendix A, Eq. (21) can be regarded as a linearized minimum (preconditioned) residual condition

$$\min_{c \in \Re^{m+1}, \|c\|=1} \left\| \sum_{j=0}^{m} c_j P(\Phi_{k-j}) \right\|_B,$$

and thus a block variant of the RMM-DIIS extrapolation scheme for a non-linear eigenvalue problem.

4. Practical implementation

Since the ABPG algorithm makes use of a residual minimization for an eigenvalue problem, there is a risk of converging towards interior eigenvalues. Indeed the residual of an eigenvalue problem is zero for any eigenpair. To remedy this issue, extrapolation is turned off if the trial solution is too far away from convergence. This reduces ABPG to a block preconditioned gradient iteration in the initial steps when no good trial solution is known.

Another common issue observed in electronic structure problems solving with ABPG is the local non-convexity of the energy surface. In that case minimizing the residual would lead towards an energy maximum. Such a potential problem can be detected by examining the extrapolation coefficients θ_j. Indeed let us consider for example the case $m = 1$ and assume the energy functional E_{KS} is a quadratic function along the search direction. Let us also assume that $E_{KS}[\Phi_k] < E_{KS}[\Phi_{k-1}]$. If the energy functional is convex, then a proper extrapolation should lead to $\theta_1 < 0.5$ (indeed $\theta > 0.5$ would mean extrapolating back along the search direction beyond the mid-point between the states $k-1$ and k which would not be compatible with a quadratic function and an energy decrease going from $k-1$ to k). This suggests that one should discard coefficients larger than 0.5. For coefficients larger than 1, we actually assume local concavity and use an extrapolation coefficient $\theta_1 = -0.5$. We also treat with precaution coefficients smaller than -3 corresponding to very large extrapolation.

The complete procedure is detailed in Algorithm 1. One notices in particular that ABPG can be implemented with only one full Hamiltonian matrix application per iteration and per wave function. Other costly operations for large scale problems include building the matrices S and H_Φ, and potentially applying the preconditioner T.

Algorithm 1. Accelerated Block Preconditioned Gradient (ABPG)

$k \Leftarrow 0,\ \tilde{m} \Leftarrow 0,\ V_{-1} \Leftarrow [\,]$
Set Φ_0 to initial guess
Compute $S = \Phi_0^T\Phi_0,\ \rho_0,\ V_{KS}[\rho_0]$ and $H_\Phi = \Phi_0^T H_0 \Phi_0$
$R_0 \Leftarrow H_k\Phi_0 - \Phi_0 S^{-1} H_\Phi$
repeat
  $P_k \Leftarrow -TR_k$
  $V_k \Leftarrow [V_{k-1}, \Phi_k, P_k]$
  flag $\Leftarrow$ *true*
  while flag *and* $\tilde{m} > 0$
    flag $\Leftarrow$ *false*
    if $dim(span(V_k)) > 2(\tilde{m}+1)N$ **then**
      remove left $2N$ columns of V_k
    end if
    Build linear system (21)
    if cond(linear system) < 100 **then**
      Solve linear system (21) to determine $\theta_j,\ j = 1,\ldots,\tilde{m}$
      if $max_j(\theta_j) > 0.5$ *or* $min_j(\theta_j) < -3$ **then**
        if $\tilde{m} > 1$ **then**
          $\tilde{m} \Leftarrow \tilde{m} - 1$, *flag* $\Leftarrow$ *true* {Reduce history length}
        else
          if $\theta_1 > 1$ **then**
            $\theta_1 \Leftarrow -0.5$ {Fixed extrapolation for concave region}
          end if
          $\theta_1 \Leftarrow min(max(\theta_1, -3), 0.)$ {Limit extrapolation}
        end if
      end if
    else
      $\tilde{m} \Leftarrow \tilde{m} - 1$, *flag* $\Leftarrow$ *true* {Reduce history length}
    end if
  end while
  $y \in R^{2(\tilde{m}+1)N}$
  for $i = 1$ to N **do**
    $y(i) = 1 - \sum_{j=1}^{\tilde{m}} \theta_j,\ y(i+N) = \beta y(i)$
  end for
  for $j = 1$ to $\tilde{m}$ **do**
    for $i = 2jN + 1$ to $2jN + N$ **do**
      $y(i) = \theta_j,\ y(i+N) = \beta y(i)$
    end for
  end for
  $\Phi_{k+1} \Leftarrow V_k y$
  $k \Leftarrow k + 1$
  Update $S = \Phi_k^T\Phi_k,\ \rho_k,\ V_{KS}[\rho_k]$ and $H_\Phi = \Phi_k^T H_k \Phi_k$
  $R_k \Leftarrow H_k\Phi_k - \Phi_k S^{-1} H_\Phi$
until $\|R_k\| < tol$

For the particular case of a finite difference discretization used in this paper, this algorithm can be efficiently parallelized using a standard domain decomposition approach. Each processor is then responsible for the data associated to the pieces of the wave functions located in a particular subdomain. Finite difference operations then require local inter-processors communications to fill up "ghosts" values at subdomain boundaries. Global inter-processors communications are needed to sum up local contributions for each matrix elements and build matrices such S, H_Φ and the linear system (21). This approach was implemented in a C++/MPI code (MGmol) developed by the author and used for the numerical experiments presented in Section 5. The parallel library ScaLapack is used for linear algebra operations on $N \times N$ matrices.

5. Numerical results

As a first illustration, let us consider a simple eigenvalue problem

$$Au = \lambda u,$$

where A is a real symmetric $M \times M$ matrix. Without loss of generality, we can assume A diagonal. Suppose the diagonal elements of A are given by $a_{ii} = \lambda_i = i$ for $i = 0, \ldots, M-1$. Let us consider the problem of finding the lowest eigenvalue $\lambda_0 = 0$ and the associated eigenvector $u_0 = (1, 0, \ldots, 0)^T$. We start with an initial guess given by $\tilde{u}_0 = (0.447, 0.894, \ldots, 0)^T$, that is with non-zero components along the eigenvectors corresponding to the lowest two eigenvalues.

Let $M = 10$ and set $m = 1$, $T = 1/\lambda_{max} \cdot I$ where λ_{max} is the largest eigenvalue of A. For this simple problem the energy is the Ritz value associated with the current iterative solution $\tilde{u}$. It is also a measure of the error during the ABPG iterations. We plot the energy as a function of the angle between $\tilde{u}$ and u_1 – in the plane defined by u_0 and u_1. The solution is reached for an angle of $\pi/2$. Results are shown in Fig. 1. It can be observed that initial iterations are not very efficient. This is due to the fact that the preconditioner is not very good and the trial solution is in a concave region. But as soon as the convex region is reached, acceleration quickly leads to the solution.

Let us now illustrate the efficiency of ABPG in real DFT application. For all the applications presented below, we used a fourth order standard finite difference scheme to discretize the KS equations on a uniform mesh. The Local Density Approximation (LDA) exchange and correlation functional was used together with norm-conserving pseudopotentials. We also assume each electronic wave function is occupied by two electrons of opposite spin. All the numerical experiments shown below were carried out on Thunder, a Linux cluster system at Lawrence Livermore National Laboratory (Intel Itanium2 Tiger4 1.4 GHz with Quadrics interconnect).

The first DFT application consists of calculating the ground state of a silicon cluster passivated with H atoms at the surface ($Si_{35}H_{36}$). The problem is discretized on a uniform mesh $64 \times 64 \times 64$ with a mesh spacing $h = 0.5$ Bohr. Convergence for the total energy and the residual defined by Eq. (12) is shown in Fig. 2 using ABPG with various history lengths m. The case $m = 0$ corresponds to a block preconditioned gradient iteration with no acceleration. The trial wave functions were initialized as Gaussian functions centered on atomic bonds. The numerical results show that the effect of acceleration is quite important for $m = 1$ and $m = 2$. Going beyond $m = 2$ does not help much. For $m = 0$, the wall clock time was 3.5 s/step using 16 processors. Adding acceleration, the overhead in computer time was about 10% for $m = 1$, going up to 25% for $m = 3$. In Fig. 2 and following, the norm refers to the $\|\cdot\|_B$ norm introduced in Section 3.

To illustrate the applicability of the algorithm described in this paper to really large problems, we consider the calculation of the electronic structure of a silicon crystal of 512 atoms. In a typical pseudopotential DFT calculation for this system, one has to compute 1024 wave functions. The problem is discretized on a uniform $64 \times 64 \times 64$ mesh. Wave functions were initialized as Gaussian functions centered on bonds. Convergence of ABPG is shown in Fig. 3. The wall clock time for $m = 0$ was measured at 180 s/step using 32 processors, with an overhead of only 1% for $m = 1$.

Problems involving dangling bonds are typically more difficult to solve than the two previous examples – which had fully saturated bonds – and iterative algorithms usually converge slower towards the ground state. To illustrate the efficiency of the ABPG algorithm on such problems, it is applied to the ground state calculation of a diamond C(100) surface. A slab made

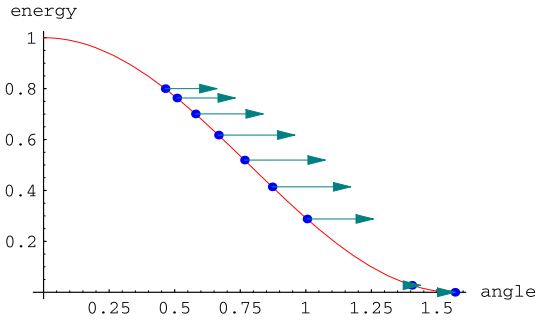

Fig. 1. Convergence of ABPG algorithm for search of lowest eigenvalue of matrix A (see text). Arrows denote magnitude of gradient at each trial solution. No acceleration is performed until the convex region is reached.

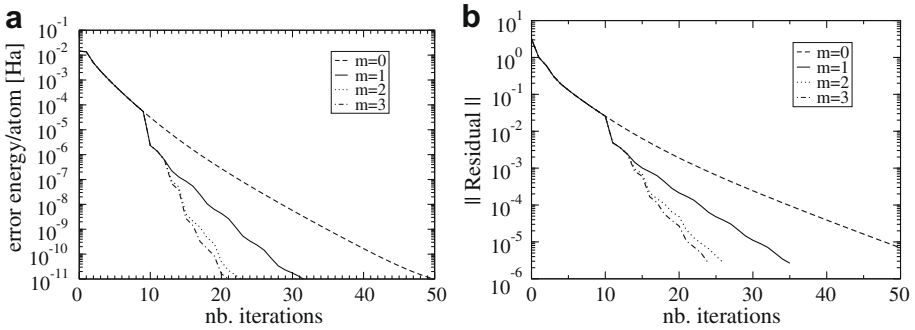

Fig. 2. Convergence of energy and residual for $Si_{35}H_{36}$ cluster for various history length (m).

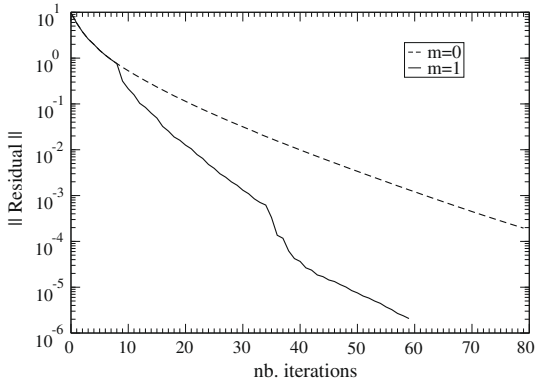

Fig. 3. Convergence for Si_{512} crystal ($N = 1024$).

of 12 layers of C atoms was used, with one surface terminated by a layer of H atoms to passivate the dangling bonds and the other surface reconstructed, as proposed for instance in [25]. The unit cell was repeated four times along each of the two axis parallel to the surface, making it a 416 atoms system, with 784 electronic wave functions to compute. The mesh used was made of 144 grid points in the direction orthogonal to the slab, and 96×96 in the plane parallel to the surface, for a mesh spacing $h = 0.28$ Bohr. Fig. 4 shows the convergence of the ABPG algorithm for a 2×1 reconstructed surface. Compared to the two previous examples, a slower but quite good convergence rate is observed. Using 108 processors for this calculation, the wall clock time was 83 s/step for $m = 0$. For $m = 1$, the overhead was about 2.5 s/step.

In practical applications, the electronic structure ground state often needs to be computed many times for slightly different atomic configurations. This is the case for molecular dynamics (MD) simulations or geometry optimizations where the electronic structure is used to evaluate forces acting on atoms at each step of the atomic configuration trajectory. In that case

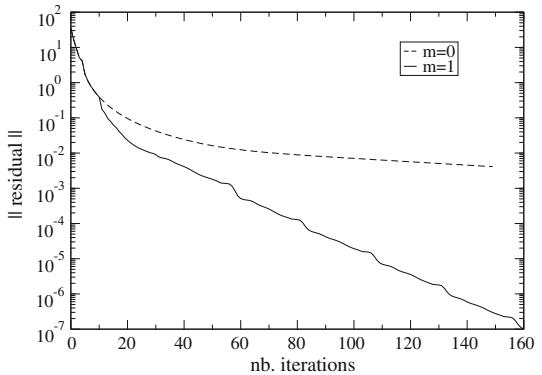

Fig. 4. ABPG Convergence for reconstructed (1 0 0) diamond surface ($N = 784$).

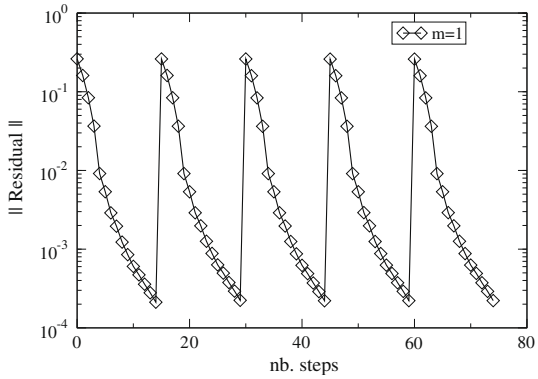

Fig. 5. ABPG convergence ($m = 1$) for a liquid water system with 64 H_2O molecules ($N = 256$): residual decrease for five consecutive steps of molecular dynamics. The line is simply a guide to the eye.

the initial trial wave functions are provided by the solution computed for the previous atomic configuration or an extrapolation using a few additional previous time steps. The problem then becomes to quickly reach the new ground state for a slightly different Hamiltonian. The performance of ABPG in this situation is illustrated in Fig. 5. It shows the convergence of the residual for five consecutive steps of molecular dynamics of liquid water at ambient conditions (64 molecules cell, mesh $128 \times 128 \times 128$). At each MD step k, wave functions are initialized with an initial guess given by

$$\widetilde{\Phi}_k = 2\Phi_{k-1} - \Phi_{k-2},$$

where Φ_{k-1} and Φ_{k-2} are the solutions obtained for the atomic configurations $k-1$ and $k-2$. For the iterations shown in Fig. 5, $m = 1$ and Anderson extrapolation is performed at every step except for the first step after an update of the atomic positions. The jumps in residual correspond to atomic configurations updates. The measured wall clock time for each ABPG step was 33 s using 64 processors.

6. Concluding remarks

This paper describes an Accelerated Block Preconditioned Gradient method to solve electronic structure problems in Density Functional Theory. The iterative algorithm is designed to solve directly the non-linear Kohn–Sham equations for accurate discretization schemes involving a large number of degrees of freedom. It makes use of a scheme similar to RMM-DIIS, using approximate solutions and preconditioned residuals from the m previous iterative steps to accelerate convergence.

As seen from the numerical results presented in the previous section, small values of m are usually appropriate for the ABPG algorithm. The optimal value for m depends on the preconditioner T, but $m = 1$ or $m = 2$ are often good values. Using larger values often results in bad conditioning for the linear system (21) and m being cut automatically (see Algorithm 1).

A major assumption for the ABPG algorithm to work well is to have a band gap in the eigenvalues spectrum, $\lambda_N < \lambda_{N+1}$. If $\lambda_N = \lambda_{N+1}$, extrapolation is not valid anymore for residuals corresponding to Ritz vectors associated to the eigenvalues close to λ_N. One way to remedy this issue is to use the single particle density matrix at finite temperature for B in the definition of the dot product $(\cdot,\cdot)_B$ (Eq. (20)) in order to progressively decrease the weight of the contributions coming from the Ritz vectors associated with eigenvalues close to λ_N. Further study is needed for this problem.

Acknowledgments

This work was performed under the auspices of the US Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 and was supported by the Office of Science, US Department of Energy, SciDAC Grant DE-FC02-06ER46262. The author would also like to thank the referees for their useful comments which helped improve the manuscript considerably.

Appendix A. Extrapolation coefficients

To determine extrapolation coefficients at step k of a DIIS iterative process [20], an extrapolated error vector $\bar{e}_k$ is introduced

$$\bar{e}_k = \sum_{i=0}^{m} c_i e_{k-i}.$$

It is made of a linear combination of the estimated current error e_k and the estimated errors at the previous m steps. The coefficients c_i are determined by requiring that the 2-norm of $\bar{e}_k$ is minimized under the constraint

$$\sum_{i=0}^{m} c_i = 1. \tag{A.1}$$

This leads to a system of $m+2$ linear equations for the $m+1$ coefficients c_i and the Lagrange parameter associated with the constraint.

A similar extrapolation scheme had been proposed earlier by Anderson [1] for solving more general non-linear equations. Anderson writes

$$\bar{e}_k = e_k + \sum_{1=1}^{m} \theta_i (e_{k-i} - e_k).$$

The latter equation can also be written as

$$\bar{e}_k = \left(1 - \sum_{1=1}^{m} \theta_i\right) e_k + \sum_{1=1}^{m} \theta_i e_{k-i}.$$

Thus, the equivalent DIIS coefficients are given by

$$c_i = \theta_i, \quad i = 1, \ldots, m,$$
$$c_0 = 1 - \sum_{1=1}^{m} \theta_i,$$

and clearly satisfy the constraint (A.1). The relation between the two formulations can actually be used as a way to eliminate one variable when solving the DIIS equations [15,18].

The particular choice of the error function varies depending on the specific quantity one tries to minimizes: residual of eigenvalue problem, computed iterative correction, electronic density self-consistent updates, or even atomic forces in geometry optimizations.

Appendix B. Inexact Newton method for eigenvalue problems

We consider the general non-linear equation

$$F(u) = 0, \tag{B.1}$$

where F is some smooth non-linear functional defined in a finite dimensional space $\mathcal{V} \in R^m$, where m is typically very large. One type of such equations is

$$A(u) = \lambda(u) u, \tag{B.2}$$

with $\lambda(u) = (u, Au)/(u, u)$, and A is a non-linear operator. The non-linearity of A – a specific feature of Density Functional Theory equations – often leads to look beyond standard eigensolvers for an efficient solution of this problem.

Newton's method is a well known iterative approach to solve non-linear equations such as (B.1) (see *e.g.* [19]). One basic iterative step to improve an approximate solution u_k at step k can be written as

$$u_{k+1} = u_k - J_k^{-1} F(u_k), \tag{B.3}$$

where $J_k := F'(u_k)$. Thus we can write a linear equation for the correction p of u_k,

$$J_k p_k = -r_k, \tag{B.4}$$

where $r_k = F(u_k)$. Unfortunately, the Jacobian J_k is often not available or is practically impossible to compute due to numerical cost. It can also be computationally very expensive or impossible to solve exactly the linear system (B.4). For large m, it is more practical to use an approximate Jacobian $\widetilde{J}_k$, leading to an inexact Newton iteration [5]

$$u_{k+1} = u_k - \widetilde{J}_k^{-1} F(u_k). \tag{B.5}$$

The inexact correction equation can then also be solved approximately, by an iterative method such as multigrid for example. Note that even if Eq. (B.4) is solvable, searching for an accurate solution may not be efficient if the quadratic model used to derive the Newton equation differs significantly from the real behavior of $F(u)$.

If we are in a quadratic regime close to a solution $\bar{u}$, where $r_k = J_k(u_k - \bar{u})$, we have, using (B.5),

$$u_{k+1} = \bar{u} + \left(I - \widetilde{J}_k^{-1} J_k\right)(u_k - \bar{u}). \tag{B.6}$$

This simple iterative process converges if the eigenvalues of the matrix $(I - \widetilde{J}_k^{-1} J_k)$ are of modulus smaller than 1. The convergence rate obviously depends on the largest eigenvalues of $(I - \widetilde{J}_k^{-1} J_k)$ and thus on how well $\widetilde{J}_k$ approximates J_k. Note that

for an eigenvector search, since the solution is defined up to normalization factor, one should instead consider the projected operator [21]

$$\left(I - u_k u_k^T\right)\left(1 - \widetilde{J}_k^{-1} J_k\right)\left(I - u_k u_k^T\right).$$

Appendix C. Matrix norm

Since we are interested in determining an invariant subspace and not a particular representation of that subspace, the idea is to use a dot product leading to an iterative scheme independent of the particular choice of the basis used to represent the search subspace. For $Q_1,\ Q_2 \in V$, let us define

$$(Q_1, Q_2)_B := Tr\left(B^{-1} Q_1^T Q_2\right) \tag{C.1}$$

for a given positive definite matrix B. Now if Φ_k is a particular representation of the trial search subspace at step k, we can find a non-singular $N \times N$ matrix C_k which maps Φ_k into a matrix made of orthonormal columns, *i.e.* such that $C_k^T \Phi_k^T \Phi_k C_k = I$. C_k also satisfies

$$C_k C_k^T = \left(\Phi_k^T \Phi_k\right)^{-1}.$$

Now if we choose $B = \Phi_k^T \Phi_k$, we have

$$(Q_1, Q_2)_B = Tr\left(C_k C_k^T Q_1^T Q_2\right) = Tr\left(C_k^T Q_1^T Q_2 C_k\right).$$

Thus the dot product $(Q_1, Q_2)_B$ between two elements of V is defined as the standard Frobenius dot product after applying a linear transform C_k to Q_1 and Q_2. Conveniently, this dot product is invariant with respect to any orthogonal transformation U which maps $\Phi_k C_k$ into any other orthonormal representation $\Phi_k C_k U$ of the trial search subspace, and thus does not depend on the particular choice of C_k.

The net result of using this dot product is that if all the matrices $\Phi_{k-j},\ j = 0, \ldots, m$ were to be transformed by multiplication on the right by a non-singular matrix $\widetilde{C}$, the extrapolation scheme would result in the same subspace, although represented by $\overline{\Phi}_k \widetilde{C}$ instead of $\overline{\Phi}_k$. Since the block preconditioned gradient iteration is also independent of the particular representation used for the subspace, the whole iteration becomes representation independent by setting $B = \Phi_k^T \Phi_k$ at each step k.

In practice, the evaluation of the dot product $(\cdot,\cdot)_B$ defined in (C.1) becomes quite expensive for large N since it involves N^2 dot products between vectors of length M. Thus the approximation

$$Tr(B^{-1} Q^T Q) \sim \sum_{i=1}^{N} (B^{-1})_{ii} \mathbf{q}_i^T \mathbf{q}_i$$

is used (see Section 3). It requires only N dots products. (Here $\mathbf{q}_i$ denotes the column i of Q.) This approximation becomes exact if Φ_k is made of orthogonal vectors and $B = \Phi_k^T \Phi_k$ is diagonal. If vectors are orthogonalized at regular intervals, this approximation remains very good. For instance, if orthonormalization is performed once after every update of the atomic positions in a molecular dynamics simulation, this is sufficient to effectively have no visible effect on the convergence rate.

References

[1] D.G. Anderson, Iterative procedures for nonlinear integral equations, J. Ass. Comput. Mach. 12 (4) (1965) 547–560.
[2] Zhaojun Bai, James Demmel, Jack Dongarra, Axel Ruhe, Henk van der Vorst (Eds.), Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.
[3] E. Cances, M. Defranceschi, W. Kutzelnigg, C. LeBris, Y. Maday, Computational chemistry: a primer, in: P.G. Ciarlet, C. Le Bris (Eds.), Handbook of Numerical Analysis, vol. X, Elsevier Science, 2003, pp. 3–270.
[4] E.R. Davidson, Monster matrices: their eigenvalues and eigenvectors, Comput. Phys. 7 (5) (1993) 519–522.
[5] R.S. Dembo, S.C. Eisenstat, T. Steihaug, Inexact Newton methods, SIAM J. Numer. Anal. 19 (2) (1982) 400–408.
[6] V. Eyert, A comparative study on methods for convergence acceleration of iterative vector sequences, J. Comput. Phys. 124 (2) (1996) 271–285.
[7] J.-L. Fattebert, J. Bernholc, Towards grid-based $O(N)$ density-functional theory methods: optimized non-orthogonal orbitals and multigrid acceleration, Phys. Rev. B 62 (3) (2000) 1713–1722.
[8] J.-L. Fattebert, F. Gygi, Linear scaling first-principles molecular dynamics with controlled accuracy, Comput. Phys. Commun. 162 (2004) 24–36.
[9] J.L. Fattebert, R.D. Hornung, A.M. Wissink, Finite element approach for density functional theory calculations on locally-refined meshes, J. Comput. Phys. 223 (2) (2007) 759–773.
[10] Thomas H. Fischer, Jan Almlof, General methods for geometry and wave function optimization, J. Phys. Chem. 96 (24) (1992) 9768–9774.
[11] Diederik R. Fokkema, Gerard L.G. Sleijpen, Henk A. van der Vorst, Accelerated inexact Newton schemes for large systems of nonlinear equations, SIAM J. Sci. Comput. 19 (2) (1998) 657–674.
[12] R.J. Harrison, Krylov subspace accelerated inexact Newton method for linear and nonlinear equations, J. Comput. Chem. 25 (3) (2004) 328–334.
[13] J. Hutter, H.P. Luethi, M. Parrinello, Electronic structure optimization in plane-wave-based density functional calculations by direct inversion in the iterative subspace, Comput. Mater. Sci. 2 (1994) 244–248.
[14] Andrew V. Knyazev, Toward the optimal preconditioned eigensolver: locally optimal block preconditioned conjugate gradient method, SIAM J. Sci. Comput. 23 (2) (2002) 517–541.

[15] G. Kresse, J. Furthmueller, Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set, Comput. Mater. Sci. 6 (1) (1996) 15–50.
[16] L.F. Mattheiss, D.R. Hamann, Linear augmented-plane-wave calculation of the structural properties of bulk Cr, Mo, and W, Phys. Rev. B 33 (2) (1986) 823–840.
[17] Klaus Neymeyr, A geometric theory for preconditioned inverse iteration applied to a subspace, Math. Comput. 71 (237) (2002) 197–216.
[18] Peng Ni, Homer Walker, A linearly constrained least-squares problem in electronic structure computations, ICCES 7 (1) (2008) 43–49.
[19] Jorge Nocedal, Stephen J. Wright, Numerical Optimization, Springer, 1999.
[20] Peter Pulay, Convergence acceleration of iterative sequences. The case of SCF iteration, Chem. Phys. Lett. 73 (2) (1980) 393–398.
[21] Gerard L.G. Sleijpen, Henk A. van der Vorst, A generalized Jacobi–Davidson iteration method for linear eigenvalue problem, SIAM Matrix Anal. Appl. 17 (2) (1996) 401–425.
[22] I. Stich, R. Car, M. Parrinello, S. Baroni, Conjugate gradient minimization of the energy functional: a new method for electronic structure calculation, Phys. Rev. B 39 (8) (1989) 4997–5004.
[23] D.M. Wood, A. Zunger, A new method for diagonalising large matrices, J. Phys. A: Math. Gen. 18 (1985) 1343–1359.
[24] Chao Yang, Juan C. Meza, Lin-Wang Wang, A constrained optimization algorithm for total energy minimization in electronic structure calculations, J. Comput. Phys. 217 (2) (2006) 709–721.
[25] Sang H. Yang, David A. Drabold, James B. Adams, Ab initio study of diamond C(100) surfaces, Phys. Rev. B 48 (8) (1993) 5261–5264.