Solving a Class of Nonlinear Eigenvalue Problems by Newton's Method *

Weiguo Gao
Department of Mathematics
Fudan University
Shanghai, China
wggao@fudan.edu.cn

Chao Yang and Juan Meza
Computational Research Division
Lawrence Berkeley National Laboratory
Berkeley, CA 94720
{cyang,jcmeza}@lbl.gov

July 6, 2009

Abstract

We examine the possibility of using the standard Newton's method for solving a class of nonlinear eigenvalue problems arising from electronic structure calculation. We show that the Jacobian matrix associated with this nonlinear system has a special structure that can be exploited to reduce the computational complexity of the Newton's method. Preliminary numerical experiments indicate that the Newton's method can be more efficient for small problems in which a few smallest eigenpairs are needed.

1 Introduction

We are concerned with solving the following type of nonlinear eigenvalue problem

$$H(X)X = X\Lambda_k, \tag{1}$$

where $X \in \mathbb{R}^{n\times k}$, $X^TX = I_k$, $H(X) \in \mathbb{R}^{n\times n}$ is a matrix that has a special structure to be defined below, and $\Lambda_k \in \mathbb{R}^{k\times k}$ is a diagonal matrix consisting of the k smallest eigenvalues of $H(X)$. This type of problem arises in electronic structure calculations [14, 10]. The nonlinearity simply refers to the dependency of the matrix H on the eigenvector X to be computed. This dependency is expressed through a vector $\rho(X)$ that represents the *charge density* of electrons in a molecule or solid. This vector is defined as

$$\rho(X) \equiv \mathrm{diag}(XX^T), \tag{2}$$

where $\mathrm{diag}(L)$ denotes the vector containing the diagonal elements of the matrix L.

*This work was supported by the Director, Office of Science, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy under contract number DE-AC02-05CH11231.

Given $\rho(X)$, the matrix $H(X)$ that we will consider in this paper is defined as

$$H(X) = L + \mathrm{Diag}(L^\dagger \rho(X) - \gamma \rho(X)^{1/3}), \tag{3}$$

where L is a discretized Laplacian, $L^\dagger$ is the inverse or pseudoinverse of L (depending on the boundary condition imposed on the continuous problem), $\mathrm{Diag}(x)$ (with an uppercase D) denotes a diagonal matrix with x on its diagonal, and $\gamma \geq 0$ is some known constant. In electronic structure calculations, $H(X)$ is often referred to as a single-particle Hamiltonian.

The nonlinear eigenvalue problem (1) can be derived from the first order necessary condition of a constrained minimization problem in which a total energy function

$$E(X) = \frac{1}{2}\mathrm{trace} X^T L X + \frac{1}{4}\rho(X)^T L^\dagger \rho(X) - \frac{3}{4}\gamma \rho(X)^T \rho(X)^{1/3}, \tag{4}$$

is minimized subject to the orthonormality constraint $X^T X = I_k$. Thus solving (1) is equivalent to solving a constrained minimization problem.

Note that the solution to (4) or (1) is not unique. If X is a solution, then XQ is also a solution for any $Q \in \mathbb{R}^{k\times k}$ such that $Q^T Q = I_k$. That is, the solution to the constrained minimization problem or, equivalently, the nonlinear equations defined by (1) is a k-dimensional invariant subspace in $\mathbb{R}^n$ rather than a specific matrix. In particular, Q can be chosen such that Λ_k is diagonal. In this case, X consists of k eigenvectors associated with the k smallest eigenvalues of $H(X)$.

Currently, the most widely used approach for solving this type of problem numerically is to apply the so called *Self Consistent Field* (SCF) iteration. In each SCF iteration, one computes approximations to a few smallest eigenvalues and their corresponding eigenvectors of a fixed Hamiltonian defined by the current approximation to the wavefunctions X, the computed eigenvector approximations are used to construct a new Hamiltonian. When difference between the new and the previous Hamiltonians is negligible, the SCF iteration is terminated, and the eigenvectors of the Hamiltonian become self-consistent.

In [15], the SCF iteration is viewed as an optimization procedure that seeks the minimizer of the total energy function indirectly by minimizing a sequence of quadratic surrogate functions. These surrogates are constructed in such a way that their gradients match with that of the total energy function at the current approximate wavefunctions.

An alternative algorithm is proposed in [15] to minimize the total energy function directly. The key ingredients of the direct constrained minimization (DCM) algorithm involve projecting the total energy function into a sequences of low dimensional subspaces and seeking the minimizer of total energy function within each subspace. The low dimensional subspace is constructed from the current eigenvector approximation, the gradient of the total energy and the previous search direction. No second derivative information is employed. Thus the convergence rate of this algorithm is at best superlinear.

Neither the SCF iteration nor the DCM algorithm is a standard optimization algorithm for solving a constrained minimization problem. A natural question one may ask is whether the existing optimization techniques can be applied directly to (1), which can be viewed a system of nonlinear equations, or (4), a constrained minimization problem. If so, how effective are they in comparison to SCF and DCM? In this paper, we explore the feasibility of applying the Newton's method to (1) or (4). In particular, we describe the structure of the Jacobian matrix associated with (1) (or the Hessian associated with (4)). We discuss how to compute the Newton step efficiently and present some numerical results that demonstrate the quadratic convergence of the Newton's method when applied to (1) or (4), and the overall cost of this approach in comparison with SCF.

2 Newton's Method

The nonlinear eigenvalue problem (1) can be viewed as a set of nonlinear equations $R(X, \Lambda) = 0$, where

$$R(X,\Lambda) = \begin{bmatrix} F(X,\Lambda) \\ G(X) \end{bmatrix} \equiv \begin{bmatrix} H(\rho(X))X - X\Lambda \\ X^T X - I \end{bmatrix}, \tag{5}$$

and Λ is a symmetric $k \times k$ matrix. Hence we may apply the Newton's method directly to obtain the solution of (1), if a good initial guess is available. Note that $X^T X = I$ contains only $k(k+1)/2$ non-redundant equations. Therefore, the total number of non-redundant equations in (5) is $nk + k(k+1)/2$, which is identical to the total number of unknowns in X and Λ. As we mentioned earlier, once the solution to (5) is obtained, it is easy to turn Λ into a diagonal matrix by finding an unitary matrix Q such that $(XQ, Q^T \Lambda Q)$ is also a solution to (5) and $Q^T \Lambda Q$ is diagonal.

Given an initial guess $(X^{(0)}, \Lambda^{(0)})$ to the solution of (1), the standard Newton's method for solving (1) proceeds as follows:

1. At the ℓth iteration, compute the Newton correction $Z^{(\ell)} \equiv (\Delta X^{(\ell)}, \Delta\Lambda^{(\ell)})$ by solving

$$J^{(\ell)} Z^{(\ell)} = -R(X^{(\ell)}, \Lambda^{(\ell)}), \tag{6}$$

   where $J^{(\ell)}$ is the Jacobian matrix of F with respect to elements of X and Λ evaluated at $(X^{(\ell)}, \Lambda^{(\ell)})$.

2. Update the solution

$$\begin{bmatrix} X^{(k+1)} \\ \Lambda^{(k+1)} \end{bmatrix} = \begin{bmatrix} X^{(\ell)} \\ \Lambda^{(\ell)} \end{bmatrix} + \beta \begin{bmatrix} \Delta X^{(\ell)} \\ \Delta\Lambda^{(\ell)} \end{bmatrix}, \tag{7}$$

   where β is an appropriate step length.

Note that we write the correction equation in the form of (6) merely for convenience. The matrices $R(X^{(\ell)}, \Lambda^{(\ell)})$ and $(\Delta X^{(\ell)}, \Delta\Lambda^{(\ell)})$ should be treated as vectors with $m = (2n + k + 1)k/2$ components, and the dimension of $J^{(\ell)}$ is $m \times m$. The structure of this Jacobian matrix will be examined in the next section.

Note that the solution to (1) is not unique. In particular, not all solutions to (1) minimize the total energy function in the original constrained minimization problem. Hence, one of the drawbacks of applying the Newton's method to (5) is that it may converge to a solution to (1) that does not yield the minimum total energy if $(X^{(0)}, \Lambda^{(0)})$ is chosen arbitrarily.

Although this problem can be mitigated somewhat by using the Newton's method to solve the constrained minimization problem (4) using for example, the augmented Lagrangian method [2] or an interior point method [3], it cannot be completely eliminated because the constrained minimization problem is nonconvex. Therefore, the optimization procedure may converge to a local minimum when the starting guess is not sufficiently close to the solution of (1).

In this paper, we focus on the local convergence of the Newton's method by assuming that a good initial guess is available. Such an initial guess can be obtained by running, for examples, a few iterations of the SCF method or the direct constrained minimization (DCM) algorithm described in [15].

Another potential problem with the standard Newton's method is that it does not take into account the invariance property of $E(X)$ and $H(X)$. Our numerical examples to be shown in Section 6 indicates that this does not appear to delay the convergence of the Newton's method. Once convergence is reached, the orthonormality constraint is automatically satisfied. On the other hand, it is possible to develop a Newton's method that preserves the orthonormality constraint throughout the Newton iteration. This approach is taken by Edelman et al. [5] who proposed to minimize the total energy function on the

Grassmann manifold defined by $X^T X = I$. We will refer to this approach as the Grassmann Manifold Newton's (GNM) method. At the ℓth GNM iteration, a search direction Z is obtained by solving

$$\begin{aligned} E_{XX}\langle Z\rangle - Z(X^T E_X) &= -G, & (8)\\ Z^T X^{(\ell)} &= = 0, & (9) \end{aligned}$$

where E_{XX} denotes the $nk \times nk$ second derivative matrix defined element-wise by

$$(E_{XX})_{ij,kl} = \frac{\partial^2 E}{\partial X_{ij} \partial X_{kl}}, \tag{10}$$

and evaluated at $X^{(\ell)}$, $E_{XX}\langle Z\rangle$ denotes an $n \times k$ matrix that satisfies

$$\text{trace}(Y^T E_{XX}\langle Z\rangle) = \text{vec}(Y)^T E_{XX} \text{vec}(Z),$$

for any $n \times k$ matrix Y, the first derivative matrix E_X is defined element-wise by

$$(E_X)_{ij} = \frac{\partial E}{\partial X_{ij}},$$

and

$$G = (I - X^{(\ell)} {X^{(\ell)}}^T) E_X.$$

To obtain a better approximation, we seek an optimal "step length" t such that

$$X(t) = YV \cos(\Sigma t) V^T + U \sin(\Sigma t) V^T,$$

where $U\Sigma V^T$ is the compact singular value decomposition of Z, yields the minimum $E(X(t))$.

The key to a successful implementation of either the standard Newton's method or the GMN method is to solve the correction equations (6), (8) and (9) efficiently. We will discuss how that can be accomplished in section 4.

3 The Structure of the Jacobian Matrix

Before we discuss how to solve (6) and (10) efficiently, let us first examine the structure of the Jacobian matrix $J^{(\ell)}$ appeared in (5) and the second derivative matrix E_{XX} in (10). Recall that the dimension of $J^{(\ell)}$ is $m \times m$, where $m = (2n + k + 1)k/2$. Hence, a brute force approach for evaluating this matrix and solving (5) will be prohibitively expensive. The dimension of E_{XX} is n. However, solving the Sylvester equation (8) would amount to solving a linear system with dimension $nk \times nk$. In this section, we will show that both $J^{(\ell)}$ and E_{XX} have special structures that can be exploited to speed up the computation.

3.1 The Jacobian matrix of $R(X, \Lambda)$

The Jacobian matrix J of $R(X, \Lambda)$ consists of the partial derivative of each entry in $R(X, \Lambda)$ with respect to elements of X and Λ. We partition J as

$$J = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix}, \tag{11}$$

where J_{11} denotes the partial derivative of F with respect to X, J_{12} the partial derivative of F with respect to Λ, J_{21} the partial derivative of G with respect to X. The J_{22} block which is partial derivative of G with respect to Λ is clearly zero.

We will now examine the structure of J_{11}, J_{12} and J_{21}. Note that the precise structures of these matrices depend on how elements of X and Λ are organized into a vector respectively.

In the following, we will rearrange the elements of X and by stacking columns of X into a single vector denoted by $\mathrm{vec}(X)$, i.e.,

$$\mathrm{vec}(X) = (x_1^T\ x_2^T\ \ldots\ x_k^T)^T.$$

The same column-major ordering scheme is used for both $H(X)X$ and Λ.

3.1.1 The structure of J_{11}

The function $\mathrm{vec}(F(X,\Lambda)X) = \mathrm{vec}(LX + \mathrm{Diag}(L^\dagger\rho)X - X\Lambda)$ is the sum of three terms. It is easy to verify that the derivative of the first term, $\mathrm{vec}(LX)$, can be expressed succinctly by

$$I_k \otimes L, \tag{12}$$

where I_k is a $k \times k$ identity matrix, and $I_k \otimes L$ denotes the *Kronecker* product of I_k and L.

Similarly, the partial derivative of the third term, $\mathrm{vec}(X\Lambda)$, with respect to $\mathrm{vec}(X)$ can be easily shown to be

$$\Lambda \otimes I_n. \tag{13}$$

Because the second term of $\mathrm{vec}(F)$ is nonlinear in X, its derivative has a more complex expression. Let us first examine the case in which X contains a single column, i.e., $X = (x_1)$. In this case, the partial derivative of $\mathrm{Diag}(L^\dagger\rho)x_1$ with respect to x_1 can be derived in a straightforward manner. Using the product rule and the observation that $\mathrm{Diag}(L^\dagger\rho)x_1 = \mathrm{Diag}(x_1)L^\dagger\rho$, we obtain

$$\begin{aligned}
\frac{\partial \mathrm{Diag}(L^\dagger\rho)x_1}{\partial x_1} &= \mathrm{Diag}(L^\dagger\rho) + \mathrm{Diag}(x_1)\frac{\partial(L^\dagger\rho)}{\partial x_1} \\
&= \mathrm{Diag}(L^\dagger\rho) + \mathrm{Diag}(x_1)\frac{\partial(L^\dagger\rho)}{\partial\rho}\frac{\partial\rho}{\partial x_1} \\
&= \mathrm{Diag}(L^\dagger\rho) + 2\mathrm{Diag}(x_1)L^\dagger\mathrm{Diag}(x_1) \\
&= \mathrm{Diag}(L^\dagger\rho) + 2L^\dagger \odot (x_1 x_1^T),
\end{aligned}$$

where $\odot$ is used to denote the Hadamard product.

The technique used above can be extended to the case in which X contains k columns. Let $e = (1, 1, ..., 1)^T$ be a vector with k ones, and define $\mathrm{Diag}(X) \equiv (\mathrm{Diag}(x_1)\ \mathrm{Diag}(x_2) \cdots \mathrm{Diag}(x_k))$. It is not difficult to verify that

$$\begin{aligned}
\frac{\partial \mathrm{vec}(\mathrm{Diag}(L^\dagger\rho)X)}{\partial \mathrm{vec}(X)} &= I_k \otimes \mathrm{Diag}(L^\dagger\rho) + \mathrm{Diag}(\mathrm{vec}(X))\frac{\partial(e \otimes (L^\dagger\rho))}{\partial \mathrm{vec}(X)} \\
&= I_k \otimes \mathrm{Diag}(L^\dagger\rho) + \mathrm{Diag}(\mathrm{vec}(X))\frac{\partial(e \otimes L^\dagger\rho)}{\partial\rho}\frac{\partial\rho}{\partial \mathrm{vec}(X)} \\
&= I_k \otimes \mathrm{Diag}(L^\dagger\rho) + 2\mathrm{Diag}(\mathrm{vec}(X))(e \otimes L^\dagger)\mathrm{Diag}(X) \\
&= I_k \otimes \mathrm{Diag}(L^\dagger\rho) + 2[(ee^T) \otimes L^\dagger] \odot (\mathrm{vec}(X)\mathrm{vec}(X)^T).
\end{aligned} \tag{14}$$

Similarly, we can show that

$$\frac{\partial \mathrm{vec}(\mathrm{Diag}(\rho^{1/3})X)}{\partial \mathrm{vec}(X)} = \frac{2}{3}[(ee^T) \otimes \mathrm{Diag}(\rho^{1/3})] \odot (\mathrm{vec}(X)\mathrm{vec}(X)^T). \tag{15}$$

It follows from (12), (13), (14) and (15) that

$$\begin{aligned}
J_{11} &= I_k \otimes [L + \mathrm{Diag}(L^\dagger\rho - \rho^{1/3})] \\
&+ 2[(ee^T) \otimes (L^\dagger - \frac{\gamma}{3}\mathrm{Diag}(\rho^{-2/3}))] \odot (\mathrm{vec}(X)\mathrm{vec}(X)^T) - \Lambda \otimes I_n.
\end{aligned} \tag{16}$$

Note that the Hessian of the total energy function E_{XX} has a similar structure to that of J_{11} with the exception that it does not contain the term involving Λ, i.e.,

$$E_{XX} = I_k \otimes [L + \mathrm{Diag}(L^\dagger \rho)] + 2[(ee^T) \otimes (L^\dagger - \frac{\gamma}{3}\mathrm{Diag}(\rho^{-2/3}))] \odot (\mathrm{vec}(X)\mathrm{vec}(X)^T).$$

3.2 The Structure of J_{21} and J_{12}

Because the only term in $F(X, \Lambda)$ that contains Λ is $X\Lambda$, its partial derivative with respect to Λ contains elements that are linear in X. Let us first ignore the symmetry of Λ. It is then easy to verify that

$$J_{12} = \frac{\partial \mathrm{vec}(X\Lambda)}{\partial \mathrm{vec}(\Lambda)} = I_k \otimes X,$$

which is a block diagonal matrix with X on each diagonal block. To take the symmetry of Λ into account, we can simply remove the leading $j-1$ columns corresponding to the jth diagonal block of $I_k \otimes X$ for $j = 2, ..., k$.

Similarly, it is not difficult to verify that

$$J_{21} = \frac{\partial \mathrm{vec}(X^T X)}{\partial \mathrm{vec}(X)} = I_k \otimes X^T + X^T \otimes I_k$$

The redundancy introduced by the symmetry of $X^T X$ can be eliminated by removing the leading $j-1$ rows corresponding to the jth diagonal block of J_{12} for $j = 2, ..., k$.

4 Search Direction Computation

We will first discuss how to compute the search direction Z in the standard Newton's method when it is applied to (5).

Because the dimension of J is generally very large, it is necessary to solve (6) by an iterative method.

4.1 Jacobian vector multiplication

When a Krylov subspace method is used to find an approximate solution to (6), we must develop an efficient procedure to carry out the matrix vector multiplication $\mathrm{vec}(W) \leftarrow J\mathrm{vec}(V)$, where W and V are both $(n+1)k$ by k, with as few floating point operations as possible. If we partition W and V conformally with J, i.e., if we let

$$W = \begin{pmatrix} W_X \\ W_\Lambda \end{pmatrix}, \quad V = \begin{pmatrix} V_X \\ V_\Lambda \end{pmatrix},$$

then it is easy to see that

$$\begin{pmatrix} \mathrm{vec}(W_X) \\ \mathrm{vec}(W_Z) \end{pmatrix} = \begin{pmatrix} J_{11}\mathrm{vec}(V_X) + J_{12}\mathrm{vec}(V_Z) \\ J_{21}\mathrm{vec}(V_X) \end{pmatrix}.$$

Because J_{12} and J_{21} are both very sparse the products $J_{12}\mathrm{vec}(V_Z)$ and $J_{21}\mathrm{vec}(V_X)$ can be computed efficiently.

The expression for J_{11} shown in (16) indicates that this $nk \times nk$ matrix is completely dense due to the presence of the term

$$2[(ee^T) \otimes (L^\dagger - \frac{\gamma}{3}\mathrm{Diag}(\rho^{-2/3})] \odot (\mathrm{vec}(X)\mathrm{vec}(X)^T). \tag{17}$$

Hence, one may be concerned about the feasibility of computing the product $J_{11}\mathrm{vec}(V_X)$ at a low cost. However, the following lemma suggests that such concern is unnecessary

because (17) turns out to have a low rank structure that allows $J_{11}\text{vec}(V_X)$ to be evaluated at a complexity proportional to that of evaluating k matrix vector multiplications $L^\dagger y_j$, for some $y_j \in \mathbb{R}^n$, $j = 1, 2, ..., k$.

Lemma 1 *Let $S \in \mathbb{R}^{n\times n}$, $X \in \mathbb{R}^{n\times k}$, and $e \in \mathbb{R}^k$ be a vector of all ones. Then the following identity holds*

$$\left[\left((ee^T)\otimes S\right)\odot\left(\text{vec}(X)\text{vec}(X)^T\right)\right]\text{vec}(Y) = \left[\left((ee^T)\otimes S\right)\left(\text{vec}(X)\odot\text{vec}(Y)\right)\right]\odot\text{vec}(X). \tag{18}$$

The identity can be easily derived from a straightforward extension of the simpler identity

$$\Big[S\odot(xy^T)\Big]z = \Big[S(y\odot z)\Big]\odot x, \tag{19}$$

where $x, y, z \in \mathbb{R}^n$.

Lemma 1 suggests that there is no need to form the matrix

$$S = \left((e_k e_k^T)\otimes(L^\dagger + \text{Diag}(\rho^{-2/3})\right)\odot\left(\text{vec}(X)\text{vec}(X)^T\right) \tag{20}$$

explicitly in order to carry out the matrix vector multiplication $J_{11}\text{vec}(V_X)$. When S is defined as in (20). The right hand side of (18) can be alternatively expressed as

$$\text{vec}[\text{Diag}(w)X],$$

where

$$w = \Big[L^\dagger + \text{Diag}(\rho^{-2/3})\Big]\sum_{i=1}^{k}(x_i\odot y_i) = \Big[L^\dagger + \text{diag}(\rho^{-2/3})\Big]\text{Diag}(XY^T).$$

Therefore, instead of performing $\mathcal{O}(n^2k^2)$ floating operations, as the left hand side of (18) would indicate, we only need to perform $k+1$ matrix vector multiplications of the form $L^\dagger z$, k matrix vector multiplications of the form Lz, as well as $\mathcal{O}(k)$ additional pointwise vector multiplication of the form $w\odot x_i$ to complete the computation of $J_{11}\text{vec}(X)$. Furthermore, because $L^\dagger$ is the inverse or pseudoinverse of the discretized Laplacian L, $L^\dagger z_i$ can be carried out using multigrid or fast convolution which has a computational complexity of $\mathcal{O}(n\log(n))$. Hence, the computational cost associated with $J_{11}\text{vec}(X)$ is at most $\mathcal{O}(n\log(n)k)$. This is comparable to the cost associated with the Hamiltonian matrix-vector multiplications $(H(X)Y)$ used in each iteration of an iterative eigensolver typically employed in the SCF iteration.

4.2 Choosing the forcing term

One of the practical issues one must address when using an iterative method to solve the Newton correction equation is the choice of a stopping criterion. A commonly used criterion is to terminate the solver when

$$\|J^{(\ell)}Z^{(\ell)} + R(X^{(\ell)},\Lambda^{(\ell)})\|_F < \eta_\ell\|R(X^{(\ell)},\Lambda^{(\ell)})\|_F,$$

for some appropriately chosen η_ℓ. The parameter η_ℓ is often referred to as the *forcing term*. When $(X^{(\ell)},\Lambda^{(\ell)})$ is far from the solution of (1), solving the Newton correction equation to high accuracy by setting η_ℓ to a small value is usually not necessary. To reduce the total computational cost, one may use an adaptive scheme proposed by Eisenstat and Walker in

[6] to gradually reduce η_ℓ as $(X^{(\ell)}, \Lambda^{(\ell)})$ move closer to the solution of (1). In particular, one may choose η_ℓ as

$$\eta_\ell = \sigma \frac{\|R(X^{(\ell)}, \Lambda^{(\ell)})\|^2}{\|R(X^{(\ell)}, \Lambda^{(\ell)})\|^2},$$

where $\sigma \in (0, 1]$. Safeguards must be placed on η_ℓ to prevent it from becoming either too close to 1 or too small in the early iterations of the Newton's method. We will refer readers to [6] for details.

5 Line Search

When the initial guess for the Newton's method is not sufficiently close to the solution of (1), taking a full Newton step, i.e., setting $\beta = 1$ in (7) may lead to a significant increase in the residual norm $\|R(X, \Lambda)\|_F$ and/or total energy $E(X)$. To overcome this problem and achieve global convergence, we may, at each Newton iteration, perform a line search procedure that judiciously chooses an appropriate step size β. A number of line search techniques can be used [8, 7, 12] here. These techniques require β to satisfy either the Armijo rule [1]

$$M(X^{(\ell)} + \beta \Delta X^{(\ell)}, \Lambda^{(\ell)} + \beta \Delta \Lambda^{(\ell)}) - M(X, \Lambda^{(\ell)}) < \sigma_1 \beta \nabla M(X^{(\ell)}, \Lambda^{(\ell)})^T Z^{(\ell)} \tag{21}$$

for some small $\sigma_1 \in (0, 1)$ or the Wolfe conditions that include both (21) and the condition that

$$\nabla M(X^{(\ell)} \beta \Delta X^{(\ell)}, \Lambda^{(\ell)} + \beta \Lambda^{(\ell)})^T Z \geq \sigma_2 \nabla M(X^{(\ell)}, \Lambda^{(\ell)})^T Z^{(\ell)},$$

where $M(X, \Lambda)$ is an appropriate merit function, and $Z^{(\ell)}$ is used above to represent the vectorized correction $(\Delta X^{(\ell)}, \Delta \Lambda^{(\ell)})$. When the Newton's method is applied to (1) directly, $\|R(X, \Lambda)\|^2$ can be used as the merit function $M(X, \Lambda)$. When an augmented Lagrangian method or an interior point method is used to solve (4) as a constrained minimization, a different merit function should be used [2, 12, 3]. In our numerical experiments, we have also tried the line search technique proposed in [9]. The main feature of this technique is that it may accept a step length even when the merit function temporarily increases. However, it seems to be more robust than other line search algorithms in terms of achieving global convergence in our experience.

Global convergence of the Newton's method can also be enhanced by using the trust region technique [4]. However, since our main interest in this paper is not the global convergence of the Newton's method, we will not discuss this technique further but simply refer readers to standard literature on this subject [4].

6 Numerical Examples

In this section, we demonstrate the convergence of the Newton's method with a few numerical examples. In these examples, we choose L as the second-order finite difference approximation to the 3-D Laplacian obtained from the standard 5-point central difference stencil. The Laplacian operator is defined on the domain $[0, 32] \times [0, 32] \times [0, 32]$ with zero Dirichlet boundary condition. We use mesh size $h = 1$ for the discretization. As a result, the dimension of the discretized problem is $n = 32^3 = 32768$. We set the constant γ in (3) to 1.0, which is an arbitrary choice. Our experiments indicate that other choices of γ yield similar convergence behavior. We use MINRES [13] as the iterative solver of the correction equation (6). Convergence of the Newton's method is declared when

$$\|R(X^{(\ell)}, \Lambda^{(\ell)})\| < 10^{-10},$$

where the residual $R(X, \Lambda)$ is defined in (5). All experiments are performed using MATLAB.

6.1 Local Quadratic Convergence

In Figure 1, we plot the change of the residual norm $\|R(X^{(\ell)}, X^{(\ell)})\|$ against the iteration number ℓ. In this experiment, we set $k = 4$, i.e., we compute the 4 smallest eigenvalues of the Hamiltonian (3). The initial guess for the Newton iteration, $X^{(0)}$, is obtained from running two SCF iterations. A random starting guess is used in the SCF iteration. We set the maximum MINRES iterations allowed in each Newton iteration to 100. The MINRES convergence tolerance is set to $\eta_\ell = 10^{-12}$, i.e. the MINRES iteration is terminated when either

$$\|J^{(\ell)}Z^{(\ell)} + R(X^{(\ell)}, \Lambda^{(\ell)})\| < 10^{-12}\|R(X^{(\ell)}, \Lambda^{(\ell)})\|, \tag{22}$$

or when the maximum MINRES iterations allowed is reached. In our experiment, the tight convergence tolerance set in (22) was never satisfied. The best relative residual norm returned from MINRES is around 2×10^{-7}. Nonetheless, the quadratic convergence of the Newton's method is quite clear from the rapid decrease of the residual norm shown in Figure 1.

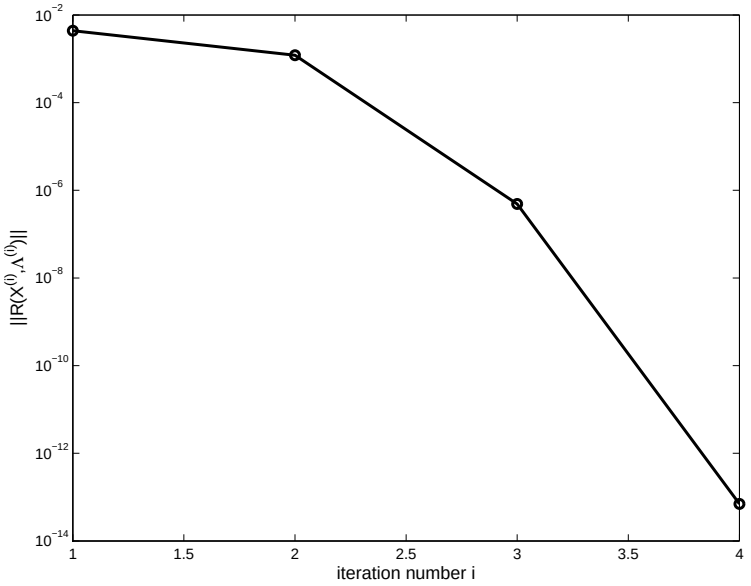

Figure 1: Quadratic convergence of the Newton's method for $n = 32^3$, $k = 4$.

6.2 Global convergence

Although the Newton's method is not guaranteed to converge when the initial guess is far away from the exact solution of (1), our experiments indicate that in many cases, convergence can be observed even when an random initial guess is chosen. In Figure 2, we plot the convergence history of the Newton's method that uses an random initial guess. To save time in our experiment, the problem we used to test the global convergence is smaller. In this experiment, we discretized the Laplacian on a $16 \times 16 \times 16$ grid. Four eigenpairs are computed. We set the maximum MINRES iterations allowed in each Newton iteration to 100 also. The MINRES convergence tolerance is set according to the Eisenstat & Walker forcing term selection rule discussed in section 4.2. Because we use the nonmonotone line search technique proposed in [9], the changes of residual norm in the early iterations are not monotonic. However, when $(X^{(\ell)}, \Lambda^{(\ell)})$ is sufficiently close to the solution of (1), quadratic convergence of $(X^{(\ell)}, \Lambda^{(\ell)})$ can be observed from Figure 2 also.

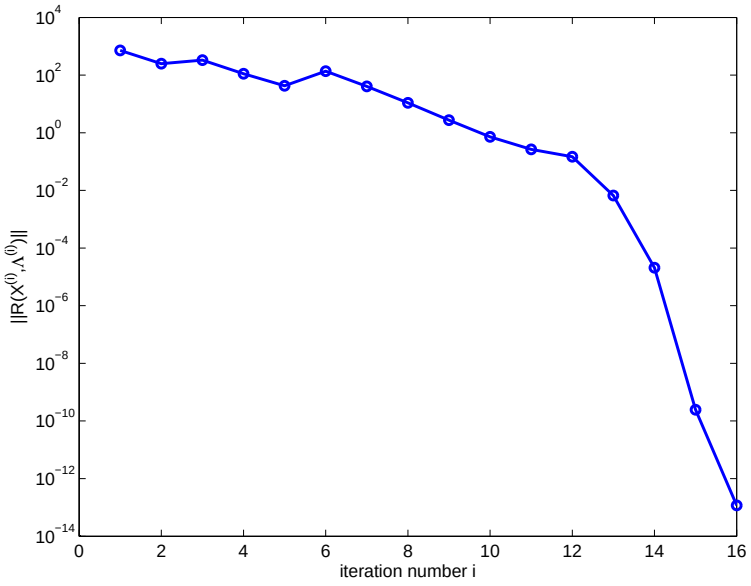

Figure 2: Global convergence of the Newton's method for $n = 10^3$, $k = 4$.

6.3 Constrained minimization

As we mentioned earlier, instead of solving (1) as a system of nonlinear equations, we can also try to solve the same problem as a constrained minimization problem with (4) being the objective function and $X^T X = I$ as the constraint. This can be done in a number of ways. For example, we can use the interior point algorithm proposed in [3]. A variant of the algorithm has been implemented in MATLAB Optimization Toolbox [11]. To use this particular implementation, we simply call the MATLAB `fmincon` function as follows

```
[X,RESNRM] = fmincon(@fetot, X0, [], [], [], [], [], [], @xnrm, ...
                     options, L, k, gamma);
```

where `fetot.m` is a MATLAB function that we wrote to evaluate the total energy defined in (4) and its gradient, `xnrm.m` is a function that we wrote to compute $X^T X - I$, and `L, k, gamma` are matrices and scalar parameters required for function and gradient calculations. The parameter `options` is a MATLAB structure that allows us to pass other parameters used by the `fmincon` function. In particular, it allows us to pass a function `HessX` that computes the product of the Hessian and an arbitrary vector. We define this structure as

```
options = optimset('Algorithm'  , 'interior-point',...
                   'Display'    , 'iter',...
                   'MaxIter'    , 5000  ,...
                   'MaxSQPIter' , 100   ,...
                   'MaxFunEvals', maxfun  ,...
                   'LargeScale' , 'on' ,...
                   'GradObj'    , 'on'  ,...
                   'GradConstr' , 'on'  ,...
                   'Hessian'    , 'on'  ,...
                   'SubproblemAlgorithm','cg',...
                   'HessMult'   , @HessX,...
                   'TolX'       , 1.e-10 ,...
                   'TolCon'     , 1.e-10 ,...
```

```
'TolFun'     , 1.e-10);
```

Figure 3 shows the convergence history of this particular version of the interior point algorithm which does not solve the barrier subproblem accurately at each step. It uses a combination of the sequential quadratic programming (SQP) procedure and the trust region technique to obtain approximate solutions to the barrier subproblems. The quadratic minimization problem in each SQP is solved iteratively using a preconditioned conjugate gradient method [3]. This approach seems to be quite robust. In our experiments, it converges for all the random initial guesses we tried. However, the convergence rate appears to be linear even when the approximation is close to the solution, as we can see from the residual norm plot shown in Figure 3. Our timing measurements also indicate that this particular approach is more expensive in terms of CPU time compared to applying Newton's method to (1) directly.

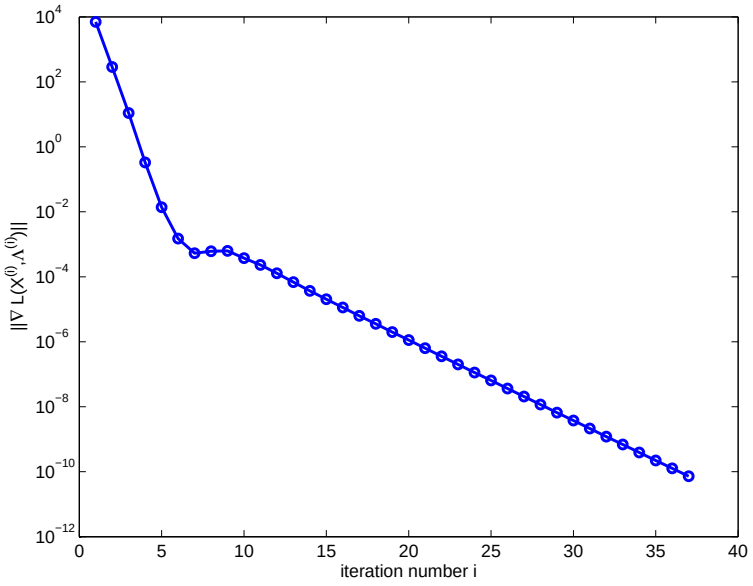

Figure 3: The interior-point algorithm implemented in the MATLAB `fmincon` function converges linearly when applied to a test problem with $n = 16^3$, $k = 4$.

6.4 Comparison with SCF

Instead of solving a sequence of linear eigenvalue problems as is done in SCF, applying Newton's method directly to (1) only requires solving a sequence of linear systems. The dimension of these linear equations is $m = nk + (k+1)k/2$, which is much larger than n, the dimension of the eigenvalue problem solved in SCF. However, as we indicated earlier, when an iterative solver such as MINRES is used to solve these systems, the matrix vector multiplication (MATVEC) can be performed with a complexity much less than m^2 due to the special structure of the Jacobian matrix. The most expensive part of the MATVEC is the computation required to perform $L^\dagger X$, where X is $n \times k$. Because L is block circulant in many cases (e.g., for periodic problems), $L^\dagger X$ can be computed efficiently by using the fast Fourier transform. The complexity associated with this computation is $\mathcal{O}(nk \log(n))$. In contrast, the SCF iteration only requires to perform one $L^\dagger \rho$ per iteration, where ρ is a vector of length n. However, the iterative procedure used to compute the smallest eigenvalues of the Hamiltonian $H(X^{(\ell)})$ defined by (3) requires repeated computation of

$H(X^{(\ell)})Y$, where Y is a $n \times k$. The complexity of this computation is also $\mathcal{O}(nk\log(n))$ when planewave discretization is used. So if the number of iterations used to solve each eigenvalue problem in the SCF iteration is comparable to the number of MINRES iterations used to solve the Newton correction equation, the Newton's method can be more efficient than the SCF iteration.

A timing comparison between the Newton's method and the SCF iteration is reported in Table 1 for the test problems we considered in this section. We vary both the problem size n and the number of desired eigenvalues k in our comparison. Each entry in Table 1 corresponds to the ratio $T_{Newton}(n,k)/T_{SCF}(n,k)$, where $T_{Newton}(n,k)$ is the total CPU time required by the Newton's method to compute k eigenpairs of a test problem of dimension n, and $T_{SCF}(n,k)$ is the CPU time required to solve the same problem by the SCF iteration. The initial guess used in each Newton test is obtained from running two SCF iterations (starting from a random initial guess.) Table 1 shows that the Newton's

(n,k)	1	4	10
10^3	0.77	0.83	1.25
16^3	0.98	1.08	2.35
32^3	2.14	1.81	3.0

Table 1: Timing comparison between the Newton's method and the SCF iteration.

method can outperform the SCF iteration when n and k are relatively small, and when a good initial guess is available. For large n and k, the Newton's method is currently not competitive with the SCF iteration. This is primarily due to the large number of MINRES iterations required to solve the Newton correction equation.

7 Conclusion

We investigated the possibility of applying the standard Newton's method to a class of nonlinear eigenvalue problems arising from electronic structure calculations. When viewed as a system of nonlinear equations, this type of problem has a very large and dense Jacobian matrix. However, the Jacobian matrix has certain low rank structure that allows us to solve the Newton correction equation efficiently using an iterative solver. We described the structure of the Jacobian matrix and illustrated how the product of the Jacobian matrix and a vector can be performed efficiently. A number of numerical examples were presented to demonstrate the local quadratic convergence of the Newton's method when applied to this type of problem. We also showed global convergence can be achieved when proper line search procedure and a judicious choice of forcing terms are used in the iterative solver. We compared the performance the Newton's method and the SCF iterations. For small problems in which a few eigenpairs are required, the Newton's method can outperform the SCF iteration when a good starting guess is available. For large problems, it is currently not as competitive. We believe one way to further improve the efficiency of the Newton's method is to identify a good preconditioner for the iterative solver used to solve the Newton correction equation.

References

[1] L. Armijo. Minimization of functions having Lipschitz-continuous first partial derivatives. *Pacific J. Math.*, 16:1–3, 1966.

[2] D. P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods.* Academic Press, New York, 1982.

[3] R. H. Bryd, M. E. Hbribar, and J. Nocedal. An interior point algorithm for large-scale nonlinear programming. *SIAM J. Optim.*, 9:877–900, 1999.

[4] A. R. Conn, N. I. M. Gould, and P. L. Toint. *Trust-Region Methods*. SIAM, Philadelphia, PA, 2000.

[5] A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Anal. Appl.*, 20(2):303–353, 1998.

[6] S. C. Eisenstat and H. F. Walker. Choosing the forcing terms in an inexact newton method. *SIAM J. Sci. Comput.*, 17(1):16–32, 1996.

[7] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, 1987.

[8] P. E. Gill, W. Marray, and M. H. Wright. *Practical Optimization*. Academic Press, London, 1981.

[9] F. Lampariello L. Grippo and S. Lucidi. A nonmonotone line search technique for Newton's method. *SIAM J. Numer. Anal.*, 23(4):707–716, 1986.

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

[11] Inc. MathWorks. *MATLAB Optimization Toolbox User's Guide*. MathWorks, 1984–2009.

[12] J. Nocedal and S. Wright. *Numerical Optimization*. Springer-Verlag, New York, 1999.

[13] C. Paige and M. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12:617–629, 1975.

[14] A. Szabo and N. S. Ostlund. *Modern Quantum Chemistry: An Introduction to Advanced Electronic Structure Theory*. Dover, 1996.

[15] C. Yang, J. C. Meza, and L. W. Wang. A constrained optimization algorithm for total energy minimization in electronic structure calculation. *Journal of Computational Physics*, pages 709–721, 2006.