Title

Efficient block preconditioned eigensolvers for linear response time-dependent density functional theory

Authors

Vecharynski, Eugene
Brabec, Jiri
Shao, Meiyue
et al.

Publication Date

2017-12-01

DOI

10.1016/j.cpc.2017.07.017

Peer reviewed

Efficient Block Preconditioned Eigensolvers for Linear Response Time-dependent Density Functional Theory

Eugene Vecharynski,*,† Jiri Brabec,*,† Meiyue Shao,*,† Niranjan Govind,*,‡ and Chao Yang*,†

Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720 , and Environmental Molecular Sciences Laboratory, Pacific Northwest National Laboratory, Richland, WA 99352

E-mail: evecharynski@lbl.gov; jbrabec@lbl.gov; myshao@lbl.gov; niri.govind@pnnl.gov; cyang@lbl.gov

Abstract

We present two efficient iterative algorithms for solving the linear response eigenvalue problem arising from the time dependent density functional theory. Although the matrix to be diagonalized is nonsymmetric, it has a special structure that can be exploited to save both memory and floating point operations. In particular, the nonsymmetric eigenvalue problem can be transformed into a product eigenvalue problem that is self-adjoint with respect to a K-inner product. This product eigenvalue problem can be solved efficiently by a modified Davidson algorithm and a modified

*To whom correspondence should be addressed
†Lawrence Berkeley National Laboratory
‡Pacific Northwest National Laboratory

locally optimal block preconditioned conjugate gradient (LOBPCG) algorithm that make use of the K-inner product. The solution of the product eigenvalue problem yields one component of the eigenvector associated with the original eigenvalue problem. However, the other component of the eigenvector can be easily recovered in a postprocessing procedure. Therefore, the algorithms we present here are more efficient than existing algorithms that try to approximate both components of the eigenvectors simultaneously. The efficiency of the new algorithms is demonstrated by numerical examples.

1 Introduction

Within the linear response (LR) framework of the time-dependent density functional theory (TDDFT), the absorption spectrum of a molecule can be estimated by solving an eigenvalue problem of the form

$$\begin{bmatrix} A & B \\ -B & -A \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \lambda \begin{bmatrix} u \\ v \end{bmatrix}, \tag{1}$$

where A and B are $n \times n$ real symmetric matrices. If we use n_o and n_v to denote the number of occupied and virtual states of the ground state Kohn–Sham Hamiltonian, respectively, then the dimension of A and B is $n = n_o n_v$, which can be extremely large. In addition, A and B are often not stored explicitly. They are available through a procedure that multiplies A and B with a vector or a block of vectors.

In most cases, $A - B$ and $A + B$ are both positive definite. Although the matrix in (1) is nonsymmetric in general, it has a special structure which ensures that its eigenvalues are real and come in positive and negative pairs. Moreover, it can be shown (see, e.g.,[1]) that a structured eigendecomposition of this matrix is given by

$$\begin{bmatrix} A & B \\ -B & -A \end{bmatrix} = \begin{bmatrix} U & V \\ V & U \end{bmatrix} \begin{bmatrix} \Lambda & 0 \\ 0 & -\Lambda \end{bmatrix} \begin{bmatrix} U & -V \\ -V & U \end{bmatrix}^*$$

where $\Lambda = \operatorname{diag}\{\lambda_1, \ldots, \lambda_n\}$ consists of the positive eigenvalues, and the eigenvectors are normalized by

$$\begin{bmatrix} U & -V \\ -V & U \end{bmatrix}^* \begin{bmatrix} U & V \\ V & U \end{bmatrix} = I. \tag{2}$$

From the eigendecomposition, we immediately obtain that if $\lambda > 0$ is an eigenvalue with corresponding right eigenvector $[u^T, v^T]^T$, then the corresponding left eigenvector is $[u^T, -v^T]^T$; in addition, the right and left eigenvectors of $-\lambda$ are $[v^T, u^T]^T$ and $[-v^T, u^T]^T$, respectively.

The positive eigenvalues of (1) correspond to absorption energies of a molecular system. The corresponding eigenvectors are related to the likelihood of these excitations. If λ_i is the ith positive eigenvalue of (1) sorted in increasing order, associated with an eigenvector $[u_i^T, -v_i^T]^T$, then the absorption spectrum of the molecular system is defined by

$$\sigma(\omega) = \sum_{i=1}^{n} [\hat{d}^T (u_i + v_i)]^2 \delta(\omega - \lambda_i), \tag{3}$$

where $d = [\hat{d}^T, \hat{d}^T]^T$ is the dipole vector and $\delta(\cdot)$ is the Dirac-delta function. In (3), the eigenvectors are required to be normalized as $u_i^T u_i - v_i^T v_i = 1$, which is consistent with the condition (2).

In many applications, only the lowest excitation levels up to energy cutoff are of interest. Therefore, we only need to compute a handful of the smallest positive eigenvalues of (1). In this case, it is not efficient to use dense diagonalization methods, such as the ones available in the ScaLAPACK[2] software package to compute these eigenpairs. Instead, iterative methods that only require multiplying A and B with vectors are more attractive. When one is only interested in the overall shape of the absorption spectrum instead of precise values of the excitation energies and their corresponding oscillator strengths, it is also possible to use iterative methods that do not compute any eigenvalue or eigenvector explicitly to obtain a rough estimate of the absorption spectrum.[3]

Iterative methods for solving (1) have been developed in the last few decades.[4–8] Some

of them have been implemented in modern quantum chemistry software packages.[9] Many of these algorithms make use of the observation that the eigenvalue problem (1) can be transformed via a unitary similarity

$$J = \frac{1}{\sqrt{2}} \begin{bmatrix} I_n & I_n \\ I_n & -I_n \end{bmatrix}$$

to

$$\begin{bmatrix} 0 & K \\ M & 0 \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} = \lambda \begin{bmatrix} y \\ x \end{bmatrix}, \tag{4}$$

where $K = A - B$ and $M = A + B$,[10] respectively. The eigenvalues of (4) are exactly the same as those of (1). The eigenvectors of these two problems are related by

$$u = \frac{1}{\sqrt{2}}(y + x), \qquad v = \frac{1}{\sqrt{2}}(y - x).$$

The paired eigenvalues $\pm\lambda$ of (4) have the corresponding eigenvectors $[\pm y^T, x^T]^T$.

It follows from (4) that solving (1) is equivalent to solving a product eigenvalue problem of the form

$$MKx = \lambda^2 x, \tag{5}$$

or

$$KMy = \lambda^2 y. \tag{6}$$

Each eigenpair (λ^2, x) of (5) yields two eigenvalues $\pm\lambda$ of (4), with the corresponding eigenvectors $[\pm y^T, x^T]^T$, where

$$y = \lambda^{-1} K x, \tag{7}$$

for $\lambda \neq 0$. Similarly, the eigenpairs (λ^2, y) of (6) define the eigenpairs of (4), with $x = \lambda^{-1} M y$.

In this paper, we focus on the product formulation of the eigenvalue problem (5). It is important to recognize that the y component of the eigenvector of (4) can be easily recovered

in a postprocessing procedure once the x component, which can be obtained by solving (5), is available. That is, it is not necessary to try to approximate x and y simultaneously in an iterative procedure for solving (4) or (1). By only focusing on the solution to (5) or (6), we can achieve significant savings in both the number of operations and storage. To solve (5) efficiently, we use the observation that MK is self-adjoint with respect to the K-inner product and modify two widely used algorithms for solving standard symmetric eigenvalue problems by simply replacing the standard Euclidean inner product with the K-inner product. We discuss how to construct preconditioners for these algorithms that use K-inner product to generate search subspaces and extract approximate eigenpairs from these subspaces. We show that these algorithms use fewer matrix vector multiplications and have a smaller storage requirement compared to existing algorithms. Our computational experiments indicate that the algorithms we propose in this paper are more efficient than existing approaches for a number of test problems.

The paper is organized as follows. In Section 2, we review two state-of-the-art techniques for solving the LR eigenvalue problem (1). We present our K-inner product based algorithms in Section 3. Issues regarding practical implementation are discussed in Section 4. Computational results are reported in Section 5 to illustrate and compare the efficiency of our proposed new algorithms with existing algorithms.

2 Existing algorithms

We start by discussing several existing approaches for solving the eigenvalue problem (1).

2.1 The Davidson algorithm

One of the widely used approaches is based on a modification of the Davidson's algorithm[11] for solving symmetric eigenvalue problems. It was presented in,[8] and is currently implemented and used in many quantum chemistry software packages.[9,12–14]

The basic idea behind the Davidson algorithm[8] is to extract approximations to both the x and y components of the k desired eigenvectors of (4) from a subspace spanned by columns of a matrix S that is constructed iteratively. To be specific, the x and y components are written as

$$X = S\hat{X} \quad \text{and} \quad Y = S\hat{Y},$$

respectively, where $S^T S = I$ and the columns of

$$\begin{bmatrix} Y \\ X \end{bmatrix} = \begin{bmatrix} S & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} \hat{Y} \\ \hat{X} \end{bmatrix}$$

form the approximations to the k desired eigenvectors of (4). It then follows from the Galerkin condition

$$\begin{bmatrix} S & 0 \\ 0 & S \end{bmatrix}^T \left(\begin{bmatrix} 0 & K \\ M & 0 \end{bmatrix} \begin{bmatrix} Y \\ X \end{bmatrix} - \begin{bmatrix} Y \\ X \end{bmatrix} \Theta \right) = 0$$

that the unknowns $\hat{X}$, $\hat{Y}$, and Θ can be obtained by solving the reduced eigenvalue problem

$$\begin{bmatrix} 0 & \hat{K} \\ \hat{M} & 0 \end{bmatrix} \begin{bmatrix} \hat{Y} \\ \hat{X} \end{bmatrix} = \begin{bmatrix} \hat{Y} \\ \hat{X} \end{bmatrix} \Theta, \tag{8}$$

where $\hat{K} = S^T K S$ and $\hat{M} = S^T M S$. Note that (8) preserves the structure of (4). The solution to (8) can be obtained by first transforming (8) into the product form

$$\hat{M}\hat{K}\hat{X} = \hat{X}\Theta^2, \tag{9}$$

and then symmetrizing the problem by multiplying (9) from the left by $\hat{K}^{1/2}$ to yield

$$\hat{K}^{1/2}\hat{M}\hat{K}^{1/2}Q = Q\Theta^2, \tag{10}$$

where $Q = \hat{K}^{-1/2}\hat{X}$. Once the k eigenvalues of (10), which appear on the diagonal of Θ^2,

and the corresponding eigenvectors Q have been computed, approximation to the desired eigenvectors of (9) are obtained by $\hat{X} = \hat{K}^{1/2}Q$. It follows from (7) that $\hat{Y} = \hat{K}\hat{X}\Theta^{-1}$ satisfies $\hat{K}\hat{M}\hat{Y} = \hat{Y}\Theta^2$.

The residuals associated with the x and y components of the approximations to the desired eigenvectors of (4) are

$$R_K = KS\hat{X} - \hat{Y}\Theta \quad \text{and} \quad R_M = MS\hat{Y} - \hat{X}\Theta.$$

If neither R_K nor R_M is small in norm, then both of these two matrices are used to expand S as

$$S \leftarrow \text{orth}\left\{S,\ T_K^{-1}R_K,\ T_M^{-1}R_M\right\},$$

where T_K and T_M are appropriately chosen preconditioners, and orth$\{X\}$ returns an orthonormal basis of X. Such an expansion allows constructing a search space that contains approximations to both the left and right eigenvectors of MK. This process is repeated either until the convergence is achieved or till the storage for S reaches its limit. In the latter case, the Davidson iteration can be restarted by simply replacing S with the current approximation to $\hat{X}$ and $\hat{Y}$. The major steps of the simplest version of such a Davidson algorithm are outlined in Algorithm 1.

We should emphasize that Algorithm 1 aims at solving the eigenvalue problem (4) in which the x and y components of the eigenvector are coupled. As a result, approximations to both x and y are generated, and two sets of residuals of the form $Ky - \theta x$ and $Mx - \theta y$ are computed and used to enlarge the subspace spanned by the columns of S. Therefore, the projection of K and M onto S in Step 3 of the algorithm requires multiplications of M and K with both $T_K^{-1}R_K$ and $T_M^{-1}R_M$.

As we shall see in the next section, it is not necessary to generate approximations to both x and y simultaneously because y can be easily recovered once x is available. This observation motivates the alternative algorithms to be presented in Section 3, which have a

Algorithm 1: The Davidson algorithm[8] for solving the linear response eigenvalue problem (4).

Input: Positive definite matrices K and M, preconditioners T_K and T_M, and a starting guess $\begin{bmatrix} Y^{(0)} \\ X^{(0)} \end{bmatrix}$.

Output: A diagonal matrix of k smallest positive eigenvalues Λ of (4), and the associated right eigenvectors $\begin{bmatrix} Y \\ X \end{bmatrix}$.

1: $X \leftarrow X^{(0)}$; $Y \leftarrow Y^0$; $S \leftarrow [X,\ Y]$;
2: **while** convergence not reached **do**
3: $S \leftarrow \text{orth}\{S\}$[a]; compute $\hat{K} \leftarrow S^T K S$ and $\hat{M} \leftarrow S^T M S$;
4: Compute k eigenpairs (Θ^2, Q) of (9) associated with the smallest positive eigenvalues; $\hat{Y} \leftarrow \hat{K}^{1/2} Q$; $\hat{X} \leftarrow \hat{K}^{-1/2} Q \Theta$; $\Lambda \leftarrow \Theta$;
5: $Y \leftarrow S\hat{Y}$; $X \leftarrow S\hat{X}$;
6: $R_K \leftarrow KX - Y\Lambda$; $R_M \leftarrow MY - X\Lambda$;
7: **for** $j = 1 \rightarrow k$ **do**
8: $\lambda \leftarrow \Lambda(j, j)$; $r_K \leftarrow R_K(:, j)$; $r_M \leftarrow R_M(:, j)$;
9: $r_K \leftarrow T_K^{-1} r_K$; $r_M \leftarrow T_M^{-1} r_M$;
10: $S \leftarrow [S, r_K, r_M]$;
11: **end for**
12: **end while**

[a] $\text{orth}\{S\}$ returns an orthonormal basis of the subspace spanned by the columns of S.

lower memory requirement and perform fewer multiplications with K and M.

2.2 Locally optimal block preconditioned conjugate gradient algorithms

It is well known that the k algebraically smallest eigenvalues of an $n \times n$ symmetric matrix A can be obtained by solving the trace minimization problem

$$\min_{X^T X = I} \text{trace}(X^T A X), \tag{11}$$

where X is of size $n \times k$; see, e.g.,.[15] If a good preconditioner for A is at hand, then (11) can be efficiently solved by the locally optimal block preconditioned conjugate gradient (LOBPCG) algorithm.[16]

Because the matrix in (1) is nonsymmetric and the desired eigenvalues are not the smallest, we cannot apply the trace minimization principle directly. However, it has been noticed in[17] that (4) can be recast as a constrained minimization problem in which the objective function is the Thouless functional

$$\varrho(x,y) = \frac{x^T K x + y^T M y}{2|x^T y|}. \tag{12}$$

The minimizer (x, y) of (12) yields the eigenvector associated with the smallest positive eigenvalue of (4); see.[17–19]

The minimization principle based on (12) was recently extended to subspaces.[10] It has been shown that the eigenvectors associated with the k smallest positive eigenvalues of (4) are determined by the two biorthogonal $n \times k$ matrices X and Y that minimize the trace functional

$$\frac{1}{2}\text{trace}(X^T K X + Y^T M Y), \qquad (\text{s.t. } X^T Y = I). \tag{13}$$

The x and y components of each eigenvector in (4) are then given by the corresponding columns of X and Y that minimize (13). In particular, this has allowed adapting the LOBPCG algorithm[16] to seek the minimizer of (13). One example of such a scheme is the LOBP4DCG algorithm presented in.[4] Another example is the recently developed indefinite LOBPCG (ILOBPCG) algorithm,[6] which is mathematically equivalent to LOBP4DCG when solving (4).

In contrast to the Davidson algorithm, LOBP4DCG does not expand the search subspace at every step. Instead, in the ith iteration, the LOBP4DCG algorithm updates the approximation to X and Y by minimizing (13) within a set of subspaces spanned by

$$\{X^{(i)},\ T_K R_K^{(i)},\ X^{(i-1)}\} \quad \text{and} \quad \{Y^{(i)},\ T_M R_M^{(i)},\ Y^{(i-1)}\}, \tag{14}$$

where $X^{(i)}$, $Y^{(i)}$ are the parts of the approximate eigenvectors at iteration i, and $R_K^{(i)}$, $R_M^{(i)}$ are

Algorithm 2: The LOBP4DCG algorithm[4] for solving the linear response eigenvalue problem (4).

Input: Positive definite matrices K and M, preconditioners T_K and T_M, a starting guess $\begin{bmatrix} Y^{(0)} \\ X^{(0)} \end{bmatrix}$.

Output: A $k \times k$ diagonal matrix Λ that contains the k smallest positive eigenvalues of (4), and the associated right eigenvectors $\begin{bmatrix} Y \\ X \end{bmatrix}$.

1: $X \leftarrow X^{(0)}$; $Y \leftarrow Y^{(0)}$; $P_M \leftarrow [\,]$; $P_K \leftarrow [\,]$;
2: $\Lambda(j,j) \leftarrow \varrho(X(:,j), Y(:,j)), \quad j = 1 \rightarrow k$;
3: **while** convergence not reached **do**
4: $\quad R_K \leftarrow KX - Y\Lambda$; $R_M \leftarrow MY - X\Lambda$;
5: $\quad V \leftarrow \mathrm{orth}\{Y, T_M^{-1}R_M, P_M\}$; $U \leftarrow \mathrm{orth}\{X, T_K^{-1}R_K, P_K\}$;
6: $\quad$ Compute nonsingular W_1 and W_2, such that $U^TV = W_1^TW_2$;
7: $\quad$ Compute k eigenpairs (Θ, G) of

$$\begin{bmatrix} 0 & W_1^{-T}U^TKUW_1^{-1} \\ W_2^{-T}V^TMVW_2^{-1} & 0 \end{bmatrix}$$

$\quad$ associated with the smallest positive eigenvalues, where the eigenvectors G are of the form $G = \begin{bmatrix} \hat{Y} \\ \hat{X} \end{bmatrix}$;
8: $\quad Y \leftarrow VW_2^{-1}\hat{Y}$; $X \leftarrow UW_1^{-1}\hat{X}$;
9: $\quad \Lambda(j,j) \leftarrow \varrho(X(:,j), Y(:,j)), \quad j = 1 \rightarrow k$;
10: **end while**

components of the projected gradient of (13) along the tangent of the constraint $X^TY = I$. The operators T_K and T_M are appropriately chosen preconditioners. A basic version of the LOBP4DCG algorithm is outlined in Algorithm 2.

Compared to the Davidson algorithm,[8] described in the previous section, LOBP4DCG performs only one multiplication of K with k vectors and one multiplication of M with k vectors in each iteration. On the other hand, the limited size of the search subspaces spanned by the columns of (14) may lead to a larger number of iterations to reach convergence. There are also a few practical issues, such as potential numerical instability that may arise in the biorthogonalization procedure (Steps 6 through 8 of Algorithm 2), one must address in order to make the algorithm robust and efficient. Similar issues can also appear in the ILOBPCG algorithm[6] because it uses an indefinite inner product that can potentially be numerically unstable.

3 New algorithms

In this section, we present two new algorithms that are based on the product formulation (5) of the LR eigenvalue problem. Our main observation is that, while nonsymmetric in the standard Euclidean inner product, the matrix MK is self-adjoint (symmetric) with respect to the K-inner product, that is,

$$\langle v, MKv\rangle_K = v^T KMKv = (MKv)^T Kv = \langle MKv, v\rangle_K,$$

where $\langle X, Y\rangle_K \equiv X^T KY$. Hence, we can solve the following constrained minimization problem

$$\min_{\langle X,X\rangle_K = I} \text{trace}\langle X, MKX\rangle_K, \tag{15}$$

which is a direct K-inner product based analogue of minimizing the trace functional (11) for computing eigenpairs of symmetric matrices. Note that the optimization in (15) is performed on the set of all $n \times k$ matrices whose columns are K-orthonormal.

The gradient of the Lagrangian $\mathcal{L}(X, \Lambda)$ associated with (15) with respect to X *in the K-inner product* is

$$\nabla_K \mathcal{L}(X, \Lambda) = 2(MKX - X\Lambda^2) \equiv 2R, \tag{16}$$

where Λ^2 is an $k \times k$ matrix of Lagrange multipliers. The first order optimality condition associated with (15) yields the eigenvalue problem

$$MKX = X\Lambda^2, \quad \text{and} \quad X^T KX = I.$$

Therefore, indeed minimization of the trace functional in (15) solves the product eigenvalue problem (5), and hence the problem (4). Note that Λ^2 may not necessarily be a diagonal matrix in (16). However, it can always be diagonalized through an orthogonal transformation. When Λ^2 is diagonal, columns of X contain the eigenvectors associated with the k smallest

eigenvalues of MK.

We should note that (15) is clearly equivalent to

$$\min_{X^TKX=I} \text{trace}(X^TKMKX). \tag{17}$$

The first order optimality condition associated with (17) *in the standard Euclidean inner product* yields a generalized eigenvalue problem

$$KMKX = KX\Lambda^2, \quad X^TKX = I. \tag{18}$$

One can use existing algorithms such as the Davidson algorithm or the LOBPCG algorithm to solve (17) or (18). However, these algorithms will in general perform one extra multiplication of K with a block of vectors unless a special preconditioner is used.

We should also point out that, because (7) holds, it is reasonable to impose the additional constraint $Y = KX\Theta$ for some Θ in the minimization of (13). It is not difficult to show that solving the constrained minimization problem

$$\min_{Y=KX\Theta, Y^TX=I} \text{trace}(X^TKX + Y^TMY) \tag{19}$$

is equivalent to solving (17).

3.1 The K-inner product Davidson algorithm

The use of the K-inner product in the objective function (15) does not change the way a subspace is constructed in a standard Davidson algorithm,[11,20] although, as we shall see, it will affect the eigenvector extraction phase of the algorithm. Starting from an initial guess of the desired eigenvectors $X^{(0)}$, we define a sequence of search spaces $S^{(i)}$ successively by

$$S^{(i)} \leftarrow \text{span}\{S^{(i-1)}, T^{-1}R^{(i)}\},$$

where $S^{(0)} = \text{span}\{X^{(0)}\}$, $R^{(i)}$ is the projected gradient associated with $X^{(i)}$ defined in (16), that is,

$$R^{(i)} = MKX^{(i)} - X^{(i)}\Lambda^{(i)2}, \tag{20}$$

where $\Lambda^{(i)2} = \langle X^{(i)}, MKX^{(i)}\rangle_K$, and T is a properly chosen preconditioner.

Since the geometry of problem (15) is based on the K-inner product, it is natural to use it for orthonormalizing the basis of the search subspace, i.e., at each step ensure that $\langle S^{(i)}, S^{(i)}\rangle_K = I$. Then projecting MK *with respect to the K-inner product* onto the subspace spanned by $S^{(i)}$ yields the reduced eigenvalue problem

$$\langle S^{(i)}, MKS^{(i)}\rangle_K C = C\Theta^2, \quad C^TC = I, \tag{21}$$

where Θ^2 is a $k \times k$ diagonal matrix containing k smallest eigenvalues of $\langle S^{(i)}, MKS^{(i)}\rangle_K$, whose associated eigenvectors are given by the orthonormal columns of the matrix C. As a result, the approximations to the targeted eigenvectors of MK are given by $S^{(i)}C$, whereas the approximate eigenvalues are determined directly by Θ^2. We summarize the basic steps of the algorithm, which we call K-Davidson, in Algorithm 3.

The choice of the preconditioner T plays a crucial role in achieving fast convergence for the K-Davidson algorithm. Because K and M are typically diagonally dominant for linear response TDDFT calculations, it is natural to choose T to be a diagonal matrix of the form $D_K D_M$, where D_K and D_M are diagonal matrices that are close to K and M, respectively. It is also possible to choose D_K and D_M to be the same, so that both are equal to some D. For example, we can set the diagonal entries of D to $\varepsilon_a - \varepsilon_j$, where ε_a and ε_j are unoccupied and occupied single-particle energies, respectively, at the ground state.

It is also possible to construct different preconditioners for different columns of $R^{(i)}$. In particular, we can apply a shifted preconditioner of the form $D_K D_M - \theta_j^{(i)2} I$ to the jth column of $R^{(i)}$, where $\theta_j^{(i)2}$ is the jth eigenvalue of the projected problem (21). This type of construction makes the Davidson algorithm behave more like the Jacobi–Davidson

Algorithm 3: The K-inner product Davidson (K-Davidson) algorithm

Input: Positive definite matrices K and M, a preconditioner T, a starting guess $X^{(0)}$.

Output: Approximate eigenvalues of (4) contained in a diagonal matrix Λ and the corresponding right eigenvector approximation $\begin{bmatrix} Y \\ X \end{bmatrix}$.

1: $X \leftarrow X^{(0)}$; $S \leftarrow [X]$;
2: **while** convergence not reached **do**
3: $S \leftarrow K$-orth$\{S\}$;[a]
4: Solve the projected eigenvalue problem (21) to obtain eigenvalue and eigenvector approximations (Θ^2, X)
5: $R \leftarrow MKX - X\Theta^2$;
6: $S \leftarrow \text{span}\{S, T^{-1}R\}$;
7: **end while**;
8: $\Lambda \leftarrow \Theta$;
9: $Y \leftarrow KX\Theta^{-1}$;

[a] K-orth$\{S\}$ returns an K-orthonormal basis of the subspace spanned by the columns of S.

algorithm[21] in which the approximation to each desired eigenvector is corrected by an inexact Newton iteration.

In contrast to Algorithm 1, the K-Davidson algorithm requires only k multiplications of K and M with vectors (that is, a total of $2k$ matrix–vector multiplications) per iteration. Because the cost of these multiplications dominates the overall computational cost, each step of the K-Davidson algorithm is two times faster than each step of Algorithm 1. Furthermore, the subspace S constructed in Algorithm 3 is expanded with only one set of the preconditioned residual vectors instead of two sets of vectors. Its memory cost is half of that of Algorithm 1 also. Note that the K-Davidson algorithm does not try to approximate the Y component of the eigenvectors in (4) directly in the iterative process. Approximation to this component can be obtained in a postprocessing step by using the relationship $Y = KX\Lambda^{-1}$ once approximation to the X component of the eigenvector is obtained. In fact, as we will discuss in Section 4, the vectors $KX^{(i)}$ are computed and stored as a by-product of the computation, and hence approximations $Y^{(i)}$ (up to a diagonal scaling) are readily available

at each iteration.

When the amount of available memory is limited, we may restrict the size of $S^{(i)}$ and restart the K-Davidson algorithm with the current approximation $X^{(i)}$, exactly the same way it is done for the standard Davidson algorithm. If some of the eigenvectors have converged, it is necessary to deflate them and remove the corresponding residual columns from $R^{(i)}$. We will further discuss these implementation details, also in Section 4.

We note that it is also possible to derive Algorithm 3 by formally applying the standard Davidson algorithm to the equivalent trace minimization problem (17). Although a straightforward implementation of such a scheme would require performing matrix–vector multiplications of the form $KMKX$ in each iteration, the second multiplication with K can be avoided by choosing the preconditioner T to be of the form KD_KD_M.

Finally, let us remark that Algorithm 3 can be viewed as a modification of the old version of the Davidson algorithm in.[22] The latter fully discards the underlying K-symmetry of the matrix MK and handles (5) as a general nonsymmetric eigenvalue problem, which in practice results in the occurrence of complex eigenvalue approximations that hinder convergence.[7] In contrast, the use of the K-inner product in Algorithm 3 redefines the eigenvector extraction procedure, resulting in a *symmetric* reduced eigenvalue problem (21) that yields only real approximations. The efficiency of the K-Davidson algorithm will be demonstrated in Section 5.

3.2 The K-inner product LOBPCG algorithm

The K-inner product can also be easily incorporated into the LOBPCG algorithm[16] to obtain its variant that solves the trace minimization problem (15). In this case, the search subspace $S^{(i)}$ constructed by LOBPCG will be spanned by

$$\{X^{(i)},\ T^{-1}R^{(i)},\ X^{(i-1)}\}, \tag{22}$$

Algorithm 4: The K-inner product LOBPCG (K-LOBPCG) algorithm

Input: Positive definite matrices K and M, a preconditioner T, a starting guess $X^{(0)}$.

Output: A diagonal matrix of k smallest positive eigenvalues Λ of (4), and the associated right eigenvectors $\begin{bmatrix} Y \\ X \end{bmatrix}$.

$X \leftarrow X^{(0)}$; $P \leftarrow [\,]$;
while convergence not reached **do**
  $R \leftarrow MKX - X(X^T KMKX)$;
  $S \leftarrow [X,\ T^{-1}R,\ P]$; $S \leftarrow K\text{-orth}\{S\}$;
  Compute k smallest eigenpairs of of (21) to obtain (Θ^2, C);
  $X \leftarrow SC$; $P \leftarrow X$;
end while
$\Lambda \leftarrow \Theta$;
$Y \leftarrow KX\Theta^{-1}$;

where $X^{(i)}$ is an approximate solution at the ith iteration, $T^{-1}R^{(i)}$ is the matrix of preconditioned residual vectors with the projected gradient $R^{(i)}$ defined in (20), and $X^{(i-1)}$ is the approximation from the previous step. We then use the K-inner product to project MK onto $S^{(i)}$, defined by (22), and obtain approximations to the desired eigenpairs by solving the generalized eigenvalue problem

$$\langle S^{(i)}, MKS^{(i)}\rangle_K C = \langle S^{(i)}, S^{(i)}\rangle_K C\Theta^2, \quad C^T\langle S^{(i)}, S^{(i)}\rangle_K C = I. \tag{23}$$

If the columns of $S^{(i)}$ are K-orthonormalized first, then (23) reduces to a standard eigenvalue problem (21). We outline the major steps of the K-inner product LOBPCG algorithm, further referred to as K-LOBPCG, in Algorithm 4.

Compared to the K-Davidson algorithm, the K-LOBPCG algorithm has smaller memory requirement unless the K-Davidson algorithm is restarted when the dimension of the subspace $S^{(i)}$ reaches $3k$, where k is the number of eigenpairs to be computed. We will show in the next section that K-LOBPCG typically outperforms K-Davidson when the dimension of $S^{(i)}$ is fixed at $3k$ for both methods.

Similar to the K-Davidson algorithm, Algorithm 4 can also be implemented with only

one multiplication of a block of k vectors by K and M, i.e., at most of $2k$ matrix–vector products per iteration are needed. A natural choice of the preconditioner T for K-LOBPCG is given by the matrix of the form $D_K D_M$, where D_K and D_M are approximations of K and M. Since K and M are diagonally dominant in the linear response TDDFT computations, both D_K are D_M are often chosen be the same diagonal matrix D, and diagonal entries of D are chosen to be the differences between the virtual and occupied energies obtained in a ground state calculation. It is possible to choose shifted D's as a preconditioners. However, our numerical experiments indicate that shifting D does not seem to improve the convergence of the K-LOBPCG algorithm.

4 Practical issues

We now discuss a number of algorithmic details that are essential for a proper implementation of the K-Davidson and K-LOBPCG algorithms. We also provide a detailed description of both methods that can be readily used as a guideline for their efficient implementation.

4.1 The conjugate direction in K-LOBPCG

Similar to the original LOBPCG algorithm, in practice, the proposed K-LOBPCG algorithm should not explicitly place the approximations $X^{(i-1)}$ from the previous iteration into the search subspace defined by (22). Instead, a block $P^{(i)}$ of the so-called conjugate directions is defined, such that

$$P^{(i)} = X^{(i)} - X^{(i-1)} C_X^{(i-1)},$$

where $C_X^{(i-1)}$ is a $k \times k$ matrix of coefficients defined from the previous iteration. It can then be observed that the span of the columns of $\{X^{(i)},\ W^{(i)},\ P^{(i)}\}$ is the same, in exact arithmetic, as that of $\{X^{(i)},\ W^{(i)},\ X^{(i-1)}\}$. Therefore, it is possible to replace $X^{(i-1)}$ in the definition (22) of the LOBPCG search subspace by the above defined block $P^{(i)}$. While, mathematically, this does not change the search subspace, the use of the conjugate directions

is preferred in practice, as it leads to a more stable numerical behavior in the presence of roundoff errors.[16]

4.2 K-orthonormality

To obtain an orthonormal basis of $S^{(i)}$ in each K-Davidson iteration, we need to first K-orthogonalize the preconditioned residual block $T^{-1}R^{(i)}$ against $S^{(i-1)}$. This can be achieved by performing

$$W = T^{-1}R^{(i)} - Z\langle Z, T^{-1}R^{(i)}\rangle_K, \tag{24}$$

where $Z = S^{(i-1)}$. The columns of W can be subsequently K-orthonormalized by first performing a Cholesky factorization of $\langle W, W\rangle_K$, that is, computing a lower triangular matrix L such that $\langle W, W\rangle_K = LL^T$, and then solving a number of triangular systems $W^+ \leftarrow WL^{-1}$. When columns of W are close to be linearly dependent, the Cholesky factorization of $\langle W, W\rangle_K$ may break down numerically. In this case, a more stable algorithm based on a truncated SVD of $\langle W, W\rangle_K$ may be used. We refer readers to[23–25] for more details.

Similarly, the K-LOBPCG search subspace is K-orthonormalized by first applying (24) with $Z = X^{(i)}$ to obtain a block of preconditioned residuals W that are K-orthogonal to the current eigenvector approximations $X^{(i)}$. This block is then K-orthonormalized, e.g., using the Cholesky factorization based scheme described above, so that the columns of $Z = [X^{(i)},\ W^+]$ are K-orthonormal. Finally, the conjugate directions $P^{(i)}$ are made K-othogonal to this Z by $P = P^{(i)} - Z\langle Z, P^{(i)}\rangle_K$, and are then themselves K-orthonormalized, which completes the construction of an K-orthonormal basis of the search subspace in the K-LOBPCG algorithm.

4.3 Convergence criterion and deflation

The linear dependence among columns of W is partially caused by a relatively small magnitude of some of its columns. The presence of such columns indicates that approximations

to certain eigenpairs have converged. Unlike some of the existing implementations in which the differences between approximate eigenvalues obtained in two consecutive iterations are used to check the convergence, we use the following convergence criterion: an approximate eigenpair (θ_j^2, x_j), with x_j properly normalized, is considered converged if

$$\|MKx_j - \theta_j^2 x_j\| \leq \tau(\|MK\| + \theta_j^2), \tag{25}$$

where τ is a predefined convergence tolerance. The 2-norm of MK can be estimated by a few steps of a power method. When a converged eigenpair is identified, we remove the corresponding column from the matrix of preconditioned residuals. In the K-LOBPCG algorithm, we also exclude the corresponding column from $P^{(i)}$. At the same time, the converged eigenvector is retained in the matrix $X^{(i)}$ and is used in the subsequent iterations. This type of technique for deflating converged eigenvectors is often called *soft locking*.

4.4 The detailed description

Algorithm 5 gives a detailed description of the K-Davidson algorithm.

As has been already mentioned, one of the key differences between the K-Davidson and K-LOBPCG algorithms is that the storage requirement for K-LOBPCG is fixed whereas K-Davidson can use an increasing amount of memory to store $S^{(i)}$. In practice, however, the memory resources are limited. Therefore a user should specify a parameter, $s_{\max}$, that determines the maximum dimension of the search subspace. Then if a search subspace reaches the dimension of $s_{\max}$, the K-Davidson algorithm should be restarted. This simply requires collapsing $S^{(i)}$ by initializing it with the current eigenvector approximations; see Steps 14–16 of Algorithm 5.

In order to obtain an efficient implementation of the K-Davidson algorithm, it is crucial to ensure that, at every iteration, it performs only one multiplication of a set of vectors with K and one multiplication with M. This can be achieved by allocating additional memory

Algorithm 5: A detailed description of the K-Davidson algorithm

Input: Positive definite matrices K and M, the preconditioner D, a starting guess $X^{(0)}$, and the maximum subspace size parameter $s_{\max}$.

Output: The diagonal matrix of k smallest positive eigenvalues Λ of (4) and the associated eigenvectors $\begin{bmatrix} Y \\ X \end{bmatrix}$.

1: $X \leftarrow K\text{-orth}\{X^{(0)}\}$; $Y \leftarrow KX$; $Y^{(M)} \leftarrow MY$;
2: Initialize $S \leftarrow X$; $KS \leftarrow Y$; $MKS \leftarrow Y^{(M)}$; $sdim \leftarrow k$; $nact \leftarrow k$;
3: **while** convergence not reached (that is, $nact > 0$) **do**
4: Use S, KS, and MKS to form problem (21) and solve it for k lowest eigenpairs;
5: Set $\Lambda \leftarrow \Theta$; $X \leftarrow SC$; $Y \leftarrow (KS)C$; $Y^{(M)} \leftarrow (MKS)C$;
6: $nact \leftarrow 0$; $W \leftarrow [\,]$;
7: **for** $j = 1 \rightarrow k$ **do**
8: $r \leftarrow Y^{(M)}(:, j) - X(:, j)\Lambda^2(j, j)$;
9: **if** $\|r\| > \tau$ **then**
10: $r \leftarrow (D^2 - \Lambda^2(j, j)I)^{-1} r$;
11: $W \leftarrow [W, r]$; $nact \leftarrow nact + 1$;
12: **end if**
13: **end for**
14: **if** $sdim + nact > s_{\max}$ **then**
15: Collapse the subspace $S \leftarrow X$; $KS \leftarrow Y$; $MKS \leftarrow Y^{(M)}$; $sdim \leftarrow k$;
16: **end if**
17: $W \leftarrow W - S((KS)^T W)$; $W^{(K)} \leftarrow KW$; $W^{(MK)} \leftarrow MW^{(K)}$;
18: $R \leftarrow \text{chol}(W^T W^{(K)})$; $W \leftarrow WR^{-1}$; $W^{(K)} \leftarrow W^{(K)} R^{-1}$; $W^{(MK)} \leftarrow W^{(MK)} R^{-1}$;
19: $S \leftarrow [S, W]$; $KS \leftarrow [KS, W^{(K)}]$; $MKS \leftarrow [MKS, W^{(MK)}]$; $sdim \leftarrow sdim + nact$;
20: **end while**
21: $Y \leftarrow Y\Lambda^{-1}$.

to store the matrices $KS^{(i)}$ and $MKS^{(i)}$ along with the search subspace $S^{(i)}$. In this case, the matrix–vector multiplications with K and M are performed only once per iteration to compute $W^{(K)} \equiv KW$ and $W^{(MK)} \equiv MKW$ at Step 17 of Algorithm 5.

The same trade-off between memory requirement and the cost of matrix–vector multiplication exists in the K-LOBPCG algorithm, whose efficient implementation is described in Algorithm 6.

To save extra multiplications with K and M, in addition to the blocks X, W, and P that

Algorithm 6: A detailed description of the K-LOBPCG algorithm

Input: Positive definite matrices K and M, the preconditioner D, and a starting guess $X^{(0)}$.

Output: The diagonal matrix of k smallest positive eigenvalues Λ of (4) and the associated eigenvectors $\begin{bmatrix} Y \\ X \end{bmatrix}$.

1: $X \leftarrow K\text{-orth}\{X^{(0)}\}$; $P \leftarrow [\,]$; $P^{(K)} \leftarrow [\,]$; $P^{(MK)} \leftarrow [\,]$;
2: $Y \leftarrow KX$; $Y^{(M)} \leftarrow MY$;
3: Solve the k-by-k eigenvalue problem $(Y^TY^{(M)})C = (Y^TX)C\Theta^2$, $C(Y^TX)C = I$;
4: $X \leftarrow XC$; $Y \leftarrow YC$; $Y^{(M)} \leftarrow Y^{(M)}C$; $\Lambda \leftarrow \Theta$;
5: **while** convergence not reached **do**
6: $W \leftarrow D^{-2}(Y^{(M)} - X\Lambda^2)$;
7: $W \leftarrow W - X(Y^TW)$; $W^{(K)} \leftarrow KW$; $W^{(MK)} \leftarrow MW^{(K)}$;
8: $R \leftarrow \text{chol}(W^TW^{(K)})$, $W \leftarrow WR^{-1}$; $W^{(K)} \leftarrow W^{(K)}R^{-1}$; $W^{(MK)} \leftarrow W^{(MK)}R^{-1}$;
9: $G \leftarrow Y^TP$; $P \leftarrow P - XG$; $P^{(K)} \leftarrow P^{(K)} - YG$; $P^{(MK)} \leftarrow P^{(MK)} - Y^{(M)}G$;
10: $R \leftarrow \text{chol}(P^TP^{(K)})$, $P \leftarrow PR^{-1}$; $P^{(K)} \leftarrow P^{(K)}R^{-1}$; $P^{(MK)} \leftarrow P^{(MK)}R^{-1}$;
11: Set $S \leftarrow [X,\ W,\ P]$; $KS \leftarrow [Y,\ W^{(K)},\ P^{(K)}]$; $MKS \leftarrow [Y^{(M)},\ W^{(MK)},\ P^{(MK)}]$;
12: Use S, KS, and MKS to form problem (21) and solve it for k lowest eigenpairs;
13: Set $\Lambda \leftarrow \Theta$; $C_X \leftarrow C(1:k,:)$; $C_W \leftarrow C(k+1:2k,:)$; $C_P \leftarrow C(2k+1:3k,:)$;
14: $P \leftarrow WC_W + PC_P$;
15: $P^{(K)} \leftarrow W^{(K)}C_W + P^{(K)}C_P$; $P^{(MK)} \leftarrow W^{(MK)}C_W + P^{(MK)}C_P$;
16: $X \leftarrow XC_X + P$;
17: $Y \leftarrow YC_X + P^{(K)}$; $Y^{(M)} \leftarrow Y^{(M)}C_X + P^{(MK)}$;
18: **end while**
19: $Y \leftarrow Y\Lambda^{-1}$.

define the search subspace, the K-LOBPCG algorithm should keep in memory and update the matrices KX, MKX, KW, MKW, KP, and MKP. Then, similar to K-Davidson, the matrix–vector multiplications with K and M can be performed only once per iteration, at Step 7 of Algorithm 5.

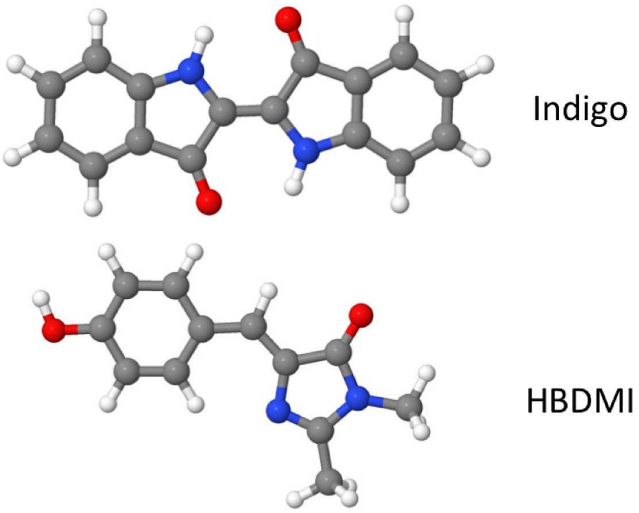


Figure 1: The molecules HBDMI and Indigo.

5 Computational Results

In this section, we compare the performance of the new algorithms presented in this paper with existing ones that have been implemented in widely used computational chemistry software packages and other algorithms recently proposed[4] to solve the linear response TDDFT eigenvalue problem.

The test problems we use in this section include the Indigo and 4'-hydroxybenzylidene-2,3-dimethylimidazoline (HBDMI)(Figure 1). All geometries were optimized in the ground state DFT calculation using the B3LYP[26] functional and 6-31G(d)[27,28] basis set. HBDMI system involves 57 occupied and 207 virtual orbitals, while Indigo has 68 occupied and 252 virtual orbitals, respectively. The dimension of K and M is thus 11799 for HBDMI and 12096 for Indigo (20 core orbitals were frozen in calculations).

The first set of tests were performed in MATLAB using the matrices K, M produced by the TDDFT module of the NWChem program.[9] We also use the ground state orbital energies ε_j, where j is an occupied orbital, and ε_a, where a is a virtual orbital, to construct a diagonal preconditioner $T = D^2$, where the diagonal elements of D are $\varepsilon_a - \varepsilon_j$ for all a and j.

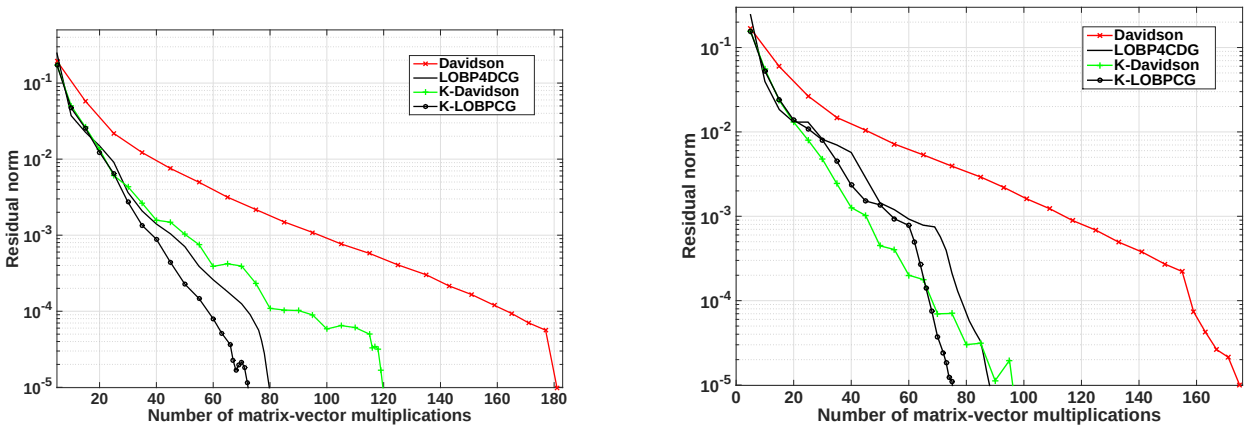
In these tests, we compute only the five lowest eigenpairs of MK. Because the dominant cost in all algorithms is in the multiplication of K and M with vectors, we measure the performance of each algorithm by the number of matrix–vector multiplications it performs. The cost of all other linear algebra operations is negligible.

We set the convergence tolerance τ in (25) to $\tau = 10^{-5}$ and require x_j to have unit norm in K-inner product for all algorithms. In all runs, the starting guess $X^{(0)}$ is chosen to be k columns from D that contain the k smallest diagonal entries of D. We should note that this initial guess is much better than a set of randomly generated vectors.

Two different settings of the Davidson and the K-Davidson algorithms are used in our tests. In the first setting, we limit the dimension of $S^{(i)}$ in the K-Davidson algorithm to be $3k$, where $k = 5$ is the number of eigenpairs to be computed. The dimension of the subspace generated in the existing Davidson algorithm (as implemented in NWChem) is set to $4k$. Both versions of Davidson algorithms are restarted when convergence is not reached before the limit on the subspace dimension is hit.

Figure 2 shows that both the K-Davidson and K-LOBPCG algorithms are much faster than the existing Davidson algorithm implemented in NWChem. For the HBDMI test problem, the LOBP4DCG algorithm is slightly slower than the K-LOBPCG algorithm, but is slightly faster than the K-Davidson algorithm. For the Indigo test problem, the performance of K-Davidson, K-LOBPCG and LOBP4DCG are comparable. The K-LOBPCG algorithm is slightly faster than the other two. Note that in all of our figures the plotted residual norm is the norm of the maximum residual of the actual linear response eigenvalue problem (4) evaluated at the approximations to the eigenvector components x_j and y_j available at the current iteration.

In the second setting, we lift the restriction on the dimension of $S^{(i)}$ in the K-Davidson algorithm and the dimension of S in the existing Davidson algorithm. Because the K-LOBPCG algorithm does not take advantage of extra available memory, its convergence behavior does not change. Both the K-Davidson algorithm and the existing Davidson algo-



(a) (b)

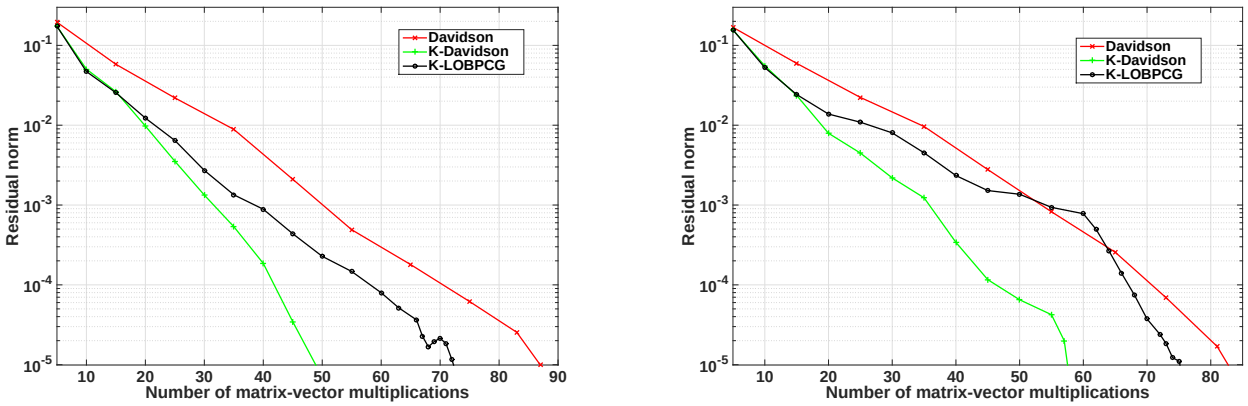
Figure 2: The number of multiplications as a measure of the performance of algorithms in the low-memory scenario for (a) HBDMI (b) Indigo.

rithm exhibit faster convergence when more memory is available, as we can see from Figure 3. However, it is clear from Figure 3 that the K-Davidson algorithm is still much faster than the existing Davidson algorithm. The K-LOBPCG algorithm is unstandably slower than the K-Davidson algorithm since the subspace from which approximate eigenpairs are extracted is much smaller. However, we can see from Figure 3 that it appears to be faster than the existing Davidson algorithm for both test problems.

To demonstrate the performance the K-Davidson algorithm for larger problems and for problems in which a wider energy range needs to be examined, we implemented the K-Davidson algorithm in the NWChem program, and tested the algorithm on the Indigo molecule using the cc-pVTZ basis set. For this test, the dimension of K and M is $n_o \times n_v = 53176$. We compute the lowest 100 excitation energies and the corresponding eigenvectors.

We performed the computation on on the Cascade system, which is equipped with 1440 Xeon E5-2670 8C 2.6 GHz 16-core CPUs, 128 GB memory per compute node and a Infiniband FDR network, and maintained at the EMSL user facility located at the Pacific Northwest National Laboratory. Both calculations utilized 128 CPU cores across 8 nodes.

In Figure 4, we plot the residual norm associated with the approximate eigenpairs obtained at each K-Davidson and existing Davidson iteration against the wall clock time.



(a) (b)

Figure 3: The number of multiplications as a measure of the performance of algorithms in the unlimited-memory scenario for (a) HBDMI (b) Indigo.

We used the same initial vectors for both algorithms. As we can see from this figure, the K-Davidson algorithm converges much more rapidly. Even though the same number of iterations is used in both algorithms before reaching convergence, the cost per K-Davidson iteration is roughly two times cheaper in the first few iterations. Furthermore, after the fourth iteration, a significantly number of approximate eigenpairs have converged in the K-Davidson algorithm. As a result, subsequent K-Davidson iterations are much cheaper because fewer matrix–vector multiplications are used in these iterations. Overall, the K-Davidson algorithm performed only 436 matrix–vector multiplications, which is much less than 1162 multiplications performed by Davidson. The total wall clock time consumed by K-Davidson is 3042 seconds, which is 2.8× less than the 8522 seconds used by the existing Davidson algorithm.

6 Conclusions

Although the observation that the linear response eigenvalue problem (1) is equivalent to the product eigenvalue problem (5) or (6) is well known, it appears that the existing algorithms for solving this type of eigenvalue problem try to approximate both the x and y components of

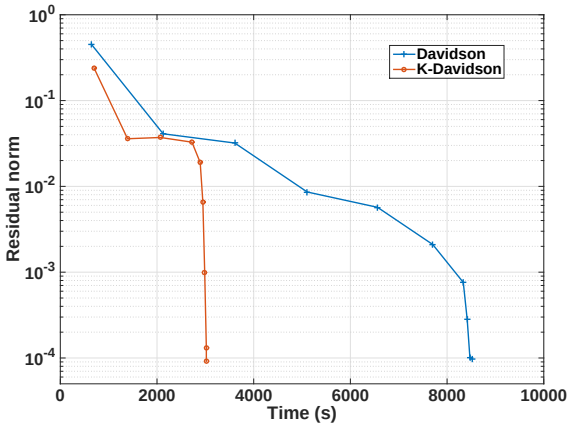


Figure 4: The residual norm as a function of execution time for the Davidson and K-Davidson implementations. The Indigo system in cc-pVTZ basis set.

the eigenvector simultaneously. The algorithms we present in this paper have two features: 1) we compute the x component of the eigenvector in an iterative procedure designed to solve the product eigenvalue problem. The y component of the eigenvector is recovered in a postprocessing procedure. 2) we use the observation that the product eigenvalue problem $MKx = \lambda^2 x$ is self-adjoint with respect to the K-inner product and simply modify standard Davidson and LOBPCG algorithms to make use of K-inner product to compute the desired eigenvalues and eigenvectors of this problem. A similar set of algorithms can be developed by noticing that KM is self-adjoint with respect to the M-inner product. We implemented our algorithms in the NWChem software package and demonstrated that they are more efficient than existing algorithms by numerical examples.

Acknowledgement

Support for this work was provided through Scientific Discovery through Advanced Computing (SciDAC) program funded by U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research and Basic Energy Sciences at the Lawrence Berkeley National Laboratory under Contract No. DE-AC02-05CH1123 and at the Pacific Northwest National Laboratory (PNNL) under Award number KC030102062653. A portion of the research was

performed using EMSL, a DOE Office of Science User Facility sponsored by the Office of Biological and Environmental Research and located at the PNNL. PNNL is operated by Battelle Memorial Institute for the United States Department of Energy under DOE contract number DE-AC05-76RL1830. The research also benefited from resources provided by the National Energy Research Scientific Computing Center (NERSC), a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

References

(1) Shao, M.; Jornada, F. H.; Yang, C.; Deslippe, J.; Louie, S. G. *Linear Algebra Appl.* **2016**, *488*, 148–167.

(2) Blackford, L. S.; Choi, J.; Cleary, A.; D'Azevedo, E.; Demmel, J.; Dhillon, I.; Dongarra, J.; Hammarling, S.; Henry, G.; Petitet, A.; Stanley, K.; Walker, D.; Whaley, R. C. *ScaLAPACK Users' Guide*; SIAM: Philadelphia, PA, 1997.

(3) Brabec, J.; Lin, L.; Shao, M.; Govind, N.; Saad, Y.; Yang, C.; Ng, E. G. *J. Chem. Theory Comput.* **(to appear)**,

(4) Bai, Z.; Li, R.-C. *SIAM J. Matrix Anal. Appl.* **2013**, *34*, 392–416.

(5) Challacombe, M. *Computation* **2014**, *2*, 1–11.

(6) Kressner, D.; Miloloža Pandur, M.; Shao, M. *Numer. Algor.* **2014**, *66*, 681–703.

(7) Olsen, J.; Jorgen, H.; Jensen, A.; Jorgensen, P. *J. Comput. Phys.* **1988**, *74*, 265–282.

(8) Stratmann, R. E.; Scuseria, G. E.; Frisch, M. J. *J. Chem. Phys.* **1998**, *109*, 8218–8224.

(9) Valiev, M.; Bylaska, E. J.; Govind, N.; Kowalski, K.; Straatsma, T. P.; Dam, H. J. J. V.; Wang, D.; Nieplocha, J.; Apra, E.; Windus, T. L.; de Jong, W. A. *Comput. Phys. Commun.* **2010**, *181*, 1477–1489.

(10) Bai, Z.; Li, R.-C. *SIAM J. Matrix Anal. Appl.* **2012**, *33*, 1075–1100.

(11) Davidson, E. R. *J. Comput. Phys.* **1975**, *17*, 87–94.

(12) Frisch, M. J.; Trucks, G. W.; Schlegel, H. B.; Scuseria, G. E.; Robb, M. A.; Cheeseman, J. R.; Scalmani, G.; Barone, V.; Mennucci, B.; Petersson, G. A.; Nakatsuji, H.; Caricato, M.; Li, X.; Hratchian, H. P.; Izmaylov, A. F.; Bloino, J.; Zheng, G.; Sonnenberg, J. L.; Hada, M.; Ehara, M.; Toyota, K.; Fukuda, R.; Hasegawa, J.; Ishida, M.; Nakajima, T.; Honda, Y.; Kitao, O.; Nakai, H.; Vreven, T.; Montgomery, J. A., Jr.; Peralta, J. E.; Ogliaro, F.; Bearpark, M.; Heyd, J. J.; Brothers, E.; Kudin, K. N.; Staroverov, V. N.; Kobayashi, R.; Normand, J.; Raghavachari, K.; Rendell, A.; Burant, J. C.; Iyengar, S. S.; Tomasi, J.; Cossi, M.; Rega, N.; Millam, J. M.; Klene, M.; Knox, J. E.; Cross, J. B.; Bakken, V.; Adamo, C.; Jaramillo, J.; Gomperts, R.; Stratmann, R. E.; Yazyev, O.; Austin, A. J.; Cammi, R.; Pomelli, C.; Ochterski, J. W.; Martin, R. L.; Morokuma, K.; Zakrzewski, V. G.; Voth, G. A.; Salvador, P.; Dannenberg, J. J.; Dapprich, S.; Daniels, A. D.; Farkas, .; Foresman, J. B.; Ortiz, J. V.; Cioslowski, J.; Fox, D. J. Gaussian 09, Revision D.01. Gaussian Inc., Wallingford, CT, 2009.

(13) Gordon, M. S.; Schmidt, M. W. In *Theory and Applications of Computational Chemistry: the first forty years*; Dykstra, C. E., Frenking, G., Kim, K. S., Scuseria, G. E., Eds.; Elsevier, Amsterdam, 2005; pp 1167–1189.

(14) Shao, Y.; Gan, Z.; Epifanovsky, E.; Gilbert, A. T. B.; Wormit, M.; Kussmann, J.; Lange, A. W.; Behn, A.; Deng, J.; Feng, X.; Ghosh, D.; Goldey, M.; Horn, P. R.; Jacobson, L. D.; Kaliman, I.; Khaliullin, R. Z.; Kús, T.; Landau, A.; Liu, J.; Proynov, E. I.; Rhee, Y. M.; Richard, R. M.; Rohrdanz, M. A.; Steele, R. P.; Sundstrom, E. J.; Woodcock III, H. L.; Zimmerman, P. M.; Zuev, D.; Albrecht, B.; Alguire, E.; Austin, B.; Beran, G. J. O.; Bernard, Y. A.; Berquist, E.; Brandhorst, K.; Bravaya, K. B.; Brown, S. T.; Casanova, D.; Chang, C.-M.; Chen, Y.; Chien, S. H.; Closser, K. D.; Crit-

tenden, D. L.; Diedenhofen, M.; DiStasio Jr., R. A.; Dop, H.; Dutoi, A. D.; Edgar, R. G.; Fatehi, S.; Fusti-Molnar, L.; Ghysels, A.; Golubeva-Zadorozhnaya, A.; Gomes, J.; Hanson-Heine, M. W. D.; Harbach, P. H. P.; Hauser, A. W.; Hohenstein, E. G.; Holden, Z. C.; Jagau, T.-C.; Ji, H.; Kaduk, B.; Khistyaev, K.; Kim, J.; Kim, J.; King, R. A.; Klunzinger, P.; Kosenkov, D.; Kowalczyk, T.; Krauter, C. M.; Lao, K. U.; Laurent, A.; Lawler, K. V.; Levchenko, S. V.; Lin, C. Y.; Liu, F.; Livshits, E.; Lochan, R. C.; Luenser, A.; Manohar, P.; Manzer, S. F.; Mao, S.-P.; Mardirossian, N.; Marenich, A. V.; Maurer, S. A.; Mayhall, N. J.; Oana, C. M.; Olivares-Amaya, R.; O'Neill, D. P.; Parkhill, J. A.; Perrine, T. M.; Peverati, R.; Pieniazek, P. A.; Prociuk, A.; Rehn, D. R.; Rosta, E.; Russ, N. J.; Sergueev, N.; Sharada, S. M.; Sharmaa, S.; Small, D. W.; Sodt, A.; Stein, T.; Stück, D.; Su, Y.-C.; Thom, A. J. W.; Tsuchimochi, T.; Vogt, L.; Vydrov, O.; Wang, T.; Watson, M. A.; Wenzel, J.; White, A.; Williams, C. F.; Vanovschi, V.; Yeganeh, S.; Yost, S. R.; You, Z.-Q.; Zhang, I. Y.; Zhang, X.; Zhou, Y.; Brooks, B. R.; Chan, G. K. L.; Chipman, D. M.; Cramer, C. J.; Goddard III, W. A.; Gordon, M. S.; Hehre, W. J.; Klamt, A.; Schaefer III, H. F.; Schmidt, M. W.; Sherrill, C. D.; Truhlar, D. G.; Warshel, A.; Xua, X.; Aspuru-Guzik, A.; Baer, R.; Bell, A. T.; Besley, N. A.; Chai, J.-D.; Dreuw, A.; Dunietz, B. D.; Furlani, T. R.; Gwaltney, S. R.; Hsu, C.-P.; Jung, Y.; Kong, J.; Lambrecht, D. S.; Liang, W.; Ochsenfeld, C.; Rassolov, V. A.; Slipchenko, L. V.; Subotnik, J. E.; Van Voorhis, T.; Herbert, J. M.; Krylov, A. I.; Gill, P. M. W.; Head-Gordon, M. *Mol. Phys.* **2015**, *113*, 184–215.

(15) Sameh, A. H.; Wisniewski, J. A. *SIAM J. Numer. Anal.* **1982**, *19*, 1243–1259.

(16) Knyazev, A. *SIAM J. Sci. Comput.* **2001**, *23*, 517–541.

(17) Tsiper, E. V. *J. Phys. B* **2001**, *34*, L401–L407.

(18) Thouless, D. J. *Nucl. Phys.* **1961**, *22*, 78–95.

(19) Tsiper, E. V. *J. Exp. Theor. Phys. Lett.* **1999**, *70*, 751–755.

(20) Liu, B. *The simultaneous expansion method for the iterative solution of several of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices*; 1978.

(21) Sleijpen, G. L. G.; Van der Vorst, H. A. *SIAM J. Matrix Anal. Appl.* **1996**, *17*, 401–425.

(22) Rettrup, S. *J. Comput. Phys.* **1982**, *45*, 100–107.

(23) Duersch, J. A.; Gu, M.; Shao, M.; Yang, C. A robust and efficient implementation of LOBPCG. 2015 (in preparation).

(24) Hetmaniuk, U.; Lehoucq, R. *J. Comput. Phys.* **2006**, *218*, 324–332.

(25) Stathopoulos, A.; Wu, K. *SIAM J. Sci. Comput.* **2002**, *23*, 2162–2182.

(26) Becke, A. D. *J. Chem. Phys.* **1993**, *98*, 5648–5652.

(27) Hehre, W. J.; Ditchfield, R.; Pople, J. A. *J. Chem. Phys.* **1972**, *56*, 2257–2261.

(28) Hariharan, P.; Pople, J. *Theor. Chim. Acta.* **1973**, *28*, 213–222.