Commun. Comput. Phys.
doi: 10.4208/cicp.060314.120215a

Vol. **18**, No. 1, pp. 167-179
July 2015

On the "Preconditioning" Function Used in Planewave DFT Calculations and its Generalization

Yunkai Zhou[1,*], James R. Chelikowsky[2], Xingyu Gao[3] and Aihui Zhou[4]

[1] *Department of Mathematics, Southern Methodist University, Dallas, TX 75275, USA.*
[2] *Center for Computational Materials, Institute for Computational Engineering and Science, and Departments of Physics and Chemical Engineering, University of Texas, Austin, TX 78712, USA.*
[3] *HPCC, Institute of Applied Physics and Computational Mathematics, Beijing, 100094, China.*
[4] *LSEC, Institute of Computational Mathematics and Scientific/Engineering Computing, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China.*

Received 6 March 2014; Accepted (in revised version) 12 February 2015

Abstract. The Teter, Payne, and Allan "preconditioning" function plays a significant role in planewave DFT calculations. This function is often called the TPA preconditioner. We present a detailed study of this "preconditioning" function. We develop a general formula that can readily generate a class of "preconditioning" functions. These functions have higher order approximation accuracy and fulfill the two essential "preconditioning" purposes as required in planewave DFT calculations. Our general class of functions are expected to have applications in other areas.

AMS subject classifications: 00A69, 05C50, 15A18, 65F15

Key words: Density functional theory, planewave, preconditioning function, eigenvalue problem.

1 Introduction

Density functional theory (DFT) [6, 9] has profound impacts on atomic scale material studies, including materials processing and design. There are now quite extensive literature on DFT. Readers interested in the role of DFT within molecular and condensed matter physics may consult a few recent books *e.g.* [3, 8, 12, 15].

*Corresponding author. *Email addresses:* `yzhou@smu.edu` (Y. Zhou), `jrc@ices.utexas.edu` (J. R. Chelikowsky), `gao_xingyu@iapcm.ac.cn` (X. Gao), `azhou@lsec.cc.ac.cn` (A. Zhou)

One major approach for DFT calculations is by using planewave basis expansion. There are several influential planewave DFT software, including the open source QUANTUM- ESPRESSO [4,18], ABINIT [5,17], and the commercial VASP [10,19]. Readers interested in learning planewave DFT calculations may start with the nice KSSOLV package written in Matlab [21].

In DFT calculations for atomic scale study of materials, the major computational cost is usually spent on solving the Kohn-Sham equation. This equation is a nonlinear eigenvalue problem, which represents certain simplification of the Schrödinger equation to make it more numerically tractable. A self-consistency loop is utilized to address the nonlinearity, which means a sequence of linearized eigenvalue problems need to be solved until self-consistency is reached [12,15]. Therefore, eigensolvers used in the DFT packages can be of crucial importance for the efficiency of the DFT calculations.

For the planewave DFT packages ([4,5,10,21] and others), two of the essential eigensolvers implemented are the preconditioned CG method and the preconditioned Davidson method. The efficiency of these eigensolvers is closely related to the following "preconditioning" function,

$$K(x)=\frac{27+18x+12x^2+8x^3}{27+18x+12x^2+8x^3+16x^4}. \tag{1.1}$$

However, few studies of this function appear in the planewave DFT literature, except that most papers utilizing a preconditioned eigensolver in the planewave setting would refer to the work of Teter, Payne and Allan [16]. The function $K(x)$ was first proposed in [16] and is now known as the TPA preconditioner.

A known property of $K(x)$ is that its derivatives up to order 3 at $x=0$ are all zeros. At first sight the $K(x)$ appears intriguing. One may wonder why $K(x)$ has to be in the form (1.1). Do the coefficients as listed in (1.1) lead to the property that $K'(0)=K''(0)=K'''(0)=0$? Can the coefficients be modified?

In this note we develop a generalization of $K(x)$. Besides readily answering the previously mentioned questions about $K(x)$, our generalization provides formulas with higher accuracy to fulfill the purposes of "preconditioning" in the planewave setting.

2 Up to order n consecutive zero derivatives at $x=0$

Our first finding is that there is no particular mystery related to $K'(0)=K''(0)=K'''(0)=0$, this property actually is independent of the coefficients listed in (1.1). In fact, it is a special case of the general result we present below.

We first define $p_n(x)$ and $g_n(x)$ as

$$p_n(x):=c_0+c_1x+c_2x^2+\cdots+c_nx^n, \tag{2.1}$$

$$g_n(x):=\frac{p_n(x)}{p_n(x)+c_{n+1}x^{n+1}}. \tag{2.2}$$

Our result says that all the derivatives of $g_n(x)$ up to order n at $x=0$ must be zero, as long as $c_0 \neq 0$.

Theorem 2.1. *Let $p_n(x)$ be any degree n polynomial as defined in* (2.1)*, with $c_0 \neq 0$. Define a rational function $g_n(x)$ as in* (2.2)*, with $c_{n+1} \neq 0$. Then the smallest positive integer i that satisfies $g_n^{(i)}(0) \neq 0$ is $i=n+1$, and $g_n^{(n+1)}(0) = -(n+1)!\frac{c_{n+1}}{c_0}$.*

The direct proof of this theorem by explicitly computing the derivatives is quite tedious and thus omitted. There is a simple way to show the first part, that is, $g_n^{(i)}(0)=0$ for all $i=1,2,\cdots,n$: Notice that $g_n(x) = 1 - \frac{c_{n+1}x^{n+1}}{p_n(x)+c_{n+1}x^{n+1}}$, from which we see that $g_n'(x)$ has n multiple roots at $x=0$. Thus the derivatives of $g'(x)$ up to order $n-1$ must be zero at $x=0$, this means the derivatives of $g(x)$ up to order n must be zero at $x=0$. However, to reach the conclusion that $g_n^{(n+1)}(0) = -(n+1)!\frac{c_{n+1}}{c_0}$ involves some tedious calculations, for which we omit here.

We use state-of-the-art symbolic computation software Mathematica® to verify the results in Theorem 2.1. Fig. 1 shows the consecutive derivatives for some small n, the computed results are exactly what are predicted by Theorem 2.1.

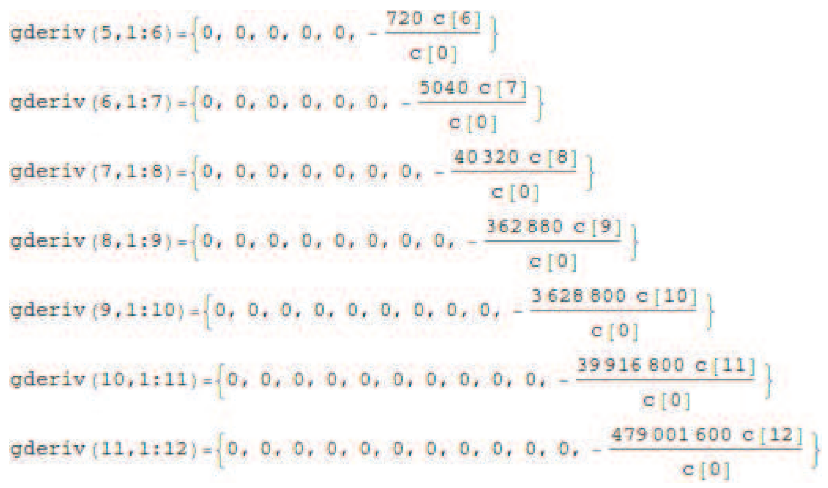
```
gderiv(5,1:6)={0, 0, 0, 0, 0, - 720 c[6] / c[0]}
gderiv(6,1:7)={0, 0, 0, 0, 0, 0, - 5040 c[7] / c[0]}
gderiv(7,1:8)={0, 0, 0, 0, 0, 0, 0, - 40320 c[8] / c[0]}
gderiv(8,1:9)={0, 0, 0, 0, 0, 0, 0, 0, - 362880 c[9] / c[0]}
gderiv(9,1:10)={0, 0, 0, 0, 0, 0, 0, 0, 0, - 3628800 c[10] / c[0]}
gderiv(10,1:11)={0, 0, 0, 0, 0, 0, 0, 0, 0, 0, - 39916800 c[11] / c[0]}
gderiv(11,1:12)={0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, - 479001600 c[12] / c[0]}
```

Figure 1: This figure shows some screen output from the Mathematica® code in Listing 1. The `gderiv(n,1:n+1)` contains the values of $g_n^{(j)}(0)$ for $j=1,\cdots,n+1$. The computed derivatives clearly agree with the results in Theorem 2.1.

The Mathematica® code is presented in Listing 1. This code contains specific Mathematica® syntax but the main structure should be self-explanatory. The two 2-d arrays '`gderiv`' and '`cputime`' are not needed for verifying the results, we used them only to measure the computational cost.

We point out that using Mathematica® to compute the derivatives can encounter difficulty pretty quickly when n is increased. The verification process turns out to be surprisingly challenging for both Mathematica® and the hardware we use. In fact, when $n=40$, Listing 1: Compute the first $n+1$ derivatives at $x=0$ (Mathematica® script)

```
Clear[n, nmin, nmax, f, g]
f[x_] := Sum[c[i]*x^i, {i, 0, n}]
g[x_] := f[x]/(f[x] + c[n + 1]*x^(n + 1))
nmin = 3; nmax = 40;
gderiv = ConstantArray[1, {nmax, nmax + 1}]; (* intentionally initialized to nonzeros *)
cputime= ConstantArray[0, {nmax, nmax + 1}];
For[n = nmin, n <= nmax, n+=1,
   For[j = 1, j <= n + 1, j+=1,
      cpu0 = TimeUsed[];
      gderiv[[n, j]] = Simplify[D[g[x], {x, j}] /. x −> 0]; (* compute the j−th derivative at x=0 *)
      cputime[[n,j]] = TimeUsed[] − cpu0;
   ];
   Print["gderiv(", n, ",1:", n+1,")=", gderiv[[n, 1 ;; n + 1]]] (* show the first n+1 derivatives in a row *)
]
```

the memory used by the Mathematica® MathKernel is already around 9 GB, owing to the exponentially increasing number of terms in the higher derivatives that the MathKernel has to compute.

In fact we wrote a more efficient code, which is about four times faster than the one in Listing 1. The idea is to progressively use the simple relationship $g^{(j)}(x)=\frac{d}{dx}g^{(j-1)}(x)$ to compute the j-th order derivative of $g(x)$, instead of using the direct Mathematica® command `D[g[x],{x,j}]`. We omit this faster but harder to read code, while pointing out that even the faster code encountered significant difficulty for $n\geq 46$. In contrast, Theorem 2.1 readily works for all integer n. We consider this as a nice example of the usefulness of theoretical results vs the brute force computations, even if the latter is carried out with state-of-the-art software and hardware.

3 Asymptotic expansion correct up to order $n+1$ in $\frac{1}{x}$

Theorem 2.1 answers the question on why $K'(0)=K''(0)=K'''(0)=0$. However, it does not explain why the coefficients of $K(x)$ are chosen as those listed in (1.1). We need to return to the work of Teter *et al.* [16], which proposed the "preconditioning" function (1.1), for some insight. Only a few lines are given in [16] on $K(x)$, however, we think these lines contain the essence for the "preconditioning" in the planewave setting. The authors give two purposes for "preconditioning", according to their physical insight: (i) $K(x)$ should approximate x well for small x, so that "the low-wave-number components are left unchanged"; and (ii), $K(x)$ approaches $\frac{1}{2(x-1)}$ with an asymptotic expansion correct up to the fourth order in $\frac{1}{x}$ for $x>1$, so that "the high-wave-number components are reduced". The factor $\frac{1}{2}$ in $\frac{1}{2(x-1)}$ is used to average "the joining of low-x and high-x expansions", as mentioned in [16].

Satisfying the $K'(0)=K''(0)=K'''(0)=0$ would fulfill purpose (i). However, it is the purpose (ii) that provides sufficient conditions to uniquely determine the coefficients in

$K(x)$ up to a scaling factor. After struggling on the meaning of the "asymptotic expansion correct up to the fourth order in $\frac{1}{x}$," we find a general approach that can be used to readily construct asymptotic expansions with higher order accuracy in $\frac{1}{x}$ for $x>1$.

Our approach is presented in the proof of Theorem 3.1. Since the $\frac{1}{2}$ in $\frac{1}{2(x-1)}$ is just a scaling factor, we can replace it with a general $\frac{1}{\xi}$ where $\xi\neq 0$.

Theorem 3.1. *Let ξ be any nonzero constant. If the coefficients $\{c_i\}$'s as in* (2.1) *and* (2.2) *satisfy*

$$\frac{c_0}{c_1}=\frac{c_1}{c_2}=\cdots=\frac{c_{n-1}}{c_n}=\frac{\xi+1}{\xi},\quad \textit{and}\quad c_{n+1}=\xi c_n, \tag{3.1}$$

then the $g_n(x)$ as defined via (2.1) *and* (2.2) *asymptotically approximates $\frac{1}{\xi(x-1)}$ correct up to the $(n+1)$-th order in $\frac{1}{x}$ for $x>1$.*

Proof. In order for $g_n(x)$ to have an asymptotic expansion accurate up to order $n+1$ in $\frac{1}{x}$ on the region $x>1$, we only need to consider the $x\to\infty$ case and make sure that the coefficients of the $\frac{1}{x^j}$ terms in the asymptotic expansion of $g_n(x)$ are the same as the corresponding coefficients in the expansion of $\frac{1}{\xi(x-1)}$ for $j=1,\cdots,n+1$.

The coefficients for all the $\frac{1}{x^j}$ terms in the expansion of $\frac{1}{\xi(x-1)}$ are easily seen to be the same constant $\frac{1}{\xi}$, since $\frac{1}{\xi(x-1)}=\frac{1}{\xi}\sum_{j=1}^{\infty}\frac{1}{x^j}$ for $x>1$.

Therefore we need the coefficients of the $\frac{1}{x^j}$ terms in the asymptotic expansion of $g_n(x)$ to be $\frac{1}{\xi}$ for all $j=1,\cdots,n+1$.

When $j=1$, it is clear that in order to get $g_n(x)\to\frac{1}{\xi}\frac{1}{x}$ as $x\to\infty$, we need $c_{n+1}=\xi c_n$. When $j=2$, we need $g_n(x)-\frac{1}{\xi}\frac{1}{x}\to\frac{1}{\xi}\frac{1}{x^2}$ as $x\to\infty$, simplifying this leads to the condition $\xi c_{n-1}=c_n+c_{n+1}$. Since $c_{n+1}=\xi c_n$ from the previous step, we get $\frac{c_{n-1}}{c_n}=\frac{\xi+1}{\xi}$.

In general, in order for the $\frac{1}{x^{j+1}}$ term, $j\geq 2$, to have coefficient $\frac{1}{\xi}$ in the expansion of $g_n(x)$, we need $g_n(x)-\frac{1}{\xi}\sum_{\ell=1}^{j}\frac{1}{x^\ell}\to\frac{1}{\xi}\frac{1}{x^{j+1}}$ as $x\to\infty$. Simplifying this leads to the condition

$$\xi c_{n-j}=c_{n-j+1}+c_{n-j+2}+\cdots+c_{n+1}.$$

The same reasoning and simplification at its previous step would lead to $\xi c_{n-j+1}=c_{n-j+2}+\cdots+c_{n+1}$. Therefore

$$\xi c_{n-j}=(\xi+1)c_{n-j+1},\quad 2\leq j\leq n.$$

Combining all the previous conditions, we have exactly (3.1). From the derivation above we see that these conditions guarantee that the asymptotic expansion of $g_n(x)$ approximates $\frac{1}{\xi(x-1)}$ correct up to the $(n+1)$-th order in $\frac{1}{x}$ for $x>1$. □

There are $n+1$ conditions in (3.1) for determining the $n+2$ coefficients $\{c_i\}$'s in $g_n(x)$. Owing to the quotient in the definition of $g_n(x)$, it is clear that the $\{c_i\}$'s are unique up to a scaling factor. For convenience, we can define

$$c_i:=(\xi+1)^{n-i}\xi^i,\quad i=0,\cdots,n,\quad \text{and}\quad c_{n+1}:=\xi^{n+1}, \tag{3.2}$$

for which (3.1) is clearly satisfied.

Based on (2.2) and (3.2), we can easily construct functions that approximate $\frac{1}{\xi(x-1)}$ correct up to any desired accuracy in $\frac{1}{x}$ for $x>1$.

Using $\xi=2$ as an example, and denoting $g_n(x)$ as $g(x,n)$, we see that $g(x,2)=\frac{9+6x+4x^2}{9+6x+4x^2+8x^3}$, and that $g(x,3)=K(x)$, which is the "preconditioning" function (1.1) used in VASP and other planewave DFT codes. This now fully explains why the coefficients in (1.1) are chosen as they are. But it is now easy to construct formulas for any $n>3$. Functions with $n=4,5,6,7$ are listed below, larger n's are omitted for brevity.

$$\begin{aligned}
g(x,4)&=\frac{81+54x+36x^2+24x^3+16x^4}{81+54x+36x^2+24x^3+16x^4+32x^5},\\
g(x,5)&=\frac{243+162x+108x^2+72x^3+48x^4+32x^5}{243+162x+108x^2+72x^3+48x^4+32x^5+64x^6},\\
g(x,6)&=\frac{729+486x+324x^2+216x^3+144x^4+96x^5+64x^6}{729+486x+324x^2+216x^3+144x^4+96x^5+64x^6+128x^7},\\
g(x,7)&=\frac{2187+1458x+972x^2+648x^3+432x^4+288x^5+192x^6+128x^7}{2187+1458x+972x^2+648x^3+432x^4+288x^5+192x^6+128x^7+256x^8}.
\end{aligned}$$

Fig. 2 shows the approximations of $g(x,n)$ for $n=3,\cdots,10$ to $\frac{1}{2(x-1)}$.

4 A general formula for higher order approximation in $[x_0,\infty)$

Having consecutive zero higher order derivatives at a given point x_0 is often a highly desirable property for developing faster and/or more accurate numerical algorithms. Thanks to the Taylor expansion, certain error function (*e.g.*, a function that depends on the distance from x to x_0) may be reduced much faster when it has consecutive zero higher order derivatives at x_0.

From Theorem 2.1, we know that there is a class of functions that enjoy the consecutive zero higher order derivatives property for free. In addition, we can request higher order asymptotic expansion accuracy of these functions to $\frac{1}{\xi(x-x_0-1)}$ for $x>x_0+1$.

We first define the following functions,

$$p_\xi(x,n):=c_0+c_1(x-x_0)+c_2(x-x_0)^2+\cdots+c_n(x-x_0)^n, \tag{4.1}$$

$$g_\xi(x,n):=\frac{p_\xi(x,n)}{p_\xi(x,n)+c_{n+1}(x-x_0)^{n+1}}, \tag{4.2}$$

where $\{c_i\}$'s are defined the same as in (3.2), independent of x_0.

Theorem 4.1. *For the $g_\xi(x,n)$ as defined via* (3.2), (4.1), *and* (4.2), *we have $\frac{\partial^i}{\partial x^i}g_\xi(x_0,n)=0$ for $i=1,\cdots,n$, and $\frac{\partial^{n+1}}{\partial x^{n+1}}g_\xi(x_0,n)=-(n+1)!\frac{c_{n+1}}{c_0}$. Moreover, the $g_\xi(x,n)$ asymptotically approximates $\frac{1}{\xi(x-x_0-1)}$ correct up to the $(n+1)$-th order in $\frac{1}{x-x_0}$ for $x>x_0+1$.*

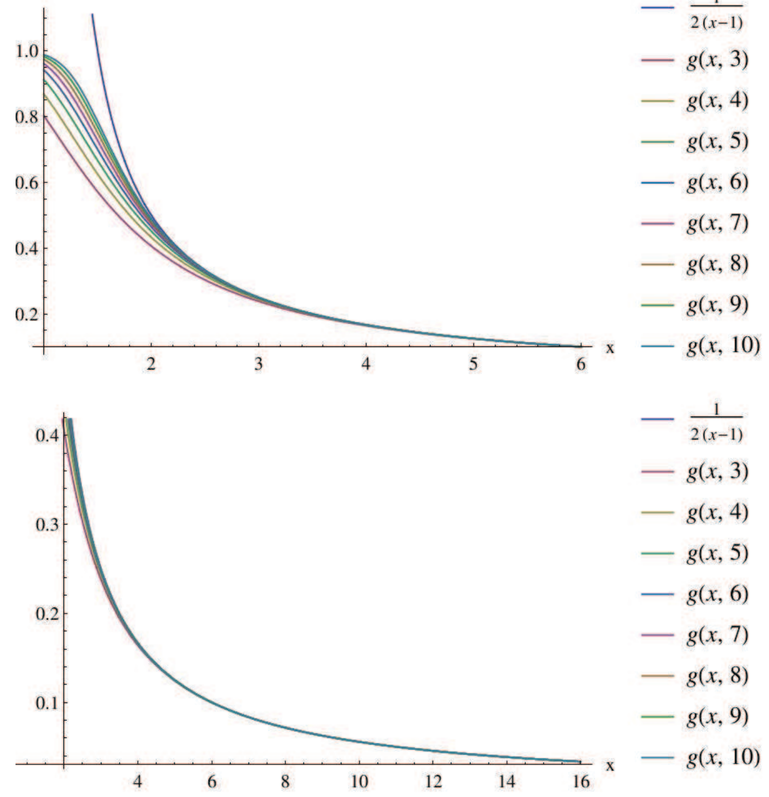

Figure 2: For this figure we set $\xi=2$, as used in VASP and other planewave DFT codes. The top graph shows the $g(x,n)$'s for $x\in[1,6]$, the differences are quite noticeable on the $[1,2]$ interval. As x becomes larger, all the $g(x,n)$'s approximate $\frac{1}{2(x-1)}$ well. The differences among the functions are already hard to observe on the $[4,16]$ interval, as shown in the bottom graph.

The proof is straightforward by a coordinate translation and then applying Theorem 2.1 and Theorem 3.1, noticing that the coefficients defined in (3.2) satisfy the condition (3.1) in Theorem 3.1.

We plot the functions $g_\xi(x,n)$, $n=2,\cdots,9$, for the simple case $x_0=0$ and $\xi=2$ on the $[0,4]$ interval in Fig. 3. The interval length is chosen such that the differences of the functions are easy to visualize.

As seen from Fig. 3, the $g_\xi(x,n)$ family of functions clearly satisfy the two "preconditioning" properties (i) and (ii) in the planewave setting, which need to keep the function value at lower x unchanged and reduce the function value at higher x, as required in planewave DFT calculations (e.g., [4,5,10,16,21]).

Although the term "preconditioning" has been used frequently for eigenvalue problems in the planewave DFT setting, the exact meaning of the preconditioning seems still not well-understood. A basic question is, what conditioning does the "preconditioning" functions (including the $K(x)$) try to improve? This is to be contrasted with the better

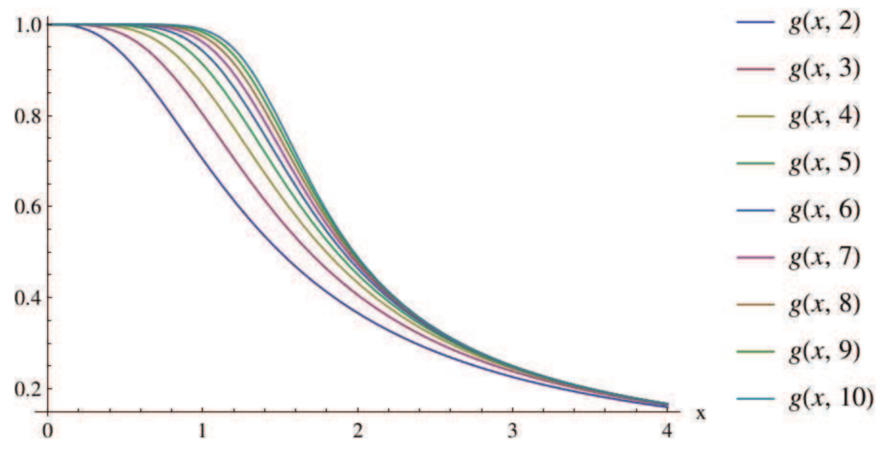

Figure 3: For this figure we again set $\xi = 2$ and label $g_\xi(x,n)$ as $g(x,n)$. As seen from this graph, for the $x<1$ region, the $g(x,n)$ becomes flatter as n increases. This is due to the higher order zero derivatives at $x=0$. Although the $\frac{1}{2(x-1)}$ is not plotted here, we know that for each fixed n, the $g(x,n)$ becomes better approximation to this function as x increases.

known preconditioning for solving linear equations, (e.g. [2,14]), for which we know the purpose of a preconditioner is to improve the condition number of the coefficient matrix for the linear equation.

The meaning of "preconditioning" for a stand-alone eigenproblem is also quite different from the concept of preconditioning for solving linear equations. That is why we quoted the word "preconditioning" when referring to eigenvalue calculations. The condition number for an eigenvalue or an eigenvector of a standard hermitian eigenproblem is a well-defined concept [20], to improve the conditioning of an eigenvector one needs to increase the gap between its associated eigenvalue and the other eigenvalues. Therefore a good "preconditioner" should be a function that can introduce a significant gap between the wanted part and the unwanted part of the spectrum. Chebyshev polynomials fit this purpose well, thus they have been used as effective "preconditioners" (or spectrum filter) [1,11,13,22,24,25]. More details on "preconditioned" eigenvalue algorithms are presented in [23].

If the x in $g_\xi(x,n)$ may be considered as the "spectrum", then the shape of the $g_\xi(x,n)$ as shown in Fig. 3 does nicely fit the purpose of introducing a significant gap between wanted (lower end) and the unwanted (higher end) of the spectrum, thus improving the conditioning of eigenvectors. This would explain why the $g_\xi(x,n)$'s are very effective in accelerating convergence of the planewave DFT eigenproblems. However, in the planewave DFT setting, the planewave basis is usually written as $e^{\imath \mathbf{G}\cdot\mathbf{r}}$ [8,12,15], where the wave vector $\mathbf{G}$ corresponds to the x in $g_\xi(x,n)$. Thus the x may not be directly considered as the "spectrum", and a better way is still needed to understand what conditioning the "preconditioning" functions try to improve.

5 Numerical tests

In this section we test the numerical performance of the "preconditioning" functions. The functions are used as the preconditioners inside a typical preconditioned-Davidson method.

The numerical tests are done in Matlab, in a simplified planewave DFT setting in which a Gaussian-type pseudopotential is used. For each of the tests shown in Figs. 4-6, we fix the Hamiltonian matrix and compute the lowest few eigenpairs only once. That is, we do not perform a self-consistency loop. The reported total number of iterations and the total number of matrix-vector products correspond to the cost of solving an eigenvalue problem in one SCF step.

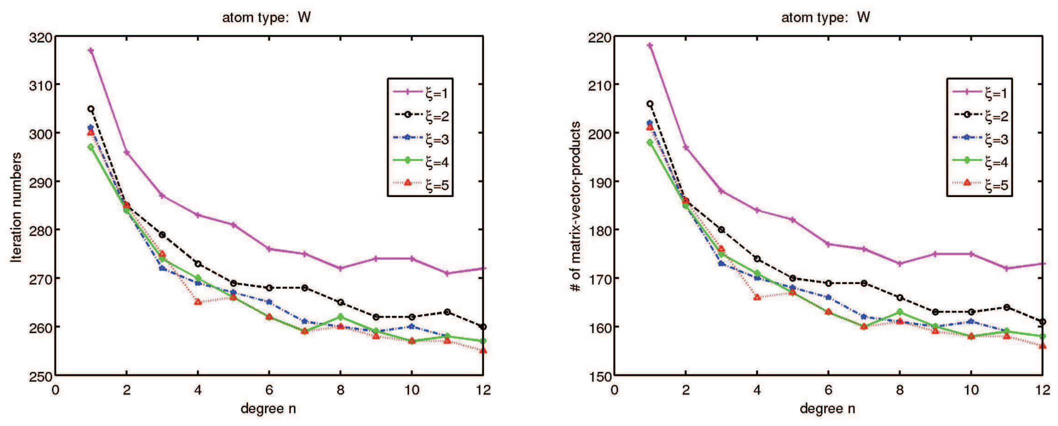

Figure 4: Total number of iterations (left) and total number of matrix-vector products (right) used for converging the lowest 100 eigenpairs. Lattice constant is determined by a Tungsten (W) atom, the number of planewave basis is 10000.

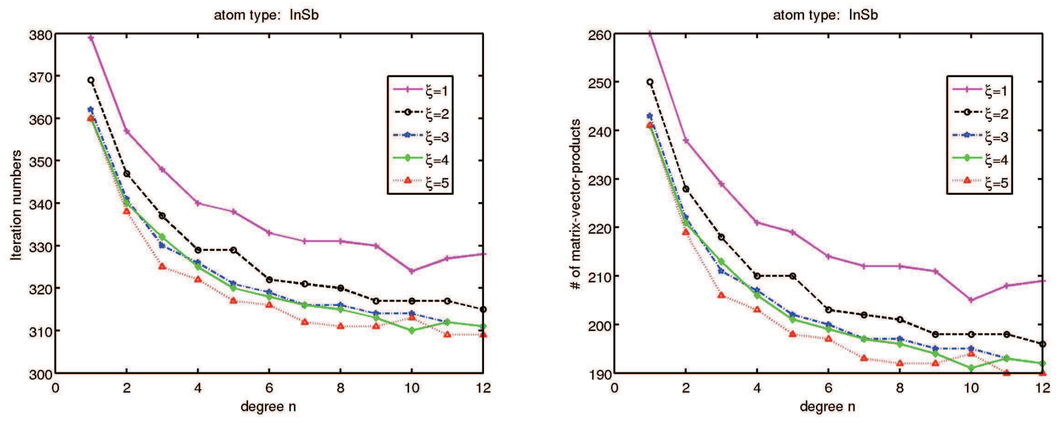

Figure 5: Total number of iterations (left) and total number of matrix-vector products (right) used for converging the lowest 120 eigenpairs. Lattice constant is determined by an InSb atom, the number of planewave basis is 11000.

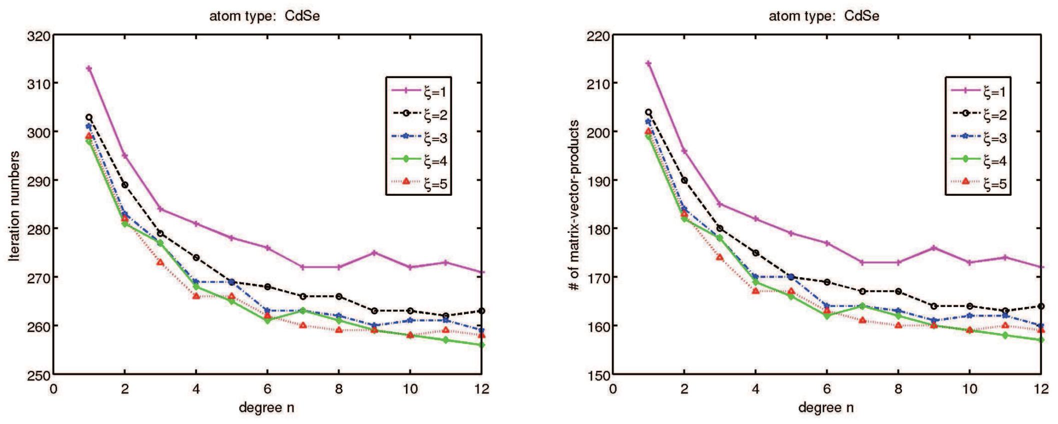

Figure 6: Total number of iterations (left) and total number of matrix-vector products (right) used for converging the lowest 100 eigenpairs. Lattice constant is determined by a CdSe atom, the number of planewave basis is 10000.

Figs. 4-6 show the performance of using different "preconditioning" function $g_\xi(x,n)$ as defined in (4.2). The coefficients for all the tests are constructed based on (3.2). For simplicity we set $x_0=0$ in (4.2), thus the plots for the $n=3,\xi=2$ case correspond to using the TPA preconditioner $K(x)$ as listed in (1.1).

As seen from Figs. 4-6, when ξ is fixed, the total computational cost generally decreases with a larger n, i.e., when higher order functions are used. While for a fixed order n, increasing ξ to be larger than 2 also contributes to reducing the computational cost. These results suggest there are indeed some benefits using formulas with higher order of approximation accuracy.

It is interesting to note that the number of matrix-vector products for these three tests are all smaller than the number of total iterations. The reason for this is that the Davidson code we use is a non-block version without deflation, and that the "preconditioning" by $g_\xi(x,n)$ is very effective. Therefore, only the lowest few eigenvalues take several iterations to converge, while the basis of the projection subspace quickly become a good approximation of the wanted invariant subspace, resulting in many of the larger eigenvalues to converge in just one to two iterations, thus many of them would require none or just one matrix-vector product to converge. If a block version of Davidson method is used, then the total iteration number should go down, because the effectiveness of the "preconditioning" function can cause several eigenvalues to converge at a single iteration step.

From Theorem 4.1 we know that the "preconditioning" function $g_\xi(x,n)$ as defined in (4.2) has two significant properties: (i) $\frac{\partial^i}{\partial x^i} g_\xi(x_0,n)=0$ for $i=1,\cdots,n$, and (ii) $g_\xi(x,n)$ asymptotically approximates $\frac{1}{\xi(x-x_0-1)}$ correct up to the $(n+1)$-th order in $\frac{1}{x-x_0}$ for $x>x_0+1$. An interesting question worth asking is the relative significance of these two properties, that is, which property contributes more significantly to the speedup in convergence?

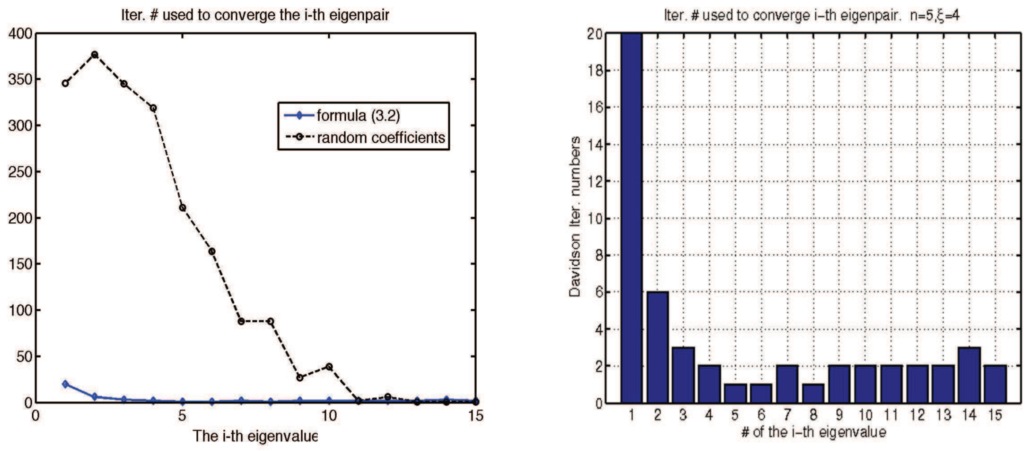

Figure 7: (Left) Iteration numbers used to converge each of the i-th eigenpair, where the lattice constant is determined by an InSb atom, the number of planewave basis is 2001, and the lowest 15 eigenpairs are computed. For the two schemes, the parameters used are $n=5$ and $\xi=4$. The relative shape of the two curves are quite similar for other parameters n and ξ, that is, for the smallest few eigenvalues, the scheme using formula (3.2) is over an order of magnitude faster than the scheme using random coefficients. (Right) Bar plot of the iteration number used to converge each of the 15 eigenpair with formula (3.2).

We do not have a direct answer to this question. However, it is quite straightforward to design a numerical test to show that a "preconditioning" function with only property (i) is not enough for the acceleration.

For simplicity we again set $x_0=0$. The result in Theorem 2.1 says that $\frac{\partial^i}{\partial x^i}g_\xi(0,n)=0$ hold true for all $i=1,\cdots,n$, independent of the coefficients c_i. Therefore, using random coefficients in (4.2) will make $g_\xi(x,n)$ satisfy property (i). In Fig. 7 we compare the results of using random coefficients with those using coefficients generated by formula (3.2). Since the scheme using random coefficients can be very slow, we only try it on a small problem. It is evident from Fig. 7 that "preconditioning" with a function having only the better known property (i) is not enough, it is the combined effect of properties (i) and (ii) that leads to faster convergence. The second plot in Fig. 7 may also been used to help explain why the number of matrix-vector products is smaller than the number of iterations in Figs. 4-6.

6 Concluding remarks

We develop some generalization of the famous TPA "preconditioning" function. The derivation of the higher order $g_\xi(x,n)$ is purely mathematical and seemingly lacks any physical insight. We argue that the necessary physical insight is already utilized in constructing the TPA function $K(x)$, while the $g_\xi(x,n)$'s are derived mainly based on $K(x)$, thus they inherit the desirable physical insight used in $K(x)$. One particularly important property is that $K(x)$ as well as all $g_\xi(x,n)$ try to approximate a function that falls off like $1/x$ when $x \to \infty$. In contrast, we also constructed functions that approximate a function that falls off like $1/x^2$ when $x \to \infty$, but the convergence speedup is far inferior to using functions that approximate $1/x$, thus the formula for approximating $1/x^2$ is not presented in this note. Whether there is a better function than $1/x$ to approximate as $x \to \infty$ is a question that requires some further investigation.

The general formula $g_\xi(x,n)$ is demonstrated to provide some improvement over the TPA function $K(x) = g_2(x,3)$ in Section 5. The improvement is likely more noticeable when the number of wanted eigenpairs becomes larger.

Other expected applications of the formula $g_\xi(x,n)$ include the approximation of the Fermi-Dirac distribution [7, p.582], and as spectrum filters for large-scale eigenvalue calculations.

Acknowledgments

Y. Zhou is supported in part by the NSF under grant DMS-1228271 and by a J.T. Oden fellowship from the University of Texas at Austin. J. R. Chelikowsky acknowledges support provided by the 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 under award number DE-SC0008877. X. Gao is supported in part by the NSF of China under grant 61300012 and the Defense Industrial Technology Development Program. A. Zhou is supported in part by the Funds for Creative Research Groups of China under grant 11321061, the National Basic Research Program of China under grant 2011CB309703, and the National Center for Mathematics and Interdisciplinary Sciences, Chinese Academy of Sciences.

We thank Bolin Liao for sharing his planewave DFT code, the numerical tests were based on modifications to the TPA preconditioner in his code. We thank Ren-cang Li for pointing out a simpler proof of the first part of Theorem 2.1, as listed on page 169. We thank the referees for the critical and insightful comments, which helped to improve the presentation.

References

[1] A. S. Banerjee, R. S. Elliott, and R. D. James. A spectral scheme for Kohn-Sham density functional theory of clusters. *ArXiv e-prints*, arXiv:1404.3773, 2014.

[2] M. Benzi. Preconditioning techniques for large linear systems: A survey. *J. Comput. Phys.*, 182(2):418–477, 2002.

[3] B. Engel and R. M. Dreizler. *Density Functional Theory: An Advanced Course*. Theoretical and Mathematical Physics. Springer, 2011.

[4] P. Giannozzi *et al.* . QUANTUM ESPRESSO: A modular and open-source software project for quantum simulations of materials. *Journal of Physics: Condensed Matter*, 21(39):395502 (19pp), 2009.

[5] X. Gonze *et al.* . ABINIT : First-principles approach of materials and nanosystem properties. *Computer Phys. Commun.*, 180:2582–2615, 2009.

[6] P. Hohenberg and W. Kohn. Inhomogeneous electron gas. *Phys. Rev.*, 136:B864–871, 1964.
[7] E. Kaxiras. *Atomic and Electronic Structure of Solids*. Cambridge University Press, 2003.
[8] J. Kohanoff. *Electronic Structure Calculations for Solids and Molecules: Theory and Computational Methods*. Cambridge Univ. Press, 2006.
[9] W. Kohn and L. J. Sham. Self-consistent equations including exchange and correlation effects. *Phys. Rev.*, 140:A1133–1138, 1965.
[10] G. Kresse and J. Furthmüller. Efficient iterative schemes for *ab initio* total-energy calculations using a plane-wave basis set. *Phys. Rev. B*, 54(16):11169–11186, 1996.
[11] A. Levitt and M. Torrent. Parallel eigensolvers in plane-wave density functional theory. *Comp. Phys. Comm.*, 187:98–105, 2015.
[12] R. M. Martin. *Electronic Structure : Basic Theory and Practical Methods*. Cambridge University Press, 2004.
[13] P. Motamarri and V. Gavini. A subquadratic-scaling subspace projection method for large-scale Kohn-Sham density functional theory calculations using spectral finite-element discretization. *ArXiv e-prints*, arXiv:1406.2600, 2014.
[14] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, second edition, 2003.
[15] D. Sholl and J. A. Steckel. *Density Functional Theory: A Practical Introduction*. Wiley-Interscience, 2009.
[16] M. P. Teter, M. C. Payne, and D. C. Allan. Solution of Schrödinger's equation for large systems. *Phys. Rev. B*, 40:12255–12263, 1989.
[17] ABINIT webpage. `http://www.abinit.org/`.
[18] Quantum ESPRESSO webpage. `http://www.quantum-espresso.org/`.
[19] VASP webpage. `http://cms.mpi.univie.ac.at/vasp/`.
[20] J. H. Wilkinson. *The Algebraic Eigenvalue Problem*. Oxford University Press, 1965.
[21] C. Yang, J. C. Meza, B. Lee, and L.-W. Wang. KSSOLV — a MATLAB Toolbox for Solving the Kohn-Sham Equations. *ACM Trans. Math. Softw.*, 36(2):10:1–35, 2009.
[22] Y. Zhou. A block Chebyshev-Davidson method with inner-outer restart for large eigenvalue problems. *J. Comput. Phys.*, 229(24):9188–9200, 2010.
[23] Y. Zhou and R.-C. Li. On the essence of "pre-conditioned" eigen-algorithms. (to be submitted).
[24] Y. Zhou and Y. Saad. A Chebyshev-Davidson algorithm for large symmetric eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 29(3):954–971, 2007.
[25] Y. Zhou, Y. Saad, M. L. Tiago, and J. R. Chelikowsky. Parallel self-consistent-field calculations using Chebyshev-filtered subspace acceleration. *Phys. Rev. E*, 74(6):066704, 2006.