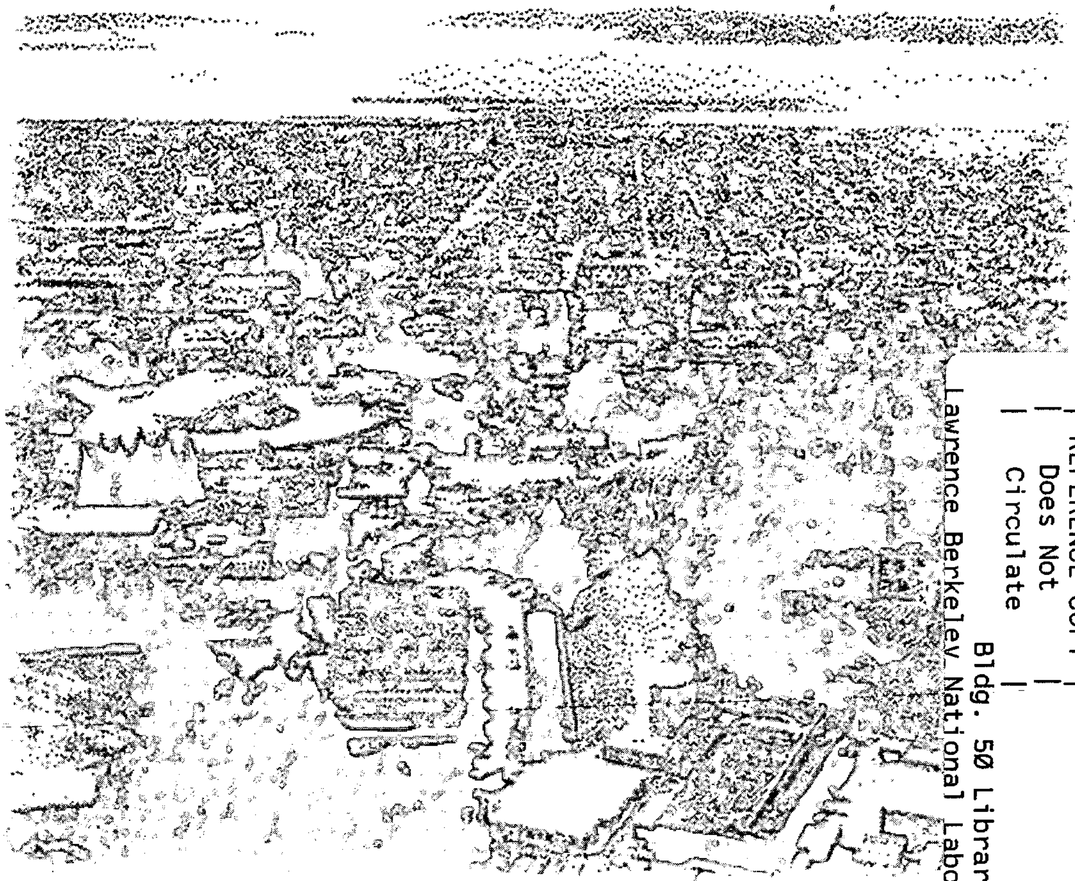
ERNEST ORLANDO LAWRENCE
BERKELEY NATIONAL LABORATORY

Dynamic Restarting Schemes for Eigenvalue Problems

Kesheng Wu and Horst D. Simon

Computing Sciences Directorate
National Energy Research
Scientific Computing Division

March 1999



LBNL-42982

Dynamic Restarting Schemes for Eigenvalue Problems

Kesheng Wu and Horst D. Simon

Computing Sciences Directorate
Ernest Orlando Lawrence Berkeley National Laboratory
University of California
Berkeley, California 94720

March 1999

This work was supported by the Director, Office of Science, Office of Laboratory Policy and Infrastructure Management, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

Dynamic Restarting Schemes For Eigenvalue Problems[†]

Kesheng Wu[‡] Horst D. Simon[‡]

March 10, 1999

Abstract

In studies of restarted Davidson method, a dynamic thick-restart scheme was found to be excellent in improving the overall effectiveness of the eigenvalue method. This paper extends the study of the dynamic thick-restart scheme to the Lanczos method for symmetric eigenvalue problems and systematically explore a range of heuristics and strategies. We conduct a series of numerical tests to determine their relative strength and weakness on a class of electronic structure calculation problems.

The Lanczos method is an effective method for computing extreme eigenvalues and the corresponding eigenvectors of large matrices. In order to limit the maximum memory usage and reduce arithmetic operations per iteration, it is often restarted. In this case, the user specifies a maximum number of Lanczos vectors to be used, say m. After m Lanczos vectors are generated, the approximate solutions are computed and evaluated. If they are not acceptable, a restarting scheme is used to extract the most important information and the information is used in the Lanczos iterations to again generate m Lanczos vectors. The most straightforward way to start the Lanczos algorithm is to give it one starting vector. Earlier restarting schemes are derived based on this observation. Variations of this scheme include restarting with one Ritz vector, restarting with a linear combination of Ritz vectors and so on [12]. A much more effective scheme named the implicit restarting scheme was discovered by Sorensen in 1992 [13]. One important characteristics of this scheme is that it allows an arbitrary number of vectors to be saved at restart. Another restarting scheme with similar features is the thick-restart scheme [15] whose history can be traced back to earlier versions of the Davidson method [6]. However, this feature of restarting with arbitrary number of starting vectors are not fully exploited until recently. To contrast with the implicit restarting scheme, the thick-restart scheme is often called an explicit restarting scheme.

Both the implicit restarting scheme and the explicit restart scheme allow one to improve the effectiveness of a restarting method. The choice of exactly what and how many vectors to

[†]This work was supported by the Director, Office of Science, Office of Laboratory Policy and Infrastructure Management, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy.

[‡]Lawrence Berkeley National Laboratory/NERSC, Berkeley, CA 94720. Email: {kwu, hdsimon}@lbl.gov.

save is one of the determining factors in the overall effectiveness of the eigenvalue methods. The restarting schemes discussed in this paper refer the strategies of making this choice. More specifically, this paper studies a number of heuristics for deciding what and how many vectors to save in the thick-restart Lanczos method for symmetric eigenvalue problems [16]. Because the strategies can not be compared analytically, we have chosen to compare them using a small set of test problems. Through this set of tests, we are going to identify some efficient schemes for a type of eigenvalue problem arise from the electronic structure calculations. Through our effort, we also hope to identify clearly unsound choices and narrow the search range for future users.

This paper is organized as follows. We start by describing the thick-restart Lanczos method in Section 1. The test problems used are described in Section 2. Section 3 contains the basic rationale behind the different restarting strategies and give a brief overview of what analytical tools are available for devising restarting strategies. The main body of the text, sections 4 and 5, contains the details of how to implement the four main restarting strategies and how to modify them in order to achieve better performance. In section 6, we summarize the observations made in sections 4 and 5, suggest how the four main strategies should be implemented and demonstrate their effectiveness on a large test problem.

1 Thick-restart Lanczos method

Many well known methods for eigenvalue problems, such as the Lanczos method [10], the Arnoldi method [12], and the Davidson method [5], have to be restarted in large scale applications either to reduce the computer memory usage or to reduce the arithmetic operations per iteration. For convenience of discussion, an iteration of the restarted method is this paper includes all operations associated with one matrix-vector multiplication. One consequence of restating these methods is that the restarted versions may take considerably more iterations to reach convergence compared to their non-restarted counterparts. An effective restarting strategy is crucial to reduce the number of iterations. In this paper, we will limit ourselves to study only real symmetric or complex Hermitian eigenvalue problems for which the Lanczos method is the most effective method. Previously, the implicit restarting scheme has been used with the Lanczos method [1, 2]. In this paper, we will study the thick-restart Lanczos method. For convenience of discussion, we briefly describe the the two major components of the thick-restart Lanczos method, the Lanczos iterations to extend the basis and the restarting procedure.

Given a matrix A, its eigenvalue λ and the corresponding eigenvector x are defined by equation $Ax = \lambda x$. The Lanczos method computes approximate values to λ and x which are also called λ and x. If there are m Lanczos vectors, they will be denoted by $q_1, \ldots, q_m$. In the process of computing m Lanczos vectors, the algorithm will also compute $\alpha_i, \beta_i, i = 1, \ldots, m$ which are used later in the Rayleigh-Ritz projection. Here is a brief description of the algorithm.

Initialization

To start solving a new eigenvalue problem, take a starting vector, normalize it and store the resulting vector as q_1. Set k to zero.

When restarting, k is set by the the restarting procedure which also provides $q_1, \ldots, q_k, q_{k+1}$, $\alpha_1, \ldots, \alpha_k$, and $\beta_1, \ldots, \beta_k$.

Iterate

```
For $i = k+1, \ldots, m$,

    1. $q_{i+1} \leftarrow Aq_i$,
    2. $\alpha_i \leftarrow q_i^T q_{i+1}$,
    3. If $i = k+1$,
           $q_{i+1} \leftarrow q_{i+1} - \alpha_i q_i - \sum_{j=1}^{k} \beta_j q_j$,
       else
           $q_{i+1} \leftarrow q_{i+1} - \alpha_i q_i - \beta_{i-1} q_{i-1}$.
    4. $\beta_i \leftarrow \|q_{i+1}\|$, $q_{i+1} \leftarrow q_{i+1}/\beta_i$.
```

This short description captures the essence of the algorithm. We have ignored the details for dealing with finite precision arithmetic in particular the re-orthogonalization procedure [16] because they are not directly relevant to the restarting strategies to be discussed. The following equations summarize the relation among the Lanczos vectors produced by this algorithm, ($Q_i = [q_1, \ldots, q_i]$)

$$
\begin{aligned}
AQ_i &= Q_i T_i + \beta_i q_{i+1} e_i, \qquad (i > k) & (1)\\
T_i &\equiv \begin{pmatrix}
\alpha_1 & & & \beta_1 & & & & & \\
& \ddots & & \vdots & & & & & \\
& & \alpha_k & \beta_k & & & & & \\
\beta_1 & \cdots & \beta_k & \alpha_{k+1} & \beta_{k+1} & & & & \\
& & & \beta_{k+1} & \alpha_{k+2} & \beta_{k+2} & & & \\
& & & & \ddots & \ddots & \ddots & & \\
& & & & & \beta_{i-2} & \alpha_{i-1} & \beta_{i-1} \\
& & & & & & \beta_{i-1} & \alpha_i
\end{pmatrix}. & (2)
\end{aligned}
$$

At the initialization step, i.e., ($i = k$), the following relation must be satisfied,

$$Aq_i = \alpha_i q_i + \beta_i q_{k+1}, i = 1, \ldots, k. \qquad (3)$$

The value k in the above formula is called the thickness in this paper. The simplest way to satisfy this relation is to supply the algorithm with one starting vector q_1 ($k = 0$). In the thick-restart Lanczos method, the restarting procedure produces k orthogonal vectors satisfying the above equation which allows it to use arbitrary number of starting vectors. The main steps of the restarting scheme can be described as follows.

1. Compute an eigen-decomposition of T_m, $T_m \equiv YDY^T$. As in the Rayleigh-Ritz projection, the diagonal elements of D are the eigenvalues of T_m and the approximate eigenvalues (the Ritz values) of A.

   If we only want to perform convergence test, it is possible to only compute the last row of Y, which will reduce the need of both the memory and the arithmetic operations.

2. Based on available information, decided what and how many Ritz values are to be saved. In our program, we order the eigenvalues of T_m in ascending order and the entire decision reduces to pick two integers k_l and k_r which indicate that $d_{1,1}, \ldots, d_{k_l,k_l}$ and $d_{k_r+1,k_r+1}, \ldots, d_{m,m}$ are to be saved. This paper is about how to choose k_l and k_r such that the whole eigenvalue program takes the least amount of time.

3. Let $k = k_l + m - k_r$. Denote the Ritz values to be saved as $\hat{\alpha}_1$, $\hat{\alpha}_2$, ..., $\hat{\alpha}_k$, and the corresponding columns of Y as y_1, ..., y_k. The Ritz vectors can be computed as $\hat{q}_1 = Q_m y_1$, ..., $\hat{q}_k = Q_m y_k$, and $\hat{q}_{k+1} \equiv q_{m+1}$. In addition, $\hat{\beta}_1 = \beta_m y_{m,1}$, ..., $\hat{\beta}_k = \beta_m y_{m,k}$.

This algorithm generates Ritz pairs as in the standard Rayleigh-Ritz projection. The difference is in what Ritz values are actually computed. In the standard Rayleigh-Ritz projection, the number of Ritz pairs to be computed is the number of eigenpairs wanted. If the smallest eigenvalues are wanted, than the smallest Ritz values are saved. If the largest eigenvalues are wanted, than the largest Ritz values are saved. In the thick-restart procedure, we typically save some largest ones and some smallest ones no matter which end of the spectrum we are interested in, and usually more Ritz pairs are saved than the standard case.

Because the matrix T_m is not tridiagonal in the thick-restart Lanczos algorithm, more arithmetic operations are need to compute an eigenvalue decomposition for it. If the basis size m is relatively small, the extra amount of arithmetic will be negligible compared to other operations in the restarted Lanczos algorithm. For this reason, we will not discuss this issue further.

2 Test problems

Through out this paper we will use a small number of test problems repeatedly. They are described in this section.

The three test problems listed in Table 1 come from two sources. The matrix si4 and si6 are from simulation of electronic properties of silicon atom clusters [3, 4]. These two real symmetric matrices are generated from the first step of the Self-Consist Field (SCF) iterations. They are relatively small so we can perform a large number of tests without consuming significantly amount of computer time. During our tests, we always compute the 12 smallest eigenvalues and the corresponding eigenvectors of si4 test problem and the 16 smallest eigenvalues and the corresponding eigenvectors for si6 test problem. Test problem InGaAs9k is generated from simulation of a 9000-atom InAs quantum dot in an GaAs

Table 1: Information about the test problems.

NAME	N	NNZ	Comment
si4	4451	84918	*Ab Initio* simulation of a four-silicon cluster
si6	7949	151524	*Ab Initio* simulation of a six-silicon cluster
InGaAs9k	137919	(full)	empirical pseudopotential simulation of an InGaAs quantum dot

surrounding [17]. This test problem has a complex Hermitian matrix which is not stored explicitly. The matrix-vector multiplication is performed through Fast Fourier Transformations (FFT).

These three test problems are chosen because the authors are involved in projects that produce similar matrices. The selection of the test problems is small. However, by restricting to these problems, we are able to perform a more thorough analyses of the different restarting strategies which may ultimately reveal more about the restarting strategies.

Since all test problems compute the smallest eigenvalues, we describe the restarting strategies based on finding the smallest eigenvalues. It should be straightforward to extended it to the case of finding the largest eigenvalues. When computing the smallest eigenvalues, the simplest thing to do is to always save a fixed number of the smallest Ritz pairs. This simple restarting scheme is called the fixed-thickness scheme in this paper. To measure the dynamic restarting schemes, we will conduct a series of tests to determine the optimal thickness for the fixed-thickness scheme. The tests are run with the starting vector $[1, 1, \ldots, 1]^T$. The Ritz pairs are declared converged if their residual norms are less than $\sqrt{\epsilon}||A||$, where ϵ is the unit round off error which is about 2.2×10^{-16}, and the norm of the matrix ($||A||$) is estimated by the largest (absolute) Ritz value ever computed in the Lanczos method. All future tests will be performed using the same starting vector and convergence tolerance.

For the two smaller test problems, si4 and si6, we have conducted the tests with three different basis size $m = 20, 50, 100$. The optimal thickness based on either the time or the number of matrix-vector multiplications are listed in Table 2. In the table, the number of matrix-vector multiplications is denote by MATVEC. The top half of Table 2 shows results that use the minimal number of matrix-vector multiplications and the bottom half of the table shows results that use the minimal amount of computer time. These results are obtained by trying all possible values of k under each given m ($n_{eig} \leq k \leq m-3$). We would like to devise a set of strategies that can automatically choose an appropriate thickness that performs no worse than results achieved here.

The timing results in Table 2 are measured on a SGI Onyx 2 running at 195 MHz. All tests involving si4 and si6 are run on this machine. Tests involving the quantum dot problem will be run on a Cray T3E parallel machine to provide a different prospective.

We have conducted similar experiment with the quantum dot test problem to compute the five smallest eigenvalues and the corresponding eigenvectors. Figure 1 shows the time and the number of matrix-vector multiplications used to solve the InGaAs9k test problem with different fixed thickness. A basis size (m) of 25 is used in this test. The timing results

Table 2: The minimal time and number of matrix-vector multiplications used to solve two silicon cluster test problems using thick-restart Lanczos method with fixed thickness ($k = k_l, k_r = m$).

minimal number of MATVEC						
	$m = 20$		$m = 50$		$m = 100$	
	MATVEC	k	MATVEC	k	MATVEC	k
si4	488	16	274	34	268	44[a]
si6	1621	16	274	22	271	43
minimal time (seconds)						
	$m = 20$		$m = 50$		$m = 100$	
	time	k	time	k	time	k
si4	5.18	12	3.19	19	4.59	14
si6	50.0	16	7.90	16	11.9	42

[a]268 MATVEC is also used when k is 58 and 72.

are obtained on 32 processors of a Cray T3E 900. The optimal thickness in this case is 12 which achieves both minimal number of matrix-vector multiplications (1806) and the minimal amount of CPU time (179.6 sec). This is a much large test problem than the two silicon cluster ones and the matrix-vector multiplications take up a much large portion of the total time too. Because the matrix-vector multiplications dominate the overall time, minimizing the number of iterations also minimizes the total time for this test problem.

Now that we have established the performance target for the test problems, next we will exam what can be used to guide our choice of restarting strategies.

3 Rationale for the heuristics

In our version of the thick-restart scheme, see page 3, the decision to be made is to choose two integers k_l and k_r, see also Figure 2. This section will review the theoretical tools that can guide us in making this decision. We will see how they are used and why additional heuristics are needed.

There are two theoretical tools that can be used to analyze the choices, *the polynomial filter* and *the approximate deflation.* The bases vectors generated by the implicitly restarted Arnoldi method and the thick-restart Lanczos method are always orthogonal bases of some Krylov subspace, $K(A, v)$ [13, 16]. The starting vector v changes after each restart. The polynomial filter refer to the relationship between these starting vectors where the vector v before and after a restart is related by a polynomial of the form $\Pi_{i=1}^{m-k}(A - \delta_i I)$. The scalar values δ_i are called the shifts. In the implicitly restart Lanczos method, they can be arbitrarily chosen. In the thick-restart Lanczos method, they are the Ritz values discarded during restart. Based on this polynomial relation, the optimal choice for the shifts are the Leja points [1]. The polynomial filter argument has strong theoretical foundation and programs based on this mechanism are found to be effective in practice [1]. However, this

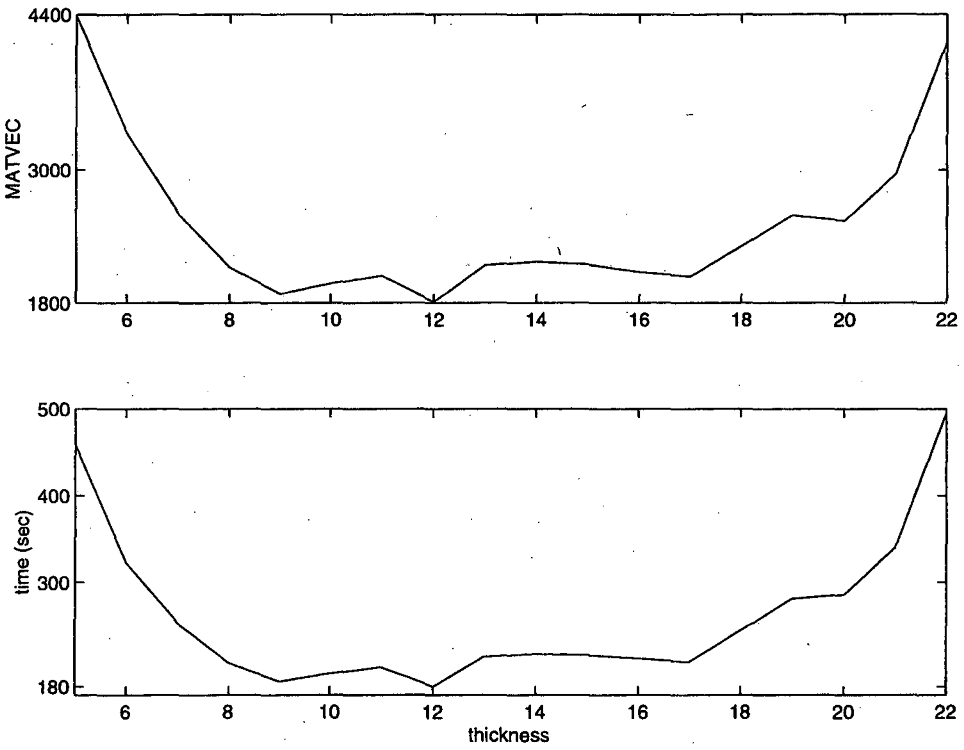


Figure 1: The time (seconds) and number of matrix-vector multiplications used by the thick-restart Lanczos method to find the five smallest eigenvalues of the InGaAs9k test problem with different fixed thickness ($m = 25$).

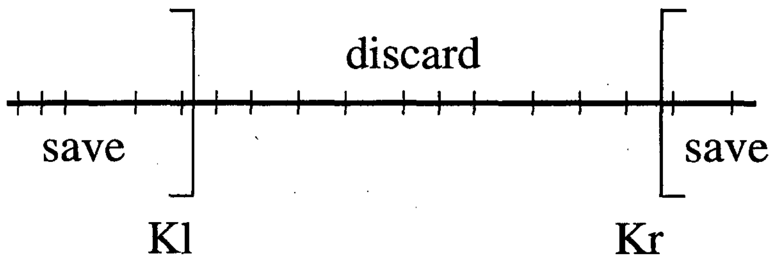


Figure 2: Schematics of selecting decision during thick-restart.

analysis does not give an clear indication of exactly how many shifts to apply or how many vectors to save when more than one eigenvalue is computed.

Another theoretical tool that can be used to guide the design of dynamic restarting heuristics is the *approximate deflation* feature of the Ritz vectors [9]. To compute eigenvalues near σ, the Ritz values near σ should be saved. Morgan's analyses indicate that the saved Ritz vectors approximately deflate the spectrum, increase the effective separation between the wanted eigenvalues and the rest of the spectrum, and consequently increase the convergence rate of the restarted method [9]. The dynamic thick-restart scheme used in the dynamic thick-restart Davidson method is successful example of using this argument [15].

Since the Lanczos method is only effective in computing the extreme eigenvalues, our implementation of the thick-restart scheme only save some largest Ritz values and some smallest Ritz values. When computing the smallest eigenvalue, the effective gap ratio used to devise the dynamic thick-restart scheme is [15]

$$\gamma = \frac{\lambda_{k_l} - \lambda_1}{\lambda_{k_r} - \lambda_1} \qquad (k_l < k_r \leq m). \tag{4}$$

Based on the approximate deflation argument, saving more nearby Ritz vectors will result in faster convergence rate for the smallest Ritz value. Obviously the maximum gap ratio is achieved when k_l is $k_r - 1$. However, in this case, the effective gap ratio is a wild overestimate of its actual value. In addition, when $k_l = k_r - 1$, an iteration of the restarted Lanczos algorithm is expensive because the Rayleigh-Ritz projection is performed after every matrix-vector multiplication and it always computes $m-1$ Lanczos vectors. Figure 1 clearly indicates that as k approaches m both CPU time and the number of iterations increase. Similar to the polynomial filter argument, the approximate deflation argument does not suggest an effective choice of how many Ritz pairs to saved either. The innovation of this paper is to augment these theoretical arguments with heuristics to make effective choices during restart.

To reduce the time and iterations in the dynamic thick-restart Davidson method, the developers of the dynamic thick-restart Davidson method require that $k_r \geq k_l + 3$ [15]. Because the function γ is a monotonic function of k_l and k_r, if no Ritz values are exactly equal to each other, this requirement leads to 3 Ritz pairs being discarded at every restart. The choice of always discarding 3 Ritz pairs and saving $m - 3$ is somewhat arbitrary. One way to remove this arbitrariness is to develop an empirical formula for deciding how many Ritz vectors should be saved. After each iteration, the residual norm is expected to decrease by a factor proportional to $e^{-\gamma}$ by the definition of γ [9]. Maximizing γ is equivalent to maximizing the residual norm reduction for each iteration. An alternative is to minimize the residual norm at the end of an entire restarted loop. If k Ritz pairs are saved, after $m - k$ iterations, the residual norm will decrease by a factor proportional to $e^{-(m-k)\gamma}$. Minimizing the residual norm at the next restart is equivalent to maximizing the quantity $\mu \equiv (m-k)\gamma$. It is clear that μ is not a monotonic function unless the Ritz values are exponential functions of their indices. Therefore maximizing μ should provide appropriate choices for k_l and k_r.

These approximate deflation based heuristics are relatively simple. Next we will see how well they actually work. There are also obvious limitations on these schemes. For example, the effective gap ratio are only meaningful if the saved Ritz values are close to the actual eigenvalues. Typical at least some of the saved Ritz values are not accurate, it might be helpful to take their errors into account. We will explore this and related issues in section 5 in order to enhance the robustness of our restarting strategies.

4 Implementing the heuristics

This section describes the details of how to implement the heuristics as actual computer programs. More specifically, we will concentrate on restarting choices based on individual heuristics. Here is a list of four heuristics that we plan to use.

1. Index based scheme – develop an empirical formula for deciding what are the appropriate values for k_l and k_r.
2. Residual norm based scheme – save Ritz pairs that are near the wanted ones and also have relatively small residual norms.
3. Maximizing the gap ratio γ.
4. Maximizing progress, i.e., finding k_l and k_r that maximize the value $\mu \equiv (m-k)\gamma$.

Typically, the restarted Lanczos algorithm is used to compute a number of eigenvalues at a time. Most of the heuristics require one Ritz pair being identified as the one currently being computed. This idea is similar to targeting in the Davidson method and we will also call the selected Ritz pair the target in this paper. When computing a number of smallest eigenvalues, the target is the smallest Ritz value that does not satisfy the convergence criteria. Clearly, other choices are possible. However, this simple choice appears to work reasonably well for the test problems. We will be using this choice throughout the rest of the discussion.

When deciding the parameters needed to make these heuristics into programs, we will use the two smaller test problems, si4 and si6. The objective of tuning these restarting schemes is to achieve the performance listed in Table 2.

4.1 Index based scheme

The rationale for this scheme is to save Ritz pairs near the wanted eigenvalues. The key here is to develop an reasonable formula that can achieve good overall performance. For simplicity, if the smallest eigenvalues are wanted, we only save the smallest Ritz values and their corresponding Ritz vectors. Using the simple formulae considered here, if there is only one eigenvalue to compute, this scheme will revert back to the fixed-thickness scheme. These formulae are based on the number of Ritz pairs already converged n_c, the number of eigenvalues wanted n_{eig}, and the basis size m. It differs from other three dynamic schemes in that it does not use information about the Ritz values or the residual norms. Given a maximum basis size m, if the thickness k is kept constant, the optimal value of k is often near $m/2$ for moderate size n_{eig} and m, see for example Figure 1. Based on this observation, our first formula for choosing k is

$$k = n_c + (m - n_c)/2. \tag{5}$$

	MATVEC			time (sec)		
	$m=20$	$m=50$	$m=100$	$m=20$	$m=50$	$m=100$
si4	536	282	301	4.65	3.29	7.21
si6	1561	282	403	34.9	8.18	18.7

The above table shows the iterations and time used by the thick-restart Lanczos method using this restarting strategy. Compared with the results in Table 2, the number of iterations (matrix-vector multiplications) and the time are close to the optimal values achieved with fixed thickness for basis sizes of 20 and 50. However, when the basis size is 100, significantly more time is used in this case. From the last column of Table 2 we see that $k = 40$ seems to be a good choice for both test problems. Base on this observation, we proposed to gradually vary k from $m/2$ to $2m/5$ as m/n_{eig} increases, for example,

$$k = n_c + (m - n_c)(\frac{2}{5} + \frac{n_{eig}}{10m}). \tag{6}$$

The time and iterations used by the thick-restart Lanczos method with this scheme of choosing k are

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	548	304	275	4.80	3.38	5.86
si6	1561	279	276	35.1	7.82	11.9

From this table, we see that choosing the thickness according to Equation 6 leads to better performance compared to using Equation 5 for si6, but not for si4. The iterations and time used by this choice of thickness are comparable to the results shown in Table 2 for most cases. Only in one case, solving si4 test problem with $m = 100$, the time used is significantly more than in the optimal fixed thickness case. Since the value of k that achieves minimal time is very close to n_{eig} for si4 test problem. We decide to test the following formula as well

$$k = n_c + n_{eig}. \tag{7}$$

The results of using this choice is as follows,

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	456	297	298	5.27	3.09	4.58
si6	1741	277	416	44.7	7.44	16.1

We see that this choice work well for $m = 50$ but not so well for smaller m and it also causes more time to be used for si6 test problem with $m = 100$. We have tested many other choices to predict the optimal k based on n_{eig}, n_c, m and parameters other then Ritz values or residual norms. None of them can consistently generate better performance than using Equation 6. We believe this is because one formula can not predict the optimal k values for the two test problems. This suggests that a robust strategy must take the spectrum information into consideration. For the moment, we accept Equation 6 as the formula to implement this strategy.

4.2 Mimicking ARPACK

The eigenvalue package ARPACK has an implementation of the implicitly restarted Lanczos method for symmetric eigenvalue problems [8]. The restarting scheme in ARPACK also determines how many vectors in a similar manner as described in previous subsection. Here we will briefly examine the scheme used in ARPACK and see how Equation 6 works in ARPACK.

Table 3: The time and number of matrix-vector multiplications used by ARPACK to solve the si4 and si6 test problems.

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	523	308	343	10.1	7.0	11.5
si6	3373	421	471	155.6	20.7	31.0

In version 2.4 (dated 07/31/96) of ARPACK, if there is no eigenvalue with zero residual norm, the selection of number of vectors to save is based on the following formula,

$$k = n_{eig} + \min(n_c, (m - n_{eig})/2). \tag{8}$$

In addition to the above formula, there is also a special case when $n_{eig} = 1$. Since we always compute more than one eigenvalue, the special case is not relevant to our test problems. By selecting k using the above equation, the thick-restart Lanczos method uses following iterations and time to solve the two silicon cluster test problems.

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	402	299	293	4.43	3.21	4.79
si6	2280	283	405	66.4	7.87	15.9

This imitation of ARPACK has very similar performance to the scheme depicted by Equation 7. The actual performance of ARPACK is shown in Table 3. Because of differences in the convergence test, our restarted Lanczos method does not use the same time or iterations[1]. When computing n_{eig} smallest eigenvalues, during convergence test, ARPACK performs the test on all n_{eig} smallest Ritz values and n_c is the count of how many have satisfied the convergence criteria. In our implementation of the restarted Lanczos method, we perform the convergence test on one Ritz pair at a time until one fails the test or all wanted ones have satisfied the convergence criteria. In other words, n_c is the size of the leading group of Ritz pairs that are converged. Because of this difference, the two convergence tests will report different n_c even if all the Ritz pairs are exactly the same. This difference causes the different number of Ritz pairs to be saved and ultimately causes the two method to behave differently.

To demonstrate that our restarting schemes can be easily used in ARPACK, we modify ARPACK (`dsaup2.f`) to use Equation 6 instead of Equation 8. The iterations and time used by this modified version of ARPACK are

[1]Both ARPACK and our thick-restart Lanczos program (TRLAN) are compiled with the same flags (`-mips4 -64 -Ofast=IP27 -OPT:alias=restrict`) and linked with the same libraries (`-L/usr/lib64 -lcomplib.sgimath`). The matrix-vector multiplications of the Compressed Sparse Row (CSR) matrices use the same function from SPARSKIT [11]. Examining the hardware event counters through `perfex` reveals that both TRLAN and ARPACK run at about the same speed (45MFLOPS for si4 $m = 100$) but ARPACK uses more floating-point operations (ARPACK 5.06×10^8 FLOP, TRLAN 2.41×10^8 FLOP, si4, $m = 100$).

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	450	308	343	6.47	6.01	10.2
si6	1338	427	471	45.6	18.3	28.4

In this test, the number of iterations used by ARPACK is significantly reduce when the basis size is small, see Table 3. When the basis size is large, the difference is small because the number of restarted loops are the same before and after the modification. Each ARPACK restarted loop builds a basis of size m before performing convergence test, a number of unnecessary matrix-vector multiplications were used before the last convergence test.

In this paper, we use the thick-restart Lanczos method to demonstrate that a good restarting scheme is useful. The brief digression here demonstrates that a good restarting scheme will benefit the implicitly restarted Lanczos method as well. In fact, this should be true for all restarted methods.

4.3 Save nearly converged Ritz pairs

This strategy tries to save the Ritz pairs that are close to the wanted eigenvalues and are also closer to convergence than an average Ritz pair. The main design choice here is what residual norms are small enough to be saved. To make the comparison concrete, we need to have reference values. One natural reference value is the maximum residual norm. Those Ritz pairs with similar residual norms probably should be ignored. As the reference value for what should be saved, we use the residual norm of the target Ritz pair. We have decided not to use the convergence criteria to determine this reference value because the convergence criteria may not always include an explicit condition on the residual norms, and even there is one the actual residual norms may always be significantly larger than the residual norm tolerance. With two reference values, now we can try to establish a formula for determining what residual norms are small enough to be saved.

Let r_{max} denote a residual vector with the largest norm and r_t be the residual vector of the target where t is its index. As usual, the Ritz values are in ascending order. When computing the smallest eigenvalues, we will save Ritz pairs $1, \ldots, k_l$ $(k_r = m)$ if $||r_i|| < s$, $(i = t+1, \ldots, k_l)$. The values of s is determined as

$$s = \max(\sqrt{r_{max} r_t}, 2r_{t+1}). \tag{9}$$

The value of s is usually $\sqrt{r_{max} r_t}$. To ensure that at least one additional Ritz pair beyond the target is saved, we added the term $2r_{t+1}$. During the actual search for k_l, we also make sure that $k_l \leq m - 3$. In addition, s must be less than $||r_{max}||$. There are two cases where s is greater or equal to $||r_{max}||$, $||r_t|| = ||r_{max}||$ or $2||r_{t+1}|| \geq ||r_{max}||$. In either case, we revert back to the strategy described in subsection 4.1.

We encapsulate all above conditions in a short program and use it in the thick-restart Lanczos method. The following table displays the time and matrix-vector multiplications used to solve the two smaller test problems.

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	548	304	275	4.87	3.37	6.02
si6	1561	279	276	35.1	7.74	12.1

The time and number of matrix-vector multiplications used to solve the two test problems are close to those using Equation 6. Solving si4 test problem with $m = 100$ again uses considerably more time than the optimal time shown in Table 2, it indicates that this residual norm based scheme has similar shortcomings as the previous one.

We attempted to use different formulae to compute s, however, none of them can vary the test results significantly. Thus, we decide to use Equation 9 for implementing this strategy.

4.4 Maximizing the effective gap ratio

The most straightforward way of implementing this strategy is to evaluate the gap ratio γ for all pairs of k_l and k_r and then select one pair that gives the maximum γ. Since γ is a monotonic function of k_l and k_r, there is no need to search through all possible combinations. In the implementation used for the dynamic thick-restart Davidson method, k_r is required to be larger than or equal to $k_l + 3$. In this case, we only need to compare different gap ratios by always setting $k_r = k_l + 3$, which significantly reduces the number of comparisons needed. The following table lists the time and matrix-vector multiplications used with this choice of k_l and k_r.

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	407	293	280	5.65	10.6	61.1
si6	1522	347	271	44.9	25.7	51.5

Similar to what is observed in dynamic thick-restart Davidson method, this particular implementation of dynamic thick-restart scheme is effective in reducing the number of matrix-vector multiplications but is not very effective in reducing the execution time of whole eigenvalue method. Table 4 shows the minimal time and iterations achieved if we first determine the determine the thickness then maximize γ. In this case, the minimal iterations are achieved with about the same thickness as those in Table 2 and the minimal times are achieved with slightly smaller k than those in Table 2. Needless to say, the optimal results achieved by using different k is considerably better than always save $m - 3$. The only exception is using $m = 20$ to compute 16 eigenvalues of si6 where both schemes save 17 Ritz vectors. In addition to first pick the thickness before maximizing γ, there are many ways to enhance the effectiveness of this strategy and we will consider them in the next section.

4.5 Maximizing μ

There is no free parameter in determining the maximum μ. We use a brute-force searching scheme to compare all pairs of k_l and k_r to find a pair that maximizes μ. In our implementation, we have the following restriction on k_l and k_r, $n_{eig} \le k_l \le k_r - 3$. The following table lists the test results of using this scheme.

Table 4: The minimal time and number of matrix-vector multiplications used to solve the two silicon cluster test problems by first deciding how many Ritz pairs to save and then choose those that maximize the effective gap ratio.

minimal number of MATVEC						
	$m = 20$		$m = 50$		$m = 100$	
	MATVEC	k	MATVEC	k	MATVEC	k
si4	443	17	288	36[a]	268	44[b]
si6	1522	17	274	32	271	43[c]
minimal time (seconds)						
	$m = 20$		$m = 50$		$m = 100$	
	time	k	time	k	time	k
si4	4.57	13	3.13	19	4.53	14
si6	44.9	17	7.87	22	11.8	43

[a]The k value of 43 can also achieve the minimum number of matrix-vector multiplications.

[b]The k value of 58, 72, 76, 79, 86, 88, 92, 93, 96 can also achieve the minimum number of matrix-vector multiplications.

[c]The k value of 81, 91, 97 can also achieve the minimum number of matrix-vector multiplications.

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	413	295	285	5.31	4.34	9.49
si6	1495	488	403	42.3	18.9	23.4

Compared to the previous case of maximizing the effective gap ratio with $k = m - 3$, this scheme uses more matrix-vector multiplications but less time. However, the results are not as good as the optimal results shown in Tables 2 or 4. We believe the main reason for this mediocre performance is that the effective gap ratio defined by the Equation 4 is not accurate when the saved Ritz values are far from the corresponding eigenvalues. For this reason, most of techniques used to enhance the scheme of maximizing γ can also be used to enhance this one.

5 Combining different schemes

In previous section, we have examined how to implement the four heuristics. Tests show that the individual heuristics works fairly well by themselves but they do not always lead to the "optimal" performances. The objective of this section is to explore a number of ways of combing the different heuristics to generate more robust strategies.

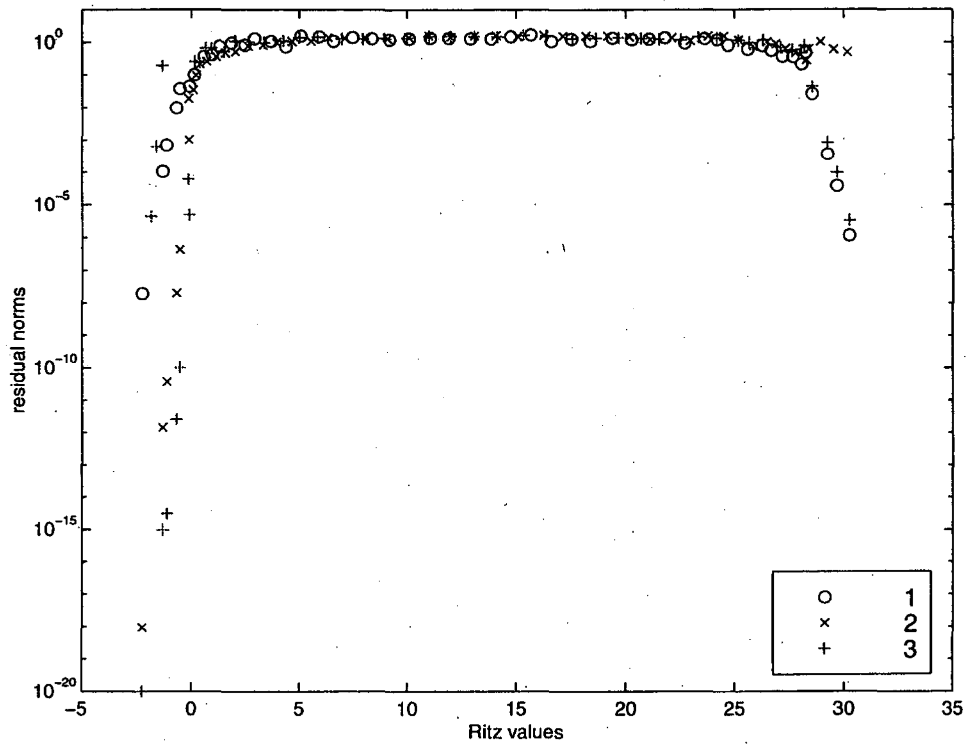


Figure 3: The residual norms of the Ritz pairs at the first three restarts when solving the si4 test problem.

5.1 Whether to save unwanted eigenvalues

Figure 3 shows the distribution of the residual norms against the Ritz values when solving si4 test problem. When restarting for the first time, the residual norms corresponding to the largest Ritz values are about the same size as those for the smallest ones. Since the Ritz pairs with largest Ritz values are discarded during the restart, the largest Ritz values computed from the second restarted loop are smaller than those computed from the first one and their corresponding residual norms are much large as well. However, after the third restarted loop, the largest Ritz values and their corresponding residual norms are almost exactly the same as those from the first. Since we don't want the largest eigenvalues, this repeated computation appears to be a waste of computing effort. A simple alternative to discarding them is saving them. The goal of this subsection is to find out whether or not it is worthwhile to save these unwanted Ritz values.

The schemes presented in subsections 4.1 and 4.3 only save the Ritz pairs near the wanted eigenvalues. Our first set of tests extend these schemes to save the largest Ritz values as well. The choices considered are: saving only the converged ones, saving a fixed number of them, and saving nearly converged ones.

To limit the numbers of tests, we start by varying the scheme described in section 4.1. The first test performed is to save the largest eigenvalues that are converged. The value of k_l is defined by Equation 6. The time and the iterations used to solve si4 and si6 test problems are

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	548	304	272	4.87	3.52	5.86
si6	1561	279	276	35.0	7.82	11.9

From this simple test we see that saving only converged unwanted eigenvalues does not significantly alter the overall performance of the restarted Lanczos method. There is only one case where the modified scheme reduces the number of matrix-vector multiplications, however, more time is used in the same case.

Part of the reason that saving only converged unwanted Ritz values does not work well is that the unwanted ones are not computed to high accuracy in the restarted Lanczos method. This is especially true when the basis size m is relatively small. One scheme to overcome this problem is to always save a fixed number of unwanted Ritz pairs. The next table is generated by always saving one unwanted Ritz pair.

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	548	304	272	4.87	3.52	5.86
si6	1561	279	276	35.0	7.82	11.9

The time and the iterations in this table are fairly close to those of the unmodified scheme shown in subsection 4.1. Because of this, we decided to save two unwanted Ritz pairs instead. The iterations and time used with this modification are

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	539	285	267	5.24	3.26	5.84
si6	2468	413	432	79.4	11.9	19.7

Saving two unwanted Ritz pairs in addition to a number of wanted Ritz pairs reduces the number of matrix-vector multiplications when solving si4 test problem. However, more time were used in most cases. A more flexible scheme is needed to decide how many unwanted Ritz pairs to save. The rationale behind the scheme described in subsection 4.1 is to save a number of unconverged ones in addition to the converged ones. Next we consider a similar scheme for the unwanted Ritz pairs. Let n_u denote the number of unwanted Ritz values that have converged. We save $n_u + 1$ unwanted Ritz values in the next set of tests. The number of iterations and time are listed in the following table.

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	527	292	268	5.16	3.33	5.98
si6	3407	423	271	111.2	12.1	11.8

This modification reduces the matrix-vector multiplications used when the basis size is 50 and 100, but it does not always reduce the time even when the numbers of iterations are reduced. The following are results of applying the same modification to the scheme of saving nearly converged wanted Ritz values, see subsection 4.3.

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	527	292	268	5.08	3.33	5.96
si6	3407	423	271	113.1	12.1	11.8

For the two heuristics that do not initially save unwanted Ritz pairs, saving unwanted ones are helpful in reducing number of matrix-vector multiplications in some cases. Since more Ritz pairs are saved, an iteration on average uses more arithmetic operations than before, therefore the modified schemes often uses more time overall.

The schemes described in sections 4.4 and 4.5 save a number of unwanted Ritz values by design. Is there a benefit to not saving those unwanted ones? By discarding the unwanted Ritz pairs, the number of vectors saved will be smaller than before. This may reduce the cost of restarting and reduce the overall execution time. The following table records the iterations and the time used to solve the test problems with a scheme that first maximizes γ ($k = m - 3$) and then reset k_r to m,

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	390	298	269	4.01	5.15	23.0
si6	2134	444	399	66.9	17.6	33.1

This modification to the scheme of maximizing gap ratio reduces the execution time of the restarted Lanczos method, but increases the number of iterations in most test cases. Similar modification is also applied to the scheme of maximizing μ, see subsection 4.5. The time and the iterations used to solve the two test smaller problems are

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	618	305	277	6.04	3.72	7.32
si6	1819	466	389	55.5	15.0	19.3

We see that more iterations are used with this modification compared to the original scheme shown in subsection 4.5 when the basis size is relatively small ($m = 20, 50$). Less time is used when the basis size is larger, ($m = 50, 100$). However, even with this modification, the scheme of maximizing μ is not able to achieve the optimal performance shown in table 2.

Overall, saving unwanted Ritz pairs using the simple schemes described in this section is beneficial only in a small number of cases. Based on this set of tests, there is no reason to change the four strategies to include or to exclude unwanted eigenvalues.

5.2 Reducing time while maximizing gap ratio

In subsection 4.4 we pointed out the need of dynamically choosing the number vectors to save when maximizing the effective gap ratio. This section will explore combining the observations made in subsections 4.1 and 4.4 to automatically achieve the optimal timing results shown in Table 4.

The first test uses Equation 6 to determine the number of Ritz pairs to be saved, then maximize the gap ratio γ under the constraint that k Ritz pairs will be saved. The iterations and time used by the thick-restart Lanczos method with this restarting scheme are shown next.

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	908	305	278	9.09	3.46	5.88
si6	3159	283	278	69.7	8.02	12.1

Clearly, when the basis size is relatively small, say $m = 20$, much more time is used with this scheme. In fact, when the basis size is small ($m = 20$), none of the variants of maximizing gap ratio uses less time than the simple index based schemes, see subsection 4.1. When the basis size is larger, $m = 50$ and $m = 100$, this combined scheme uses about the same amount of time as shown in subsection 4.1 but more time than the optimal case shown in Table 4.

The second test uses Equation 7 to determine the number of Ritz pairs to be saved, then maximize the gap ratio γ under the constrain that k Ritz pairs will be saved. The resulting number of iterations and time are listed in the following table.

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	493	293	292	6.15	3.18	4.68
si6	1335	280	403	39.0	7.78	15.7

This set of results are again close to simply saving k smallest Ritz values, see subsection 4.1.

Maximizing μ with a fixed k is same as maximizing γ. For this reason, there is no need to apply the same modification to the scheme of maximizing μ. However, in the implementation used to produce the results shown in subsection 4.5, we limited $k_l \le k_r - 3$. When k_l and k_r are close to each other, the value of μ is significantly larger than its actual value. To avoid this situation, we mandate a larger separation between k_l and k_r, for example, $k_l \le k_r - (m - n_c)/2$. The test results of using this modified version of maximizing μ are

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	468	287	294	4.58	3.42	6.73
si6	2024	511	403	65.0	15.2	18.7

We see that this choice works reasonably well for si4 test problem but not so well for si6 test problem. Using the constraint $k_l \le k_r - (m - n_c)/2$, the number of Ritz pairs saved is guaranteed to be less than $(m + n_c)/2$. When the basis size is small, this leads to too few Ritz pairs being saved.

The next modification relax the search range to $k_l \le k_r - \min(m - n_{eig}, 2(m - n_c)/5)$. This change increases the limit on k and allows more Ritz pairs to be saved. This added flexibility helps to reduce the time and iterations used to solve the si6 test problem as shown in the following table.

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	471	297	274	4.77	3.59	5.83
si6	1209	456	276	32.2	13.9	13.1

By ensuring a large separation between k_l and k_r, we are able to achieve much better performance than allowing them to be arbitrarily close. However, we have not achieved the performance target shown in Table 4.

5.3 Ensuring convergence

The schemes described in subsections 4.1, 4.2 and 4.3 do not use the Ritz values when deciding the thickness. One simple way of taking the Ritz values into account is to ensure that a reasonably large gap ratio is achieved. The goal of this modification is to ensure that the wanted eigenvalues can be computed within specified number of iterations. If the residual vector of the current target is r_t, the effective gap ratio is γ, and the tolerance on residual norm is c, then the number of iterations required to make the target converge may be estimated as $ln(\|r_t\|/c)/\gamma$. If n_c is the number of Ritz pairs already converged and n_{matvec} is the number of matrix-vector multiplications (iterations) used so far, the average number iterations to compute each pair is $(n_{matvec} + ln(\|r_t\|/c)/\gamma)/(n_c + 1)$. Assuming the rest of eigenvalues are equally difficult to compute, the total number of iterations needed to compute all wanted eigenvalues is

$$\frac{n_{eig}}{(n_c+1)}\left(n_{matvec} + \frac{ln(\|r_t\|/c)}{\gamma}\right).$$

The maximum number of iterations to be used is usually specified by the user. From the above expression we can derive the desired γ to ensure solutions are found within the specified number of iterations. It is possible that the quantity $n_{matvec}n_{eig}/(n_c + 1)$ is larger than the maximum iterations allowed. In this case, the above formula will compute an invalid γ ($\gamma < 0$). If this happens, we compute a minimal γ that will ensure the current target will converge in the remaining iterations. In addition, we always make sure that at least three Ritz pairs are discarded during restart.

The first test to incorporate this heuristic is implemented as a modification to the simple index based scheme. The number of Ritz pairs to save is first computed using Equation 6. Additional Ritz pairs are saved to make sure the desired minimal γ is achieved. The following table lists the number of iterations and time used to solve the two silicon cluster test problems.

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	488	304	275	4.29	3.70	5.91
si6	1369	279	276	32.8	7.86	11.9

When the basis size is small ($m = 20$), the performance of the restarted Lanczos method with this modification is better than without it, see subsection 4.1. When basis size is larger,

$m = 50, 100$, the performance differences is fairly small. The reason is that with large basis size, Equation 6 already leads to large enough gap ratio and the modification does not change the thickness used.

When the basis size m is 50 or 100, Equation 7 prescribes a smaller thickness and this may lead to less time being used. The next test uses the smaller value of Equation 6 and 7,

$$\begin{aligned} &k_l = k_x, \qquad k_r = m, \\ &k_x \equiv n_c + \min\left((m - n_c)(\frac{2}{5} + \frac{n_{eig}}{10m}), n_{eig}\right), \end{aligned} \tag{10}$$

then modifies the thickness to ensure the minimal gap ratio is achieved. The test yields the following results.

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	488	297	298	4.29	3.01	4.62
si6	1369	277	416	32.8	7.40	16.0

When the basis size is 50 or 100, the time used by this combined scheme is very close to those used with Equation 7 alone. This is again because the modification to ensure the minimal gap ratio did not change the thickness.

The same modification can be applied to the scheme of saving nearly converged Ritz pairs as well, see subsection 4.3. The following table lists the test results.

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	488	304	275	4.26	3.51	5.96
si6	1369	279	276	32.9	7.81	12.0

From all the above tests, we see that adding this modification is useful when the basis size is small. When the basis size is large, our dynamic thick-restart scheme already achieve the desired effective gap ratio, therefore the additional modification does not change the actual number of Ritz pairs saved.

5.4 Using biased estimate

The computed effective gap ratio can be much larger than its actual value when the Ritz values are different from the corresponding eigenvalues. Here we will use an alternative formula for compute the gap ratio to see whether or not we can generate more effective schemes.

If the residual vector of a Ritz pair is r, then the actual eigenvalue is in the range of $(\lambda - \|r\|, \lambda + \|r\|)$ which is often called the trust region [7, 10]. In fact, if we want to compute the smallest eigenvalue and λ_1 is the smallest Ritz value, the actual eigenvalue is most likely in the range of $[\lambda - \|r\|, \lambda]$. In this case, we can use $\lambda - \|r\|$ as the biased estimate of the eigenvalue [14]. This biased estimate has been successfully used as the shift in for the

Davidson method [14]. Here we will use it as an alternative way of computing the effective gap ratio.

The biased estimates can be closer to the actual eigenvalue than the Ritz values in some cases. We can use the biased estimates in place of the Ritz values in the formula of effective gap ratio

$$\hat{\gamma} = \frac{\lambda_{k_l} - \|r_{k_l}\| - \lambda_1 + \|r_1\|}{\lambda_{k_r} + \|r_{k_r}\| - \lambda_1 + \|r_1\|}.$$

Because the residual norms have more complex relation with their indices, $\hat{\gamma}$ is no longer a monotonic function of k_l and k_r. However, both $\|r_{k_l}\|$ and $\|r_{k_r}\|$ may be quite small if k_l and k_r are close to m. If we use a brute-force method to search for a pair of k_l and k_r that gives the maximum $\hat{\gamma}$ we may reach the conclusion that the pair $k_l = m-1$ and $k_r = m$ is the best, which it is obviously not. Our first implementation of this scheme uses the same restriction as in the dynamic thick-restart Davidson method, that is, $k_l \leq k_r - 3$. The following table shows the results of using this scheme.

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	699	324	280	9.94	11.3	60.9
si6	2182	606	418	71.7	42.3	84.8

This test shows that using $\hat{\gamma}$ produces roughly the same overall performance in the restarted Lanczos method.

This test demonstrates that $\hat{\gamma}$ has similar shortcomings as γ. Since allowing k_l and k_r to vary arbitrarily does not give the desired results, we can let them vary near some values that are known to be good. In previous section, we have seen that saving k_x smallest Ritz values works reasonably well. Now we will try to find the maximum $\hat{\gamma}$ near k_x to see whether we can further enhance it effectiveness. Since this scheme only searches for a local maxim $\hat{\gamma}$, we will avoid the pitfalls mentioned before. To construct a computer program out this idea, we need to decide exactly what range to search for k_l. Let's first consider search in the range of $(k_x, (k_x + m)/2)$. The test results are

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	575	283	274	5.84	3.72	7.26
si6	888	285	416	23.2	9.60	21.5

In five out of the six test cases, more time is used compared to simply setting k_l to k_x. Next we restrict the above search range with an additional condition of $\lambda_{k_x} \leq \lambda_{k_l} \leq \lambda_{k_x} + \|r_{k_x}\|$. The following table lists the results of using this scheme to determine what Ritz pairs to save.

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	517	293	298	4.45	3.03	4.59
si6	1181	277	432	27.8	7.74	17.2

The time used to solve the test problems is significantly reduced compared to the previous implementation. The only case that more time is used than the optimal fixed thickness scheme is solving the si6 test problem with $m = 100$. Overall, this scheme leads to good performance for the restarted Lanczos method.

Near the end of the previous subsection, we have tested a scheme of saving k_x smallest Ritz values and also ensuring the minimal gap ratio is achieved. Instead of checking that γ is larger than the minimal value, we can alternatively making sure $\hat{\gamma}$ is larger than the minimal value. The following table lists the test results of using this alternative scheme.

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	474	297	298	4.27	3.04	4.55
si6	1463	277	416	34.8	7.38	16.1

This set of results are roughly the same as using γ instead of $\hat{\gamma}$. In addition, with either one of the modifications, the thick-restart Lanczos method uses about the same amount of time and iterations as simply saving k_x smallest Ritz values. The main reason for this is that saving k_x Ritz pairs already ensure the minimal gap ratio for the two test problems.

Similar to replacing γ with $\hat{\gamma}$ when maximizing the gap ratio, we can also replace γ with $\hat{\gamma}$ when maximizing μ, i.e., maximizing $(m-k)\hat{\gamma}$. The following table contains the results of maximizing $(m-k)\hat{\gamma}$ under the constraint of $k_l < k_r - \min(m - n_{eig}, 2(m - n_c)/5)$.

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	465	303	281	4.68	3.61	7.21
si6	1786	501	279	46.6	15.2	13.1

The corresponding results of maximizing $(m-k)\gamma$ is shown in the last table of subsection 5.2. Comparing the two tables, we see that slightly more iterations and time are used when maximizing $(m-k)\hat{\gamma}$.

With the limited number of tests performed in here, we do not see a clear advantage of using the biased estimate when computing the effective gap ratio. We will simply use γ whenever the effective gap ratio is needed in the remaining of this paper.

5.5 Saving degenerate Ritz pairs

One of the heuristics not yet considered is related to degeneracies. If two Ritz values are nearly identical and one of them is to be saved, we should save them both. In a Lanczos method implemented in the floating-point arithmetic, the Ritz values corresponding to the degenerate eigenvalues are never exactly identical to each other. The main difficulty to implement this heuristic is how to determine degeneracies. The criteria we use for judging whether or not two Ritz values will eventually converge to two identical eigenvalues is whether their trust region overlap. In this subsection we will explore a few different ways of implementing this strategy.

To limit the number of tests to perform, we start by using this heuristic as a modification to selecting k based on Equation 10. In other word, when computing the smallest eigenvalues, we first set $k_r = m$ and $k_l = k_x$, then modify k_l to include Ritz values that are close to λ_{k_x}. To determine whether λ_i and λ_{i+1} could be considered as degenerate, our first test is $\lambda_i > \lambda_{i+1} - \|r_{i+1}\|$. Using this criteria, the results of solving the two silicon cluster test problems are

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	552	292	295	5.23	3.05	4.58
si6	1218	408	412	29.3	11.2	16.0

The modification reduces the iterations for larger m when solve si4 test problem but the same is not true for si6 test problem.

The second criteria for testing whether two Ritz values could eventually converge to the same eigenvalue is $\lambda_i + \|r_i\| > \lambda_{i+1}$ and the time and the iterations used to solve the two silicon cluster test problems are

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	600	294	297	5.47	3.07	4.64
si6	1839	275	412	43.0	7.37	16.4

When the basis size is 20, both above modifications increase the time and iterations used compared to the original scheme shown in subsection 5.3.

The previous two criteria for testing degeneracies are combined to form the third test. In this case, both $\lambda_i > \lambda_{i+1} - \|r_{i+1}\|$ and $\lambda_i + \|r_i\| > \lambda_{i+1}$ have to be satisfied in order for λ_i and λ_{i+1} to be considered *degenerate* and the results of using this test criteria are

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	533	294	297	4.65	3.02	4.60
si6	1458	276	412	33.3	7.39	16.1

Since both λ_i and λ_{i+1} are expected to decrease in the future iterations and $\|r_{i+1}\|$ is usually larger than $\|r_i\|$, we modified the above test to be $\lambda_i - \|r_i\| > \lambda_{i+1} - \|r_{i+1}\|$ and $\lambda_i + \|r_i\| > \lambda_{i+1}$. With this test, the trust region of λ_i is completely covered by that of λ_{i+1}. The resulting time and iterations used by the thick-restart Lanczos method are

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	534	296	297	4.78	3.06	4.60
si6	1323	277	412	29.5	7.39	16.0

Comparing the results of using these four set of testing criteria for determining potential degeneracies, we see that the time and iterations used with small basis size ($m = 20$) have been steadily decreasing. However, the time and iterations used with larger basis sizes are almost the same. For future discussion, we will use the last set of testing criteria.

Since the idea of guaranteeing minimal gap ratio was found to be useful when the basis size is small, we try to combine this heuristic and the above tests for degeneracy to see whether or not the resulting scheme is even better for small basis size. The resulting time and iterations used by the restarted Lanczos method are

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	504	296	297	4.49	3.06	4.60
si6	1395	277	415	33.8	7.46	16.3

The set of results are roughly the same as modifying the strategy of saving k_x smallest Ritz pairs to ensure the minimal gap ratio is achieved, see subsection 5.3. The additional modification of saving those Ritz values that are potentially degenerate does not significantly change the overall effectiveness of the eigenvalue method. Since both modifications are very simple to implement, we can use both of them without significantly cost and we will do so in the future tests.

The same two heuristics can also be easily applied to the strategy of saving nearly converged Ritz pairs, see subsection 4.3. The time and iterations used with this modified version of the strategy are as follows,

	MATVEC			time (sec)		
	$m = 20$	$m = 50$	$m = 100$	$m = 20$	$m = 50$	$m = 100$
si4	504	303	274	4.38	3.40	5.87
si6	1395	279	275	34.0	7.82	11.9

Compared with the results without these two modifications, see subsection 4.3, the modified scheme is considerably more effective when the basis size is small. Compared with the scheme which only has the modification to guarantee the minimal gap ration, see the last experiment of subsection 5.3, we see that the newly added modification of saving degenerate eigenvalues does not significantly change the overall performance.

6 Putting it together

In proceeding sections, we have identified four basic strategies of determining what and how many Ritz pairs to save for the thick-restart scheme, (1) saving a number of Ritz pairs based on indices, (2) saving a number of Ritz pairs based on residual norms, (3) saving Ritz pairs to maximize the *effective gap ratio* γ and (4) saving Ritz pairs to maximize μ. We have tested a number of formulae for the first two schemes and achieved a reasonably good performance on the test problems. In their simplest forms, the last two schemes save too many Ritz pairs in most test cases and are only effective in reducing the number of iterations. Because they save too many Ritz pairs, each iteration is more expensive on average. This leads to more time being used by the Lanczos method with these two restarting strategies.

We have experimented with modifying the four basic strategies by (1) saving the unwanted Ritz values in addition to the wanted one, (2) forcing the last two schemes to save less Ritz pairs, (3) using an alternative formula for the effective gap ratio, (4) maintaining a reasonable gap ratio, and (5) saving potentially degenerate Ritz pairs. When a relatively small portion of the Lanczos basis is saved during restarting, including additional unwanted Ritz pairs can reduce the number of iterations. In our tests, this happens when the basis size is 50 or 100. When the number of Ritz pairs saved is close to the basis size m, saving additional unwanted Ritz pairs generally causes more time to be used. This is the case when the basis size is close to the number of eigenvalues wanted. If we first choose a moderate size k then try to maximizing γ, the difference between k_l and k_r is fixed and the resulting choices often lead to better performance than allowing k_l and k_r to become arbitrarily close. Computing the gap ratio using the biased estimates of the eigenvalues do not significantly alter the overall effectiveness of the eigenvalue method compared to computing gap ratio using Ritz values. The last two modifications of maintaining a reasonable gap ratio and saving degenerate Ritz pairs are relatively inexpensive to implement and fairly useful when the basis size is close the the number of eigenvalues wanted.

As an example of how to use all the different heuristics, we will describe our final implementation of the four restarting strategies.

1. Index based scheme When trying to find a number of the smallest eigenvalues, at each restart, this scheme selects

$$k_l = n_c + \min\left(n_{eig}, (m - n_c)(\frac{2}{5} + \frac{n_{eig}}{10m})\right)$$

smallest Ritz pairs. This basic choice is modified to ensure a minimal gap ratio is achieved, see subsection 5.3 and potentially degenerate Ritz pairs are saved, see subsection 5.5. No unwanted Ritz pairs are saved in this case ($k_r = m$).

2. Residual norm based scheme This scheme saves Ritz pairs near the wanted eigenvalues and have residual norms smaller than $r_s = \max(2r_{t+1}, \sqrt{r_{max}r_t})$, see subsection 4.3. Two modifications are applied after the basic steps are taken, they ensure a reasonable gap ratio is achieved and save potentially degenerate Ritz pairs. No unwanted Ritz pairs are saved.

3. Maximizing effective gap ratio This scheme first determines the number Ritz pairs to be saved, $k = \max(n_{eig}, (3m + 2n_c)/5)$, then search for a combination of k_l and k_r that gives the largest γ. The above formula gives the following relation for k_l and k_r is $k_l = k_r + \min(m - n_{eig}, 2(m - n_c)/5)$. No further modification is applied.

4. Maximizing progress This scheme search through all possible choices of k_l and k_r to maximize the value of $\mu = (m - k) * \gamma$. The constraint on k_l and k_r is $k_l \le k_r - \min(m - n_{eig}, 2(m - n_c)/5)$.

Table 5: Time and iterations used to compute the five smallest eigenvalues of InGaAs9k test problem.

	MATVEC	time (sec)
index scheme	1469	144.6
residual scheme	2813	274.2
max γ	3461	336.4
max μ	984	104.2

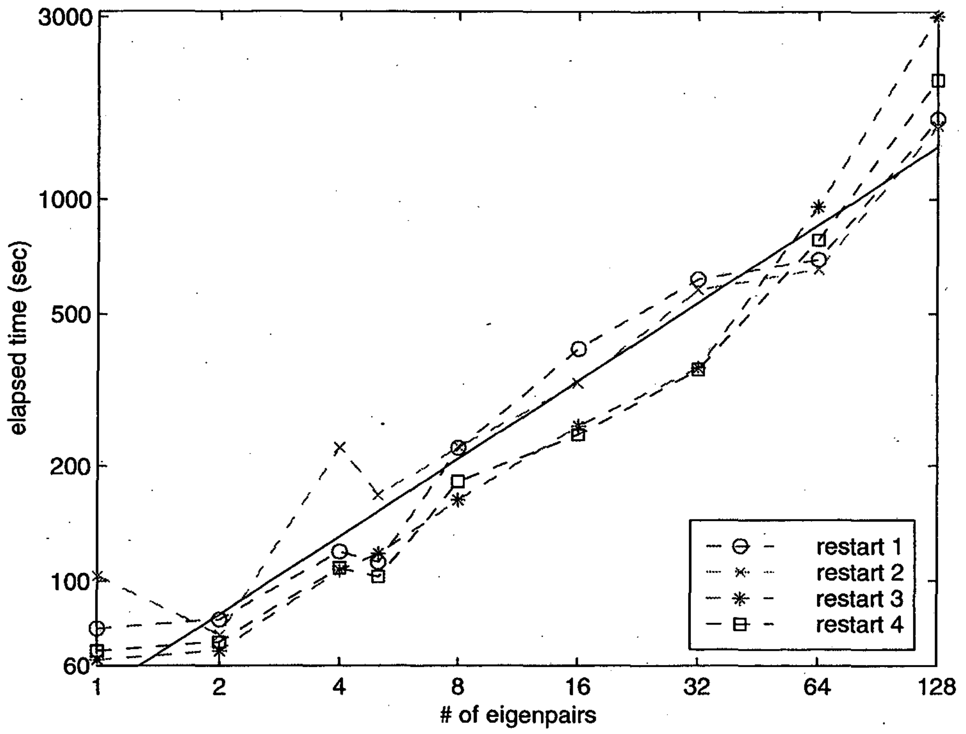


Figure 4: The time (seconds) used to find different number of eigenvalues of the InGaAs9k test problem.

Table 5 lists how these four schemes perform when computing the five smallest eigenvalues of the InGaAs9k test problem. The elapsed time is measured on 32 processors of a Cray T3E-900. Compared to the timing results shown in Figure 1, we see that the Lanczos method with restarting scheme one and four uses significantly less time than the minimal time used with the fixed thickness scheme.

In many electronic structure calculations, a large number of eigenvalues and eigenvectors are computed. Figure 4 demonstrates how the thick-restart Lanczos method scales as the number of eigenvalues increases. In this test, the basis size used is always $n_{eig} + 20$. In other word, the Lanczos only need workspace to store 21 vectors, 20 for the Lanczos vectors, one for the residual vector. The n_{eig} vectors needed to store the eigenvectors are used by the program to store Ritz vectors as well. From the plot we see that the time required to compute the smallest eigenvalue is almost the same as computing two smallest ones. When the number of eigenvalues to be computed is between 2 and 32, the last two restarting schemes are slightly better than the first two. Using one of the last two restarting schemes, doubling the number of eigenvalues, the restarted Lanczos method takes about 45 percent more time. When computing more than 32 eigenvalues, in other words, when $n_{eig} > m/2$, the first two restarting schemes become more competitive than the last two. In the range tested, the Lanczos method with the first scheme uses 65 percent more time when the number of eigenvalues doubles, and the Lanczos method with the second restarting scheme uses about 55 percent more time. No matter which restarting strategy is used, the thick-restart Lanczos method scales sublinearly with the number of eigenvalues. This suggests that the thick-restart Lanczos method may be used to compute a large number of eigenvalues.

In general, the behavior of the thick-restart Lanczos method is determined by the details of the spectrum distribution. The fact that the optimal thickness is significantly different for si4 and si6 test problem when $m = 100$ demonstrates this point clearly. Through the tests, we have demonstrated the importance of developing good restarting strategy and have showed how to implement the four different restarting schemes. Our tests shown that the thick-restart Lanczos using these restarting strategies is capable of efficiently computing a large number of eigenvalues and eigenvectors of a large matrix.

References

[1] J. Baglama, D. Calvetti, and L. Reichel. Iterative methods for the computation of a few eigenvalues of a large symmetric matrix. *BIT*, 36:400–421, 1996.

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

[3] J. R. Chelikowsky, N. Troullier, and Y. Saad. Finite-difference-pseudopotential method: electronic structure calculations without a basis. *Phys. Rev. Lett.*, 72:1240–3, 1994.

[4] J. R. Chelikowsky, N. Troullier, K. Wu, and Y. Saad. Higher order finite difference pseudopotential method: an application to diatomic molecules. *Phys. Rev. B*, 50:11355–11364, 1994.

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

[6] Ernest R. Davidson. Super-matrix methods. *Computer Physics Communications*, 53:49–60, 1989.

[7] G. H. Golub and C. F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, MD 21211, third edition, 1996.

[8] R. Lehoucq, D. Sorensen, and C. Yang. *ARPACK USERS GUIDE: solution of large scale eigenvalue problems with implicitly restarted Arnoldi methods.* SIAM, Philadelphia, PA, 1998. ARPACK Software is available at `http://www.caam.rice.edu/software/ARPACK/`.

[9] Ronald B. Morgan. On restarting the Arnoldi method for large nonsymmetric eigenvalue problems. *Mathematics of Computation*, 65(215):1213–1230, July 1996.

[10] Beresford N. Parlett. *The symmetric eigenvalue problem.* Classics in Applied Mathematics. SIAM, Philadelphia, PA, 1998.

[11] Yousef Saad. SPARSKIT: A basic toolkit for sparse matrix computations. Technical Report 90-20, Research Institute for Advanced Computer Science, NASA Ames Research Center, Moffet Field, CA, 1990. Software currently available at `ftp://ftp.cs.umn.edu/dept/sparse/`.

[12] Yousef Saad. *Numerical Methods for Large Eigenvalue Problems.* Manchester University Press, 1993.

[13] D. S. Sorensen. Implicit application of polynomial filters in a K-step Arnoldi method. *SIAM J. Matrix Anal. Appl.*, 13(1):357–385, 1992.

[14] A. Stathopoulos, Y. Saad, and C. F. Fischer. Robust preconditioning of large, sparse, symmetric eigenvalue problems. *Journal of Computational and Applied Mathematics*, 64:197–215, 1995.

[15] A. Stathopoulos, Y. Saad, and K. Wu. Dynamic thick restarting of the Davidson and the implicitly restarted Arnoldi methods. *SIAM J. Sci. Comput.*, 19(1):227–245, 1998.

[16] Kesheng Wu and Horst Simon. Thick-restart Lanczos method for symmetric eigenvalue problems. Technical Report 41412, Lawrence Berkeley National Laboratory, 1998.

[17] Alex Zunger. Electronic-structure theory of semiconductor quantum dots. *MRS Bulletin*, 23(2):35–42, February 1998.

Prepared for the U.S. Department of Energy under Contract No. DE-AC03-76SF00098