Using small eigenproblems to accelerate power method iterations

Sara Pollock
Department of Mathematics, University of Florida

L. Ridgway Scott
University of Chicago

July 5, 2021

Abstract

We consider algorithms that operate on sequences of Krylov vectors generated by the (inverse) power method that accelerate computation of the dominant eigenvalue. The algorithms use subsequent Krylov vectors to form a small eigenproblem which is solved exactly. The dominant eigenvector is used to restart the Krylov process. We explain this in detail using just two subsequent vectors, and then we generalize the algorithm to use multiple Krylov vectors. We show how this is related to restarted Arnoldi and other methods.

Eigenproblems for large systems occur in stability analysis of numerical schemes [4], in stability analysis for partial differential equations [14], and much more. The power method is often preferred due to

- its ease of implementation,
- the fact that it is matrix-free, and
- the limited amount of storage required.

In many cases [4, 14], one is interested in the smallest eigenvalues. The inverse power method requires repeated solution of the associated system of equations. Codes for a solver can be modified to become a codes for the inverse power method.

The convergence of the power method [13] is governed by the ratio of the largest eigenvalues. Often this ratio is very close to 1, and correspondingly the rate of convergence is slow. We examine here variants of the power method that achieve faster convergence with a controlled increase in storage.

1 Getting more out of the power method

In the power method iteration, one computes successive matrix-vector products

$$y^k = A\hat{y}^{k-1}, \qquad \hat{y}^k = c_k y^k,$$

where c_k is a scaling constant, e.g., $c_k = \|y^k\|^{-1}$ for some norm. By induction,

$$y^k = \left(\prod_{i=0}^{k-1} c_k\right) A^k y^0.$$

Thus to analyze the power method, it suffices to consider the unscaled iteration

$$x^k = Ax^{k-1}.$$

The scaling can be applied as needed to avoid under/over-flow. The sequence of vectors are called Krylov vectors.

Generically, the power method tends to the eigenvector associated with the largest (in magnitude) eigenvalue of A. The eigenvalue can be approximated by the Rayleigh quotient

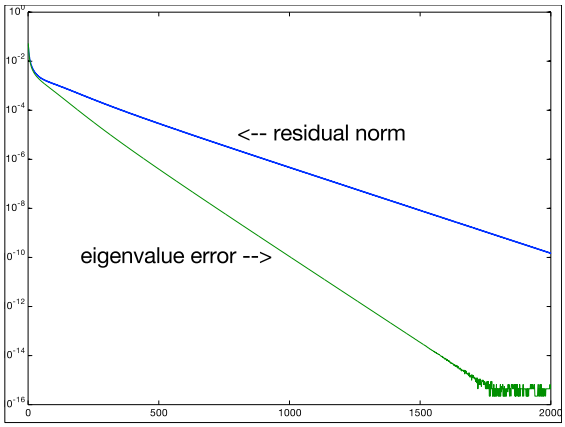
$$\lambda_k = \frac{(x^k)^t A x^k}{(x^k)^t x^k}.$$

Note that the scaling of the vector iterates does not impact the value of λ_k, i.e., you could work with either the scaled vectors y^k or unscaled x^k.

There are typically many vectors generated by a power iteration sequence, so one might ask if there is a way to extract information from sets of Krylov vectors.

2 Using small eigenproblems

Standard algorithms used to approximate eigenproblems project onto subspaces of Krylov vectors, especially for large, sparse systems [5, 15]. This is typically done to reduce the computational complexity. Here we explore this idea in the simplest case of just two Krylov vectors. For this reason, we refer to it as the two-step algorithm. This can be easily generalized to involve more than two steps. The two-step algorithm involves two Krylov vectors, but its output is just a single vector and a single eigenvalue estimate, even though internally the method produces two eigenvalue estimates and two eigenvectors. We will see by example that it is necessary to discard the extra information. This is in accord with what is recommended in the SLEPc package [5, 12]: to compute k eigenpairs, at least $2k$ Krylov vectors should be used. We will see that our approach is very similar to an implicitly restarted Arnoldi method [8].

(a) 

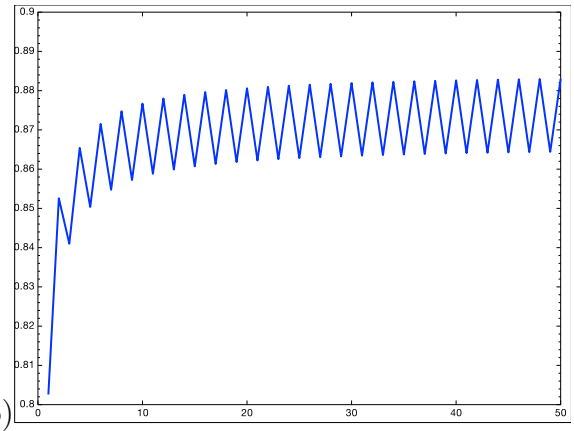
(b) 

Figure 1: (a) Residual norm and eigenvalue error for the two-step algorithm for the largest eigenvalue for Matrix 2 in [9] using the formulation (2.8). (b) Second eigenvalue in the two-step algorithm for the eigenproblem for Matrix 2 in [9]. In both cases, the horizontal axis is the number of iterations of the two-step algorithm which is half the number of matrix-vector multiplications.

2.1 The two-step algorithm

Starting with an initial vector $\hat{\mathbf{y}}$, we normalize

$$\mathbf{y} = \|\hat{\mathbf{y}}\|^{-1}\hat{\mathbf{y}}. \tag{2.1}$$

Next compute a Krylov vector $\tilde{\mathbf{w}} = A\mathbf{y}$ and then orthogonalize (and compute the residual)

$$\mathbf{r} = \tilde{\mathbf{w}} - (\tilde{\mathbf{w}}^t\mathbf{y})\mathbf{y}. \tag{2.2}$$

Finally normalize

$$\mathbf{w} = \|\mathbf{r}\|^{-1}\mathbf{r}. \tag{2.3}$$

(If by chance $\mathbf{r} = \mathbf{0}$, this is a good thing; see section 2.2.) Note that

$$\|\mathbf{r}\|\,\mathbf{y}^t\mathbf{w} = \mathbf{y}^t\mathbf{r} = \mathbf{y}^t\big(\tilde{\mathbf{w}} - (\tilde{\mathbf{w}}^t\mathbf{y})\mathbf{y}\big) = \mathbf{y}^t\tilde{\mathbf{w}} - \tilde{\mathbf{w}}^t\mathbf{y} = 0.$$

Thus $\mathbf{y}$ and $\mathbf{w}$ are orthonormal vectors.

Now we project the eigenproblem for A onto the space spanned by $\mathbf{y}$ and $\mathbf{w}$. Thus we seek a and b such that

$$A(a\mathbf{y} + b\mathbf{w}) = aA\mathbf{y} + bA\mathbf{w} = \lambda(a\mathbf{y} + b\mathbf{w}). \tag{2.4}$$

Taking dot products with (2.4), we get

$$a\,\mathbf{y}^tA\mathbf{y} + b\,\mathbf{y}^tA\mathbf{w} = \lambda a, \qquad a\,\mathbf{w}^tA\mathbf{y} + b\,\mathbf{w}^tA\mathbf{w} = \lambda b.$$

Thus

$$\begin{pmatrix} \mathbf{y}^tA\mathbf{y} & \mathbf{y}^tA\mathbf{w} \\ \mathbf{w}^tA\mathbf{y} & \mathbf{w}^tA\mathbf{w} \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \lambda \begin{pmatrix} a \\ b \end{pmatrix}. \tag{2.5}$$

Note that if A is a symmetric matrix, then (2.5) is a symmetric eigenproblem, since then $\mathbf{y}^t A\mathbf{w} = \mathbf{w}^t A^t\mathbf{y} = \mathbf{w}^t A\mathbf{y}$.

We can turn this into an algorithm as follows. Given an initial $\mathbf{y}$, we perform the steps (2.1) through (2.5). Choosing the eigenvalue with maximum modulus, with corresponding eigenvector $(a, b)^t$, we set

$$\hat{\mathbf{y}} = a\mathbf{y} + b\mathbf{w}. \tag{2.6}$$

The output if the algorithm is then λ and $\hat{\mathbf{y}}$. We can repeat the process as long as desired.

2.2 Early termination

There is one small issue with all Krylov based methods. If we start with $\mathbf{y}$ as an exact eigenvector, that is, $A\mathbf{y} = \lambda\mathbf{y}$, then $\mathbf{y}$ and $A\mathbf{y}$ are linearly dependent. In this case, division by zero occurs in step (2.3) when we try to normalize $\mathbf{r}$ since in this case $\mathbf{r} = \mathbf{0}$. Thus in principle a check needs to be made that the initial vector is not an exact eigenvector. This occurs if and only if $\mathbf{r} = \mathbf{0}$. So we can add an early exit (with complete success) if this is detected.

More precisely, $\mathbf{r}$ is the residual:

$$\mathbf{r} = \tilde{\mathbf{w}} - (\tilde{\mathbf{w}}^t\mathbf{y})\mathbf{y} = A\hat{\mathbf{y}} - \frac{\tilde{\mathbf{w}}^t\mathbf{y}}{\|\hat{\mathbf{y}}\|}\hat{\mathbf{y}}, \tag{2.7}$$

for the eigenvalue $\lambda = \tilde{\mathbf{w}}^t\mathbf{y}\|\hat{\mathbf{y}}\|^{-1}$. If this residual is below a desired tolerance, we can exit with $\hat{\mathbf{y}}$ unchanged and λ given by

$$\lambda = \tilde{\mathbf{w}}^t\mathbf{y}\|\hat{\mathbf{y}}\|^{-1} = \tilde{\mathbf{w}}^t\hat{\mathbf{y}}\|\hat{\mathbf{y}}\|^{-2}.$$

2.3 Test of the two-step algorithm

We tested the two-step method for the eigenproblem for Matrix 2 in [9], as indicated in Figure 2(a). We found that it is not as good as the Augmented method in [9], but better than the Simple method in [9] (and thus much better than the power method), for Matrix 2. The two-step method hits a round-off floor once the eigenvalue error approaches machine precision, as indicated in Figure 2(a). Plotted in Figure 2(a) are the residual norm $\|A\mathbf{y} - \lambda\mathbf{y}\|$ and absolute eigenvalue error, which is $|1 - \lambda|$. Since this value apparently becomes zero for some iterations, we actually plot $|1 - \lambda| + \texttt{eps}$ where `eps` is the machine precision value (2.2204e-16) in `octave`. Figure 2(a) displays the expected [3] behavior that the eigenvalue error is proportional to the square of the eigenvector error.

Since there are two eigenvalues computed in the two-step algorithm, it might be of interest to know whether the second (smaller) eigenvalue has any significance. For example, does it approximate the second-largest (in absolute magnitude) eigenvalue? Unfortunately, Figure 1(b) indicates that the answer is no.

Since the algorithm involves just two Krylov vectors, there is some concern as to what happens if there are two eigenvalues ($\lambda_2 = -\lambda_1$) with maximum absolute value. We tested

the algorithm for the diagonal matrices

$$A = \begin{pmatrix} -1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & -1 \end{pmatrix},$$

with initial guess $\hat{\mathbf{y}} = (1, 1, 1)^t$. For A, the dominant eigenvalue converged quickly to -1, and for B, the dominant eigenvalue converged quickly to 1. In both cases, the second eigenvalue wandered aimlessly. Thus it appears that the algorithm does not fail in the case of two dominant modes with opposite signed eigenvalues, but its choice of sign is not predictable.

2.4 Simplifying the two-step algorithm

The number of matrix-vector multiplications in the computation of the matrix in (2.5) can be minimized in each iteration. For example,

$$\mathbf{y}^t A\mathbf{y} = \mathbf{y}^t \tilde{\mathbf{w}}.$$

Note that (2.2) implies that the residual satisfies

$$\|\mathbf{r}\|^2 = \mathbf{r}^t \tilde{\mathbf{w}} = \mathbf{r}^t A\mathbf{y} = \|\mathbf{r}\| \mathbf{w}^t A\mathbf{y},$$

so that

$$\|\mathbf{r}\| = \mathbf{w}^t A\mathbf{y}.$$

In the symmetric case,

$$\mathbf{y}^t A\mathbf{w} = \mathbf{w}^t A\mathbf{y} = \|\mathbf{r}\|.$$

Thus only $A\mathbf{w}$ needs to be calculated in addition to $A\mathbf{y}$ in each iteration. Thus, in the symmetric case, (2.5) becomes (recall that $\tilde{\mathbf{w}} = A\mathbf{y}$)

$$\begin{pmatrix} \mathbf{y}^t \tilde{\mathbf{w}} & \|\mathbf{r}\| \\ \|\mathbf{r}\| & \mathbf{w}^t A\mathbf{w} \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \lambda \begin{pmatrix} a \\ b \end{pmatrix}. \tag{2.8}$$

Results using the formulation (2.8) are depicted in Figure 1(a). By contrast, Figure 2(a) shows what happens if we use instead the formulation (2.5) which is substantially less accurate.

3 The 2×2 eigenproblem

Solving the 2×2 eigenproblem is easily done by formulæ. The characteristic polynomial is

$$\det \begin{pmatrix} \alpha - \lambda & \beta \\ \beta & \gamma - \lambda \end{pmatrix} = (\alpha - \lambda)(\gamma - \lambda) - \beta^2 = \lambda^2 - (\alpha + \gamma)\lambda + \alpha\gamma - \beta^2. \tag{3.1}$$

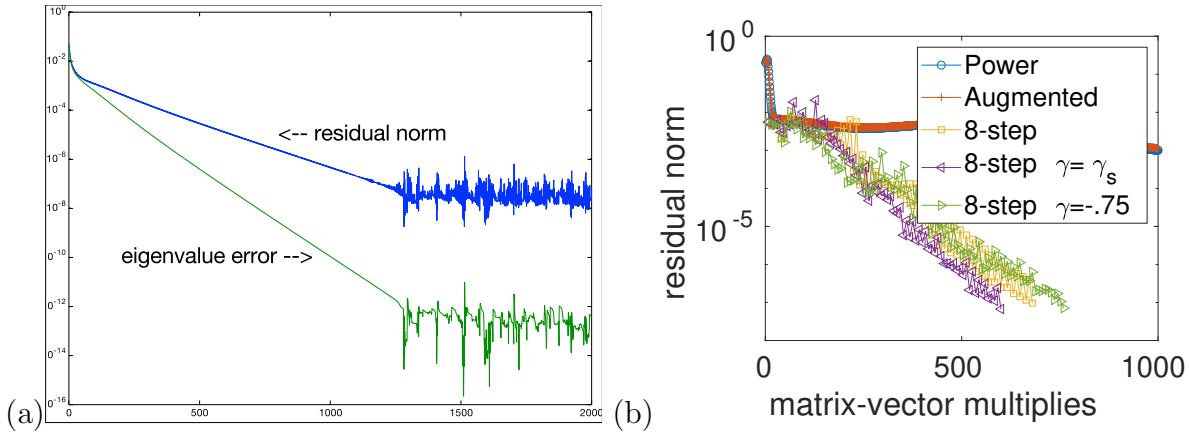


Figure 2: (a) Residual norm and eigenvalue error for the two-step algorithm for the eigenproblem for Matrix 2 in [9] using the formulation (2.5). The horizontal axis is the number of iterations of the two-step algorithm which is half the number of matrix-vector multiplications. (b) Comparison of the eigenvalue error for two-step method and two steps of the power method for the eigenproblem for Matrix 2 in [9]. The horizontal axis is half the number of matrix-vector multiplications.

By the quadratic formula,

$$\begin{aligned}\lambda &= \tfrac{1}{2}\big(\alpha+\gamma \pm \sqrt{(\alpha+\gamma)^2 - 4(\alpha\gamma-\beta^2)}\big)\\ &= \tfrac{1}{2}\big(\alpha+\gamma \pm \sqrt{(\alpha-\gamma)^2 + 4\beta^2}\big).\end{aligned} \tag{3.2}$$

When $\alpha = -\gamma$, then

$$\lambda = \tfrac{1}{2}\big(\pm \sqrt{(2\alpha)^2 + 4\beta^2}\big) = \pm\sqrt{\alpha^2+\beta^2}.$$

The extreme eigenvalue λ_e, the one with the largest absolute magnitude, is given by

$$\lambda_e = \tfrac{1}{2}\big(\alpha+\gamma+\operatorname{sign}(\alpha+\gamma)\sqrt{(\alpha-\gamma)^2+4\beta^2}\big), \tag{3.3}$$

where we need to define sign(0) appropriately, e.g., sign(0) = 1. The other eigenvalue λ_o is thus given by

$$\lambda_o = \tfrac{1}{2}\big(\alpha+\gamma-\operatorname{sign}(\alpha+\gamma)\sqrt{(\alpha-\gamma)^2+4\beta^2}\big), \tag{3.4}$$

The eigenvector $(a,b)^t$ corresponding to λ_e satisfies

$$(\alpha-\lambda_e)a + \beta b = \beta a + (\gamma-\lambda_e)b = 0.$$

Thus we get two equations for a and b:

$$\frac{a}{b} = -\frac{\beta}{\alpha-\lambda_e}, \quad \frac{b}{a} = -\frac{\beta}{\gamma-\lambda_e}.$$

These are the same equations, since (3.1) implies that

$$\frac{\beta}{\alpha-\lambda_e} = \frac{\gamma-\lambda_e}{\beta}.$$

If $|\beta| < |\alpha - \lambda_e|$, then $|a| < |b|$. So we can choose $b = 1$ and we will have $|a| < 1$. If $|\beta| \geq |\alpha - \lambda_e|$, then $|\beta| \leq |\gamma - \lambda_e|$. That is, we can choose

$$(a, b) = \begin{cases} \big(-\beta/(\alpha - \lambda_e), 1\big) & |\beta| < |\alpha - \lambda_e| \\ \big(1, -\beta/(\gamma - \lambda_e)\big) & |\beta| \geq |\alpha - \lambda_e| \end{cases} \tag{3.5}$$

and we will have $|(a, b)|_\infty \leq 1$. Formulæ (3.3) and (3.5) provide the solution of the 2×2 eigenproblem for the dominant mode. The values of a and b can then be used in (2.6).

3.1 An example

Suppose that $\hat{y} = v_1 + rv_2$. Then $\tilde{w} = A\hat{y} = \lambda_1 v_1 + r\lambda_2 v_2$ and $\hat{y}^t\hat{y} = 1 + r^2$. For simplicity, we assume that $\lambda_1 > \lambda_2 > 0$. Thus

$$y^t A y = \|\hat{y}\|^{-2}\hat{y}^t A\hat{y} = \frac{(v_1 + rv_2)^t(\lambda_1 v_1 + r\lambda_2 v_2)}{1 + r^2} = \frac{\lambda_1 + r^2\lambda_2}{1 + r^2}.$$

$$y = \frac{1}{\sqrt{1 + r^2}}\big(v_1 + rv_2\big), \qquad \tilde{w}^t y = (\lambda_1 v_1 + r\lambda_2 v_2)^t\hat{y}\|\hat{y}\|^{-1} = \frac{\lambda_1 + r^2\lambda_2}{\sqrt{1 + r^2}}$$

$$\begin{aligned} \hat{w} &= \tilde{w} - (\tilde{w}^t y)y = \lambda_1 v_1 + r\lambda_2 v_2 - \frac{\lambda_1 + r^2\lambda_2}{1 + r^2}(v_1 + rv_2) \\ &= \big(1 + r^2\big)^{-1}\big((1 + r^2)(\lambda_1 v_1 + r\lambda_2 v_2) - (\lambda_1 + r^2\lambda_2)(v_1 + rv_2)\big) \\ &= \big(1 + r^2\big)^{-1}\big(r^2(\lambda_1 - \lambda_2)v_1 + r(\lambda_2 - \lambda_1)v_2\big) \\ &= \frac{r(\lambda_1 - \lambda_2)}{1 + r^2}\big(rv_1 - v_2\big). \end{aligned} \tag{3.6}$$

$$\|\hat{w}\| = \frac{r(\lambda_1 - \lambda_2)}{\sqrt{1 + r^2}}, \qquad w = \frac{1}{\sqrt{1 + r^2}}\big(rv_1 - v_2\big).$$

$$w^t A y = \frac{1}{1 + r^2}\big(rv_1 - v_2\big)^t\big(\lambda_1 v_1 + \lambda_2 rv_2\big) = \frac{r(\lambda_1 - \lambda_2)}{1 + r^2}.$$

$$w^t A w = \frac{1}{1 + r^2}\big(rv_1 - v_2\big)^t\big(\lambda_1 rv_1 - \lambda_2 v_2\big) = \frac{r^2\lambda_1 + \lambda_2}{1 + r^2}.$$

Thus (2.5) becomes

$$\frac{1}{1 + r^2}\begin{pmatrix} \lambda_1 + r^2\lambda_2 & r(\lambda_1 - \lambda_2) \\ r(\lambda_1 - \lambda_2) & r^2\lambda_1 + \lambda_2 \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix} = \lambda\begin{pmatrix} a \\ b \end{pmatrix}. \tag{3.7}$$

Thus the terms in (3.3) can be evaluated as

$$\alpha + \gamma = \lambda_1 + \lambda_2, \qquad \alpha - \gamma = \frac{(1 - r^2)(\lambda_1 - \lambda_2)}{1 + r^2}, \qquad \beta = \frac{r(\lambda_1 - \lambda_2)}{1 + r^2}.$$

Therefore the quantity under the square root in (3.3) is

$$(\alpha-\gamma)^2+4\beta^2=(\lambda_1-\lambda_2)^2\frac{(1-r^2)^2+4r^2}{(1+r^2)^2}=(\lambda_1-\lambda_2)^2.$$

From (3.3), we have

$$\lambda_e=\tfrac{1}{2}\big(\lambda_1+\lambda_2+\operatorname{sign}(\lambda_1+\lambda_2)|\lambda_1-\lambda_2|\big)=\lambda_1. \tag{3.8}$$

By contrast, (3.4) yields

$$\lambda_o=\tfrac{1}{2}\big(\lambda_1+\lambda_2-\operatorname{sign}(\lambda_1+\lambda_2)|\lambda_1-\lambda_2|\big)=\lambda_2. \tag{3.9}$$

The quantities in (3.5) are

$$\alpha-\lambda_e=\frac{\lambda_1+r^2\lambda_2}{1+r^2}-\lambda_1=\frac{r^2(\lambda_2-\lambda_1)}{1+r^2}=-r\beta.$$

Equation (3.5) becomes

$$(a,b)=\begin{cases}\big((1/r),1\big) & 1<r\\ (1,r) & 1\geq r.\end{cases} \tag{3.10}$$

The output $\hat{y}$ of the algorithm is thus

$$\begin{aligned}\hat{y}=ay+bw&=\begin{cases}(1/r)y+w & 1<r\\ y+rw & 1\geq r.\end{cases}\\ &=\frac{1}{\sqrt{1+r^2}}\begin{cases}(1/r)v_1+v_2+rv_1-v_2 & 1<r\\ v_1+rv_2+r^2v_1-rv_2 & 1\geq r\end{cases}\\ &=\frac{1}{\sqrt{1+r^2}}\begin{cases}(1/r)v_1+v_2+rv_1-v_2 & 1<r\\ v_1+rv_2+r^2v_1-rv_2 & 1\geq r\end{cases}\\ &=\frac{1}{\sqrt{1+r^2}}\begin{cases}(1/r)v_1+rv_1 & 1<r\\ v_1+r^2v_1 & 1\geq r\end{cases}\\ &=c_rv_1,\end{aligned} \tag{3.11}$$

where

$$c_r=\begin{cases}\sqrt{1+r^{-2}} & 1<r\\ \sqrt{1+r^2} & 1\geq r.\end{cases}$$

Thus the two-step method is exact on a 2×2 system.

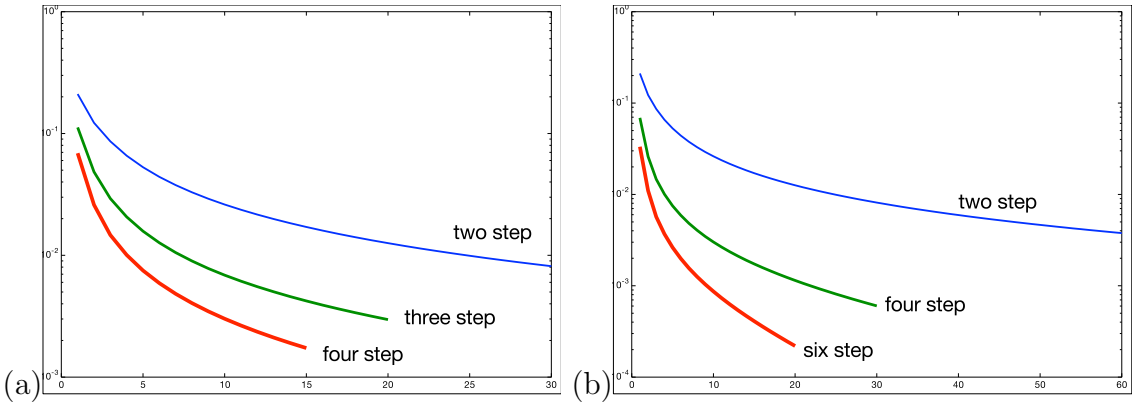


Figure 3: Performance of k-step methods. Relative eigenvalue error for the largest eigenvalue of the diagonal matrix A with eigenvalues $1, \ldots, n$, with $n = 1000$. Starting vector is $\mathbf{y} = (1, 1, \ldots, 1)^t$. The horizontal axis is the number of iterations of the k-step method. (a) The total number of matrix-vector operations is 60 for all three values of k. Uses algorithm (4.2). (b) The total number of matrix-vector operations is 120 for all three values of k. Uses algorithm (A.4).

3.2 Comparison of two-step and power methods

To compare properly the two-step method with the power method, we need to consider comparable work. When the matrix-vector multiplication is the dominant work, then a proper comparison would be to compare the two-step method with two steps the power method. The resulting eigenvalue errors are given in Figure 2(b). We see that the two-step method initially outperforms two steps of the power method. But as the iterations continue, the results converge, and asymptotically for large iteration numbers, their performance is identical (data not shown).

4 k-step methods

General k-step Krylov methods are studied extensively in [11]. We recall some notation from [11]. Start with a vector $\mathbf{y}_1$, and define $\mathbf{y}_{j+1} = A\mathbf{y}_j$, for $j = 1, \ldots, k-1$. Analogous to (2.4) we solve for coefficients $a_1, \ldots, a_k$ satisfying

$$A \sum_j a_j \mathbf{y}_j = \lambda \sum_j a_j \mathbf{y}_j. \tag{4.1}$$

Taking dot products, we get the generalized eigenvalue problem

$$K\mathbf{a} = \lambda M\mathbf{a},$$

where

$$K_{ij} = \mathbf{y}_i^t A \mathbf{y}_j, \qquad M_{ij} = \mathbf{y}_i^t \mathbf{y}_j.$$

Therefore $\mathbf{a}$ can be determined by solving the eigenproblem

$$M^{-1}K\mathbf{a} = \lambda\mathbf{a}. \tag{4.2}$$

The output of the k-step method is defined to be

$$\mathbf{y} = \sum_{i=1}^{k} a_i \mathbf{y}_i,$$

where $\mathbf{a}$ is the eigenvector corresponding to the extreme eigenvalue for (4.2).

In Figure 3(a), we depict the error in the extreme eigenvalue resulting from multiple steps of the k-step algorithm for a matrix with eigenvalues $1, \ldots, n$, with $n = 1000$. Since one step of the k-step algorithm involves k matrix-vector operations, we have compared algorithms with the same number of matrix-vector operations: 30 2-step iterations, 20 3-step iterations, and 15 4-step iterations, each with a total of 60 matrix-vector operations.

Although the naive approach (4.2) to the k-step method works for small k, it fails for larger k, due to ill conditioning. Thus we orthogonalize the Krylov vectors.

4.1 Krylov vector orthogonalization

The algorithm for computing (4.2) is changed as follows. Starting with a vector $\hat{\mathbf{y}}_1$, we define Krylov vectors $\hat{\mathbf{y}}_{j+1} = A\hat{\mathbf{y}}_j$, for $j = 1, \ldots, k$. These vectors are orthogonalized by the modified Gram–Schmidt algorithm [2, 10] to get $\mathbf{y}_1, \ldots, \mathbf{y}_k$. We provide the details in appendix A.

In Figure 3(b), we show that the use of orthogonal Krylov vectors allows us to extend to more steps, but for larger steps the algorithm fails due to the increasing condition number of K, as indicated in Figure 4(a).

In Figure 4(b), we compare k-step methods with two steps of $k/2$ step methods. We see that it is more effective to use one iteration of the k-step method.

4.2 Arnoldi algorithm to the rescue

The Arnoldi algorithm involves only a small change in the order of orthogonalization and multiplication by the matrix A, as explained in [11]. But this subtle change makes the algorithm far more robust, as shown in [11, Figure 1].

4.3 LOBPCG and k-step methods

Locally Optimal Block Preconditioned Conjugate Gradient (LOBPCG) methods [7, 1, 6] are discussed in [11]. Their close relationship with k-step methods is explained there.

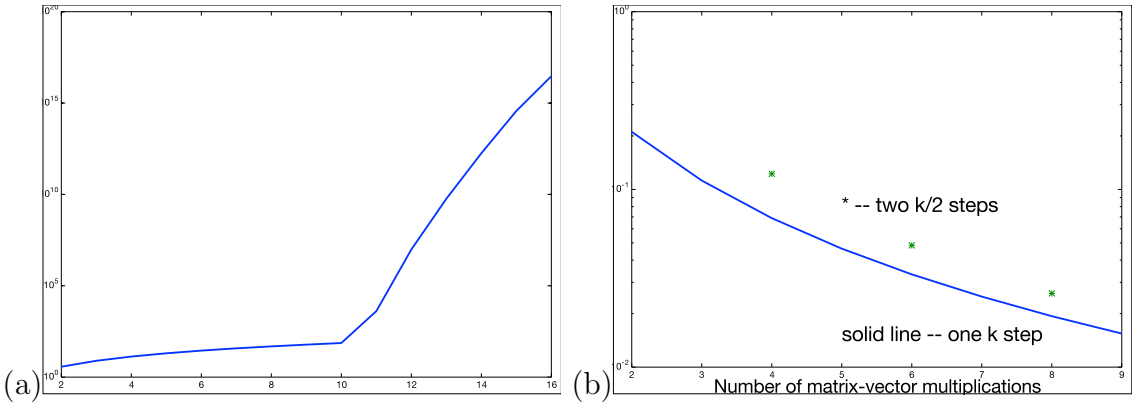


Figure 4: (a) Condition number of K for k-step methods for the diagonal matrix A with eigenvalues $1, \ldots, n$, with $n = 1000$. Starting vector is $\mathbf{y} = (1, 1, \ldots, 1)^t$. The horizontal axis is k. (b) Comparison of k-step methods and two steps of $k/2$ step methods. Relative eigenvalue error for the largest eigenvalue of the diagonal matrix A with eigenvalues $1, \ldots, n$, with $n = 1000$. Starting vector is $\mathbf{y} = (1, 1, \ldots, 1)^t$. Uses algorithm (A.4). Solid line is the error for one iteration of the k-step method. The asterisks (*) indicate the error for two iterations of the $k/2$-step method, plotted at k so that the number of matrix-vector multiplications is comparable.

5 Conclusions and perspectives

We showed that small eigenproblems can be solved to enhance the performance of the power method. We demonstrate instabilities of different implementations of the k-step method, the Arnoldi approach being the most stable. We have examined the performance as a function of k for various test matrices.

The Arnoldi algorithm is typically thought of as providing a matrix factorization [8]. We are viewing the Arnoldi algorithm as a particular numerical implementation of an approximation problem using Krylov vectors. It is striking that such diverse views of the same algorithm (Arnoldi) are available, including the popular LOBPCG method [7, 1, 6], as described in [11].

It is equally noteworthy that the three implementations discussed here of the same k-step Krylov method, which are identical in exact arithmetic, can have such divergent behavior in floating point arithmetic. It would be ideal to have an analysis of the differences of the algorithms that could explain their behavior when subjected to small imperfections.

Acknowledgments

SP is supported in part by the National Science Foundation NSF DMS-1852876 and NSF DMS-2011519. Both authors would like to thank Nilima Nigam for valuable feedback.

References

[1] Peter Benner and Thomas Mach. Locally optimal block preconditioned conjugate gradient method for hierarchical matrices. *PAMM*, 11(1):741–742, 2011.

[2] Åke Björck. Numerics of Gram–Schmidt orthogonalization. *Linear Algebra and Its Applications*, 197:297–316, 1994.

[3] Eric Cancès and L. Ridgway Scott. van der Waals interactions between two hydrogen atoms: The Slater-Kirkwood method revisited. *SIAM Journal on Mathematical Analysis*, 50(1):381–410, 2018.

[4] Patrick Farrell, Lawrence Mitchell, L. Ridgway Scott, and Florian Wechsung. Robust discretization and multigrid solution for incompressible and nearly incompressible continua. *TBD*, 2021.

[5] Vicente Hernandez, Jose E. Roman, and Vicente Vidal. SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):351–362, 2005.

[6] Andrew Knyazev. Recent implementations, applications, and extensions of the Locally Optimal Block Preconditioned Conjugate Gradient method (LOBPCG). *arXiv preprint arXiv:1708.08354*, 2017.

[7] Andrew V. Knyazev. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM Journal on Scientific Computing*, 23(2):517–541, 2001.

[8] Richard B. Lehoucq and Danny C. Sorensen. Deflation techniques for an implicitly restarted arnoldi iteration. *SIAM Journal on Matrix Analysis and Applications*, 17(4):789–821, 1996.

[9] Nilima Nigam and Sara Pollock. A simple extrapolation method for clustered eigenvalues. *Numerical Algorithms*, DOI 10.1007/s11075-021-01108-7, 2021.

[10] Christopher C. Paige, Miroslav Rozložník, and Zdenvek Strakos. Modified Gram-Schmidt (MGS), least squares, and backward stability of MGS-GMRES. *SIAM Journal on Matrix Analysis and Applications*, 28(1):264–284, 2006.

[11] Sara Pollock and L. Ridgway Scott. Extrapolating the Arnoldi algorithm to improve eigenvector convergence. *International Journal of Numerical Analysis & Modeling*, submitted, 2021.

[12] Jose E. Roman, Carmen Campos, Eloy Romero, and Andrés Tomás. SLEPc users manual. D. Sistemes Informatics i Computació Report No. DSIC-II/24/02, Universitat Politecnica de Valencia, Valencia, Spain, 2015.

[13] L. Ridgway Scott. *Numerical Analysis*. Princeton Univ. Press, 2011.

[14] L. Ridgway Scott. Kinetic energy flow instability with application to Couette flow. Research Report UC/CS TR-2020-07, Dept. Comp. Sci., Univ. Chicago, 2020.

[15] Gilbert W. Stewart. A Krylov–Schur algorithm for large eigenproblems. *SIAM Journal on Matrix Analysis and Applications*, 23(3):601–614, 2002.

A Modified Gram–Schmidt Orthogonalization

First define $\mathbf{y}_1 = \nu_1 \hat{\mathbf{y}}_1$ where $\nu_1 = \|\hat{\mathbf{y}}_1\|^{-1}$. Then for $j = 2, 3, \ldots, k$, define

$$\begin{aligned} \widetilde{\mathbf{y}}_j &= \hat{\mathbf{y}}_j - \sum_{l=1}^{j-1} c_{j,l}\mathbf{y}_l\,, \qquad c_{j,l} = \hat{\mathbf{y}}_j^{\,t}\,\mathbf{y}_l\,, \quad l = 1, \ldots j-1, \\ \nu_j &= \|\widetilde{\mathbf{y}}_j\|^{-1}, \qquad \mathbf{y}_j = \nu_j\,\widetilde{\mathbf{y}}_j. \end{aligned} \tag{A.1}$$

so that the matrix $C = (c_{j,l})$ is a strictly lower triangular matrix.

A.1 Orthogonality

Note that for $j = 2$

$$\mathbf{y}_1^t\widetilde{\mathbf{y}}_2 = \mathbf{y}_1^t\hat{\mathbf{y}}_2 - c_{2,1}\mathbf{y}_1^t\mathbf{y}_1 = \mathbf{y}_1^t\hat{\mathbf{y}}_2 - c_{2,1} = \hat{\mathbf{y}}_2^t\mathbf{y}_1 - c_{2,1} = 0.$$

We claim by induction that, for $j \geq 2$,

$$\mathbf{y}_i^t\widetilde{\mathbf{y}}_j = 0 \quad \forall\, 1 \leq i < j. \tag{A.2}$$

We have shown this holds for $j = 2$. Assuming (A.2) holds for a given j, we have

$$\mathbf{y}_i^t\mathbf{y}_j = \delta_{ij} \quad \forall\, 1 \leq i \leq j.$$

Thus (A.1) implies that, for any $i \leq j$, we have

$$\mathbf{y}_i^t\widetilde{\mathbf{y}}_{j+1} = \mathbf{y}_i^t\hat{\mathbf{y}}_{j+1} - \sum_{l=1}^{j} c_{j+1,l}\mathbf{y}_i^t\mathbf{y}_l = c_{j+1,i} - \sum_{l=1}^{j} c_{j+1,l}\delta_{il} = 0.$$

This completes the induction step to prove (A.2).

A.2 Computing K

In order to compute (4.1), note that

$$\frac{1}{\nu_j}A\mathbf{y}_j = A\widetilde{\mathbf{y}}_j = A\hat{\mathbf{y}}_j - \sum_{l=1}^{j-1} c_{j,l}A\mathbf{y}_l = \hat{\mathbf{y}}_{j+1} - \sum_{l=1}^{j-1} c_{j,l}A\mathbf{y}_l. \tag{A.3}$$

Taking dot products, we see that (4.1) is equivalent to

$$K\mathbf{a} = \lambda\mathbf{a}, \tag{A.4}$$

where again $K_{ij} = \mathbf{y}_i^t A\mathbf{y}_j$, with the new definition of $\mathbf{y}_i$, but the matrix M is now the identity. Now we give the formulæ for K. First

$$K_{1,1} = \mathbf{y}_1^t A\mathbf{y}_1 = \nu_1\mathbf{y}_1^t A\hat{\mathbf{y}}_1 = \nu_1\mathbf{y}_1^t\hat{\mathbf{y}}_2 = \nu_1\hat{\mathbf{y}}_2^t\mathbf{y}_1 = \nu_1 c_{2,1}.$$

Then for $j = 2, 3, \ldots, k$, we can compute from (A.3) that

$$K_{1,j} = \nu_j\Big(\mathbf{y}_1^t\hat{\mathbf{y}}_{j+1} - \sum_{l=1}^{j-1} c_{j,l}K_{1,l}\Big) = \nu_j\Big(c_{j+1,1} - \sum_{l=1}^{j-1} c_{j,l}K_{1,l}\Big), \tag{A.5}$$

where we define $c_{k+1,i} = \hat{\mathbf{y}}_{k+1}^t\mathbf{y}_i$ for $i = 1, \ldots, k$. This defines C as a strictly lower triangular $(k+1)\times(k+1)$ matrix. In the case that A is symmetric, (A.5) also defines $K_{i,1} = K_{1,i}$, $i = 2, \ldots, k$. Then for $i = 2, \ldots, k$, we have from (A.3) that

$$K_{i,j} = \nu_j\Big(\mathbf{y}_i^t\hat{\mathbf{y}}_{j+1} - \sum_{l=1}^{j-1} c_{j,l}K_{i,l}\Big) = \nu_j\Big(c_{j+1,i} - \sum_{l=1}^{j-1} c_{j,l}K_{i,l}\Big) \quad \text{for } j = 2, 3, \ldots, i.$$

This defines the lower triangle of K. In the case that A is symmetric, this also defines $K_{j,i} = K_{i,j}$, for $j = 2, 3, \ldots, k$ and $i = 2, \ldots, j$, which is the upper triangle of K.

Since $H = K$ in exact arithmetic, K is upper Hessenberg (and tridiagonal if A is symmetric) in exact arithmetic.