Complete Iterative Method for Computing Pseudospectra

Thierry Braconnier
CERFACS, 42 avenue G. Coriolis, 31057 Toulouse, France.

May 28, 1997

Keywords: Pseudospectra, singular values.

Abstract

Efficient codes for computing pseudospectra of large sparse matrices usually use a Lanczos type method with the shift and invert technique and a shift equal to zero. Then, these codes are very efficient for computing pseudospectra on regions where the matrix is nonnormal (because $\|(A-zI)^{-1}\|_2$ is large) but they lose their efficiency when they compute pseudospectra on regions where the spectrum of A is not sensitive ($\|(A-zI)^{-1}\|_2$ is small). A way to overcome this loss of efficiency using only iterative methods associated with an adaptive shift is proposed.

1 Introduction

The ε-pseudoeigenvalue and ε-pseudospectrum are defined as:

- λ is an ε-pseudoeigenvalue of A if

$$\lambda \text{ is an eigenvalue of } A+E \text{ with } \|E\|_2 \leq \varepsilon \|A\|_2$$

- The ε-pseudospectrum of A is defined by

$$\Lambda_\varepsilon(A) = \{z \in \mathbb{C};\, z \text{ is an } \varepsilon\text{−pseudoeigenvalue of } A\}$$

For a fixed ε, the contour of $\Lambda_\varepsilon(A)$ can be defined as $\{z \in \mathbb{C};\ \|A\|_2 \|(A-zI)^{-1}\|_2 = \varepsilon^{-1}\}$. The graphical representation of

$$z \mapsto \log(\|A\|_2\ \|(A-zI)^{-1}\|_2) \tag{1}$$

is called the *spectral portrait* of A [9] and the level curves of (1) for given values of ε, when no ambiguity holds, are also called the pseudospectra [15].

To generate pseudospectra, the following steps can be taken:

1. Determine a region $\Re$ of interest in $\mathbb{C}$.
2. Discretise the region $\Re$.
3. For each point z of the discretisation, compute the value $\|(A-zI)^{-1}\|_2$.
4. Use a visualisation tool to display the computed values for $z \in \Re$

Note that a way for determining $\Re$ based on the computation of the field of values of A has been proposed in [5]. The goal of this paper is to examine how the main computation, i.e., the computation of $\|(A - zI)^{-1}\|_2$ (Step 2) can be done efficiently for any complex number $z \in \Re$. One could use a singular value decomposition (SVD), since $\|(A - zI)^{-1}\|_2 = \sigma_{\min}^{-1}(A - zI)$, where $\sigma_{\min}(A - zI)$ is the smallest singular value of $A - zI$. However, this approach is not suitable for large matrices. For such cases, techniques using iterative eigensolvers (a Lanczos type methods) have been developed [7, 14, 4].

A description of algorithms proposed in [14, 4] and a description of a new approach based on adaptive shift technique is given in Section 2. Section 3 gives a comparison on a numerical example.

2 Computation of $\|(A - zI)^{-1}\|_2$

2.1 Usual Approaches for Computing $\|(A - zI)^{-1}\|_2$

The methods proposed in [14] (referred to as M1) and in [4] (referred to as M2) use the Lanczos method [12]. They also use the shift and invert technique [8] with a shift equal to zero. They then compute the largest eigenvalue of $A_z = [(A - zI)^*(A - zI)]^{-1}$. The efficiency of this implementation increases when ε is small because the target value becomes closer to the shift. In practice, it has been observed that due to the structure of the spectrum of A_z, the number of iterations for the Lanczos process increases with ε. Thus, these codes are very efficient for computing pseudospectra either on regions where the matrix is nonnormal or for z close to an eigenvalue because $\|(A - zI)^{-1}\|_2$ is large. But they lose their efficiency when they compute pseudospectra on regions where the spectrum of A is not sensitive, i.e., $\|(A - zI)^{-1}\|_2$ is small.

Since for the Lanczos algorithm, the matrix is only accessed by the matrix-vector product $y \leftarrow A_z x$, M1 and M2 perform the computation for each z as follows.

Algorithm 1

1. *Compute the LU factorization of* $A - zI = LU$
2. *Each time we need to perform* $y \leftarrow A_z x$ *we solve* $U^* L^* L U y = x$

2.2 The New Approach

The key point is to use an idea similar to the Wielandt deflation. Let A be a matrix, let x_1 be the normalized right eigenvector of A associated with the eigenvalue λ_1 and let A_1 be given by

$$A_1 = A - \sigma x_1 x_1^*,$$

where σ is a given complex number. Then, an easy derivation shows that $1/(\lambda_1 - \sigma)$ is an eigenvalue of A_1^{-1} and its corresponding eigenvector is x_1.

Adapting this remark for the computation of $\|(A - zI)^{-1}\|_2$, we then apply a Lanczos type algorithm on the matrix

$$B_z = [(A - zI)^*(A - zI) - \sigma u u^*]^{-1},$$

where σ will estimate $\|(A - zI)^{-1}\|_2$ and u will estimate the corresponding right eigenvector of A_z. Then, the computation at each node z can be as follows.

Algorithm 2

1. *Choose* σ *and* u

2. *Perform* m *steps of Lanczos on* B_z

3. *Estimate* $\lambda_{\max}$ *and its corresponding eigenvector* $x_{\max}$

4. *If the stopping criterion is satisfied then stop.*

5. *Set* $\sigma \leftarrow \lambda_{\max}$, $u \leftarrow x_{\max}$, *and goto* 2

For the Lanczos process applied on B_z we need to perform $y \leftarrow B_z x$ as fast as possible. Thanks to the Sherman-Morisson formulae [10], we have

$$(C + ts^*)^{-1} = C^{-1} - C^{-1}t(I + s^*C^{-1}t)^{-1}s^*C^{-1},$$

where C is a matrix and t, s are two vectors. If we set $C = A_z^{-1}$, $t = -\sigma u$ and $s = u$ then the computation of $y \leftarrow B_z x$ can be done for each z as follows.

Algorithm 3 shows how to perform the computation using a direct method.

Algorithm 3

1. *Compute the LU factorization of* $A - zI$

2. *Solve* $U^*L^*LUw = u$

3. *Each time we need to perform* $y \leftarrow B_z x$

   (a) *Solve* $U^*L^*LUy = x$

   (b) *Compute* $\alpha = (u^*y)/(1 - \sigma u^*w)$

   (c) *Set* $y \leftarrow y + \sigma\alpha w$

The Lanczos type method which uses Algorithm 3 is referred to as M3. The interesting idea here is that we can adapt the shift σ at each restart of the Lanczos process but we only need to perform the LU factorization of $A - zI$ once for each z. Note that Step 1 is performed once for each z and Step 2 is performed once when σ and u are determined, i.e., when the Lanczos process is restarted.

Since direct methods can require too much memory and in general are not efficiently implemented in parallel it can be interesting to perform the computation of $y \leftarrow B_z x$ using an iterative method. Note that B_z is a symmetric matrix and that as σ approaches $\|(A - zI)^{-1}\|$ the matrix B_z^{-1} becomes more and more indefinite. Recently, it has been shown that the `symmlq` method is one of the most robust Krylov methods for indefinite system [13].

Algorithm 4

1. *Compute an approximate inverse* M *of* $A - zI$

2. *Solve* $M^*Mw = u$

3. *Each time we need to perform* $y \leftarrow B_z x$

   (a) *Solve resp.* $M^*My = x$

   (b) *Compute* $\alpha = (u^*y)/(1 - \sigma u^*w)$

(c) Set $y \leftarrow y + \sigma\alpha w$

The Lanczos type method implemented using Algorithm 4 is referred to as M4. The preconditioner M we use for the `symmlq` method is the incomplete LU factorization of $A - zI$ with no fill-in allowed, i.e., `ILU(0)` [1]. We next summarize the five methods used for the comparisons and give the corresponding parameters for their use.

2.3 Summary of the Methods

The first method is the SVD method. Its storage is of order n^2 and it computes *all* the SVD decomposition of A. See [10] for more details.

The method M1 has been proposed by [14]. It uses a Lanczos method implemented with the shift and invert technique and a shift equal to zero. At each step of the Lanczos process, M1 computes the Ritz values and check for convergence. If the convergence is not achieved then the next step of the Lanczos process is performed until a given fixed size $m_{\max}$ ($m_{\max} = 30$ for the test example) for which the whole process is restarted using the computed eigenvector as starting vector. The required storage is of order $m_{\max} n$.

The method M2 has been proposed by [4]. It also uses a Lanczos method implemented with the shift and invert technique and a shift equal to zero. This time, m steps ($m = 16 \approx \sqrt{n}/216$ for the test example) of the Lanczos process are performed and the convergence check is only done at this iteration. This Lanczos method is associated with Chebychev acceleration and uses a continuation technique [11, 6]. The required storage is of order $n^{\frac{3}{2}}$.

The method M3 uses a Lanczos method implemented with an adaptive shift technique and the direct method LU for computing the Lanczos vectors. The size of the projection has been set to $m = 3$ because the method needs good estimates for the shift σ and the vector u. The continuation technique is also crucial in order to start the computation at the next node of the grid with a good initial guess u. The required storage is of order $3n$.

The method M4 is the iterative version of method M3 where the LU factorization is replaced by the preconditioned `symmlq` method. The required storage is of order $3n$. Note that the storage required for the preconditioner can be much smaller than for storing the L and U factors.

3 Comparison of different methods

The test matrix A is a sparse normally distributed random matrix (Matlab function `sprandn.m`) of size $n = 1000$. This matrix is reordered using the symmetric reverse Cuthill-McKee permutation (Matlab function `symrcm.m`). Figure 1 shows the pattern of A.

We have chosen to varry z from -20 to 20 with a stepsize equal to 0.5. Figure 2 shows the function $z \mapsto -\log_{10}(\|(A - zI)^{-1}\|_2)$ computed using the SVD method. We can see that $\|(A - zI)^{-1}\|_2$ takes a wide range of values between 0.1 and 10^{12}. Thus, the classical approaches M1 and M2 shall perform well for z corresponding to large values of $\|(A - zI)^{-1}\|_2$ and shall lose efficiency for $|z|$ close to 20.

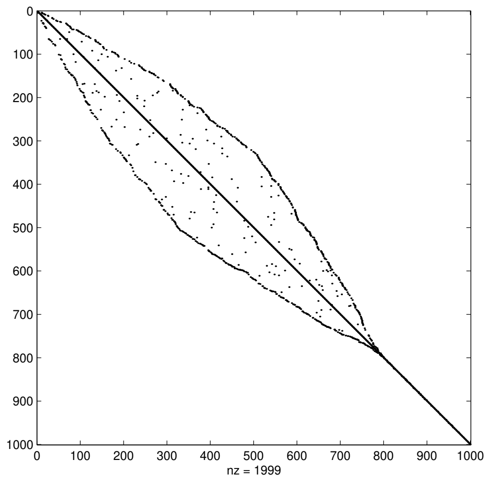

Figure 1: *Pattern of A.*

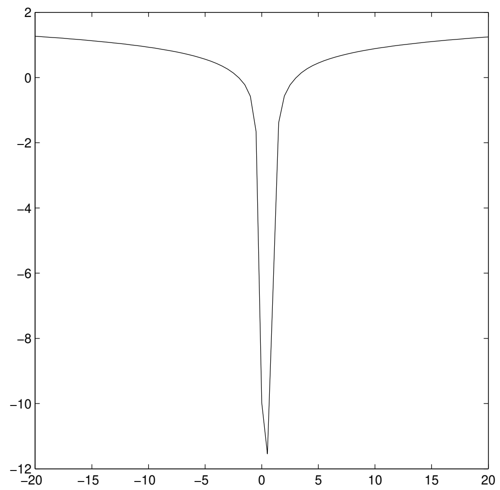

Figure 2: $z \mapsto -\log_{10}(\|(A - zI)^{-1}\|_2)$.

Table 1 shows the results for the computation using the five methods. The column labeled T gives the computational time, the column labeled T/T_{M4} gives the speed-ups relative to M4. L represents the number of Lanczos steps for computing the 81 values of $\|(A - zI)^{-1}\|_2$ and L/L_4 the corresponding ratio with respect to M4. We can see that the new approaches (M3 and M4) are two times faster than the classical approaches (M1 and M2) and two orders of magnitude faster than SVD on this example. Note that all the methods produce values as shown in Figure 2.

Method used	T (in sec)	T/T_{M4} (in sec)	L	L/L_4
SVD	9318.3	101	–	–
M1	152.4	1.65	2097	3.41
M2	145.8	1.57	1326	2.16
M3	73.2	0.79	597	0.97
M4	92.3	–	615	–

Table 1: *Experiments on A.*

Then, for each node, we have plotted the size of the projection $m_{\mathbf{CV}}$ required to achieve a stopping criterion equal to 10^{-4}. Figures 3 to 6 and the two last columns of Table 1 allows us to understand the difference observed in Table 1.

We can see on Figure 3 that the shape for M1 is very close to the shape of the function given by Figure 2. It means that the number of iterations for the Lanczos process increases with ε when M1 is used. For M2, except on the first node, the size remains equal to $m_{\mathbf{CV}} = 16$ for the whole computation as shown on Figure 4. On this example, the continuation technique is very efficient.

For the new approaches, i.e, M3 and M4, we can see on Figures 5 and 6 that the size $m_{\mathbf{CV}}$ does not depend on the value of $\|(A - zI)^{-1}\|_2$ except for the nodes where $\|(A - zI)^{-1}\|_2$ is very large, i.e. the shift σ is very small. For such values of $\|(A - zI)^{-1}\|_2$, the singular values are clustered near zero and we think that for the new approaches, the convergence rate can be affected by the shift.

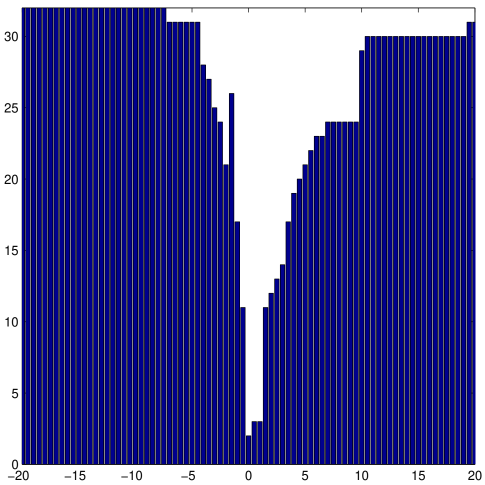

Figure 3: *M1:* $z \mapsto m_{\mathtt{CV}}$.

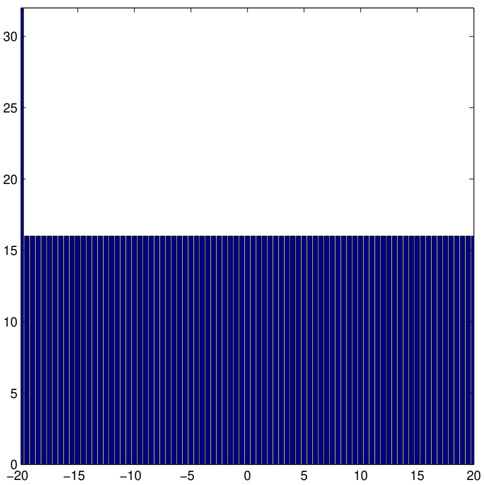

Figure 4: *M2:* $z \mapsto m_{\mathtt{CV}}$.

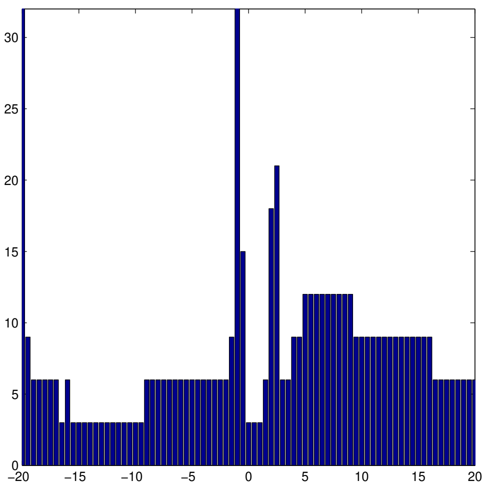

Figure 5: *M3:* $z \mapsto m_{\mathtt{CV}}$.

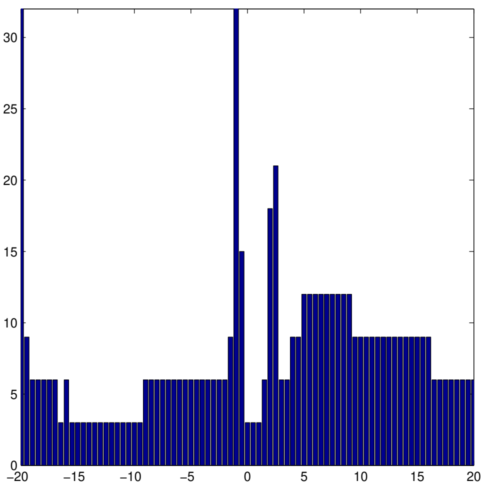

Figure 6: *M4:* $z \mapsto m_{\mathtt{CV}}$.

The last comment concerns the fact the M3 is faster than M4. The main reason for this is, as for all iterative methods, the choice of the preconditioner.

Looking at the previous results, the following trick can be proposed. For each nide z,

1. Perform the LU factorization or compute an approximate inverse of $A - zI$
2. Perform m Lanczos step using M1 or M2,
3. Compute $\lambda_{\max}$ and the corresponding eigenvector $x_{\max}$
4. If $\lambda_{\max} \leq \tau$ then continue the Lanczos method using M1 or M2,
5. Else continue the Lanczos method using M3 or M4
6. Go to Step 3.

By this way, the regions where $\|(A - zI)^{-1}\|_2$ is large (small shifts are needed), we use the classical approaches and when $\|(A - zI)^{-1}\|_2$ is samll (larger shifts are needed), we then we use the new approaches which is more efficient on these regions. The threshold τ for switching from classical approaches to new approaches seems experimently to be close to $\tau = 0.1$.

4 Conclusion

We propose a Lanczos method implemented with an adaptive shift technique for computing pseudospectra. We illustrate that this new approach allows us to efficiently compute pseudospectra on any region $\Re$, i.e., for a wide range of values of $\|(A - zI)^{-1}\|_2$. We have seen that this method can be implemented using only an iterative method. For such methods, the issue of preconditioning should be studied more and new preconditioner such as those proposed in [3, 2] should be investigated for pseudospectra computations.

References

[1] O. Axelsson. Survey of preconditioned iterative methods for linear systems of equations. *BIT*, 25:166–187, 1985.

[2] J. Baglama, D. Calvetti, G. H. Golub, and L. Reichel. Adaptively preconditioned GMRES algorithms. In *Proceedings of the SIAM Sparse Matrices Conference*, Philadelphia, 1996. SIAM.

[3] M. Benzi and M. Töuma. A comparison of some preconditioning techniques for general sparse matrices. In *IMACS, Iterative Methods in Linear Algebra II*, pages 191–203, 1995.

[4] T. Braconnier and N. J. Higham. Computing the field of values and pseudospectra using the Lanczos method with continuation. *BIT*, 36(3):422–440, 1996.

[5] T. Braconnier, R. A. McCoy, and V. Toumazou. Using the field of values for pseudospectra generation. Technical report, CERFACS, 1997. Preprint.

[6] M. Brühl. A curve tracing algorithm for computing the pseudospectrum. *BIT*, 36:3, 1996.

[7] J. F. Carpraux, J. Erhel, and M. Sadkane. Spectral portrait for non Hermitian large sparse matrices. *Computing*, 53:301–310, 1994.

[8] T. Ericsson and A. Ruhe. The spectral transformation Lanczos method. *Math. Comp.*, 34:1251–1268, 1980.

[9] S. K. Godunov. Spectral portraits of matrices and criteria of spectrum dichotomy. In J. Herzberger and L. Atanassova, editors, *International symposium on computer arithmetic and scientific computation*, Oldenburg, Germany, October 1991. North-Holand.

[10] G. Golub and C. Van Loan. *Matrix computations.* Johns Hopkins University Press, 1989. Second edition.

[11] S. H. Lui. Computation of pseudospectra by continuation. *SISC*, 1995. To appear.

[12] B.N. Parlett. *The Symmetric Eigenvalue Problem.* Prentice-Hall, Englewood Cliffs, 1980.

[13] G. L. G. Sleijpen, H. A. Van der Vorst, and J. Modersitzky. The main effects of rounding errors in Krylov solvers for symmetric linear systems. Technical Report 1006, University of Utrecht, Dept. of Mathematics, 1997.

[14] V. Toumazou. *Portraits spectraux de matrices : un outil d'analyse de la stabilité.* Ph. D. dissertation, Université Raymond Poincaré, Nancy I, CERFACS Tech. Rep. TH/PA/96/46, 1996.

[15] L. N. Trefethen. Pseudospectra of matrices. In D. F. Griffiths and G. A. Watson, editors, *14th Dundee Biennial Conference on Numerical Analysis*, 1991.