An evaluation of software for computing eigenvalues of sparse nonsymmetric matrices. Part I.

R. B. Lehoucq[1] and J. A. Scott[2]

[1]Argonne National Laboratory,
Mathematics and Computer Science Division,
`lehoucq@mcs.anl.gov`
`http://www.mcs.anl.gov/home/lehoucq/index.html`

[2]Computing and Information Systems Department,
Atlas Centre, Rutherford Appleton Laboratory.
`sct@letterbox.rl.ac.uk`
`http://www.cis.rl.ac.uk/people/jscott/contact.html`

Eigenvalues and Beyond.
17 October 1995.

International Linear Algebra Year & SIAM National Meeting.

Outline

- Introduction
- Algorithms
- Software review
  - Key features of codes
  - User interface
  - Input parameters
  - Stopping criteria
  - Error handling
  - Storage requirements
- Numerical experiments
  - Design of experiments
  - Test examples
  - Verification
  - Findings
- Future directions

Introduction

- Increased interest in computing selected eigenvalues (and, optionally, eigenvectors) of large sparse nonsymmetric matrices.
- High quality software is now appearing.
- Aim to review, compare, and evaluate this software.
  - Performance
  - Storage
  - Accuracy
  - Robustness
  - User interface

Notation

- $\mathbf{Ax} = \mathbf{x}\lambda$
- $\mathbf{n}$ denotes the order of $\mathbf{A}$.
- $\mathbf{m}$ denotes the subspace dimension.
- $\mathbf{X_m} = \left[\, \mathbf{X_r} \;\; \bar{\mathbf{X}}_{\mathbf{m-r}} \,\right]$ denotes the matrix representation of the subspace.
- $(\mathbf{z}, \theta)$ denotes an eigenpair of the projection of $\mathbf{A}$ onto the column space of $\mathbf{X_m}$.
- Computed eigenpairs are called *Ritz* pairs, $\mathbf{Ay} \approx \mathbf{y}\theta$ where $\mathbf{y} = \mathbf{X_m z}$.
- $\mathbf{r}$ denotes the number of desired Ritz values.
- $\mathbf{T_m}$ denotes the quasi-triangular Schur matrix associated with the projection.
- $\mathbf{u}$ denotes relative machine precision.
- ϵ denotes the user prescribed convergence tolerance.

Algorithms

Three general classes of methods are currently widely used:

- Subspace (simultaneous) iteration
- Arnoldi methods
- Lanczos methods

A comprehensive comparative study of these methods for large sparse nonsymmetric matrices has not been carried out.

Other relevant material

- Survey by Parlett 1984 for symmetric matrices.
- Progress report by Bai 1995 for nonsymmetric matrices.
- Recent books by Saad '92 and Chatelin '93 on eigenvalue algorithms.

Subspace Iteration

1. *Start:* Choose an initial set of vectors $\mathbf{X_m} \leftarrow [\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_m}]$.
2. *Iteration:* Until convergence **do**
   - Compute $\mathbf{X_m} \leftarrow \mathbf{A^p X_m}$.
   - Possibly compute $\mathbf{Q_m R_m} = \mathbf{X_m}$ and then $\mathbf{X_m} \leftarrow \mathbf{Q_m}$.
   - Projection $\mathbf{B_m} = (\mathbf{X_m^T X_m})^{-1} \mathbf{X_m^T A X_m}$.
   - Form $\mathbf{T_m} = \mathbf{Z_m^T B_m Z_m}$ and set $\mathbf{X_m} \leftarrow \mathbf{X_m Z_m}$.

- Popular in structural engineering
- Generalises the power method, $1 < m \ll n$ is the subspace dimension.
- Computes r ($r < m \ll n$) eigenvalues of largest modulus
- Requires matrix-matrix products $\mathbf{AX_m}$.
- Convergence theory exists (Stewart '76, Watkins and Elsner '91, Saad '92, Chatelin '93).

Arnoldi's Method

- Krylov subspace method
- Builds an orthogonal basis for

$$\mathbf{K_m}(\mathbf{v_1}, \mathbf{A}) = [\mathbf{v_1}, \mathbf{A v_1}, \ldots, \mathbf{A^{m-1} v_1}]$$

- Truncated reduction of $\mathbf{A}$ to Hessenberg form $\mathbf{H_m}$

$$\mathbf{AV_m} = \mathbf{V_m H_m} + \mathbf{h_{m+1,m} v_{m+1} e_m^T}$$

$$\mathbf{V_m^T A V_m} = \mathbf{H_m}$$

- As m increases, extremal well-separated eigenvalues of $\mathbf{A}$ are well approximated by a subset of the eigenvalues of $\mathbf{H_m}$.
- Storage increases with m.
- Restarting limits storage requirements (select an improved $\mathbf{v_1}$)
  - explicit restarting (Saad 1980)
  - implicit restarting (Sorensen 1992)
- Requires matrix-vector products $\mathbf{Ax}$
- Method can be "blocked".

Lanczos Method

- Krylov subspace method
- Builds biorthogonal bases for $\mathbf{K_m}(\mathbf{v_1}, \mathbf{A})$ and $\mathbf{K_m}(\mathbf{u_1}, \mathbf{A^T})$
- Can be regarded as truncated procedure for the tridiagonalization of $\mathbf{A}$

$$\mathbf{AV_m} = \mathbf{V_m T_m} + \delta_{\mathbf{m+1}} \mathbf{v_{m+1} e_m^T}$$

$$\mathbf{A^T W_m} = \mathbf{W_m T_m^T} + \beta_{\mathbf{m+1}} \mathbf{w_{m+1} e_m^T}$$

$$\mathbf{W_m^T A V_m} = \mathbf{T_m}$$

- As m increases, extremal well-separated eigenvalues of $\mathbf{A}$ are well approximated by a subset of the eigenvalues of $\mathbf{T_m}$.
- Rigorous convergence theory is lacking
- Storage limited to $6n$ ... cost is a biorthogonality strategy (Day 1993)
- Requires matrix-vector products $\mathbf{Ax}$ *and* $\mathbf{A^T x}$
- Method can be "blocked".
- Potentially unstable. Care needed to prevent breakdown (look-ahead strategies).

Software

Many research codes developed

Codes developed as part of large packages (e.g. by engineers)

Little library-quality software

Our criteria for selection

- code must be widely available (e.g. through netlib or a software library)
- documentation must be provided with the code.
- C or FORTRAN implementations.

Subspace iteration

`LOPSI`	Stewart and Jennings	1981	TOMS
`SRRIT`	Stewart and Bai	1978,1993 (submitted)	TOMS
`EB12`	Duff and Scott	1991	HSL

Arnoldi

`ARNCHEB`	Braconnier	1993	ftp
`EB13`	Scott	1993	HSL
`ARPACK`	Sorensen, Lehoucq and Vu	1995	netlib

Lanczos

`NSLEVAL`	Cullum and Willoughby	1986	netlib
`EIGLAL`	Freund and Nachtigal	1990	netlib
`ABLE`	Bai, Day and Ye	Under development	
`MSC`	MacNeal-Schwendler Co.	Licensed	

This study currently covers subspace iteration and Arnoldi.

LOPSI

- W.J. Stewart and A. Jennings 1981.
- Computes r eigenvalues of largest modulus and corresponding eigenvectors.
- Combines subspace iteration with a *lopsided,* oblique projection.
- Avoids orthogonalising sets of vectors.
- $\mathbf{A}$ must be supplied in a fixed format.
- Includes locking if $r > 1$.

SRRIT

- Original version G.W. Stewart 1978.
- Revised version G.W. Stewart and Bai 1993.
- Computes r eigenvalues of largest modulus.
- Computes a partial Schur form.
- Gram-Schmidt with reorthogonalisation used to orthogonalise sets of vectors.
- User must supply subroutine to form $\mathbf{AX_m}$.
- Includes locking if $r > 1$.
- Groups equimodular eigenvalues.
- Exploits BLAS and LAPACK.

EB12

- Duff and Scott 1991.
- Computes r right-most eigenvalues or eigenvalues of largest modulus.
- Chebychev polynomials used if right-most eigenvalues sought.
- Computes a partial Schur form.
- Optionally computes eigenvectors.
- Classical Gram-Schmidt used to orthogonalise sets of vectors.
- Reverse communication used for $\mathbf{AX_m}$.
- Includes locking if $r > 1$.
- Exploits BLAS.

ARNCHEB

- Braconnier 1993.
- Explicitly restarted Arnoldi method.
- Implements algorithms of Saad
- Computes r right-most eigenvalues or eigenvalues of largest modulus or eigenvalues of largest imaginary parts.
- Uses approximate eigenvectors to restart.
- (Fixed degree) Chebychev polynomial used when restarting.
- Computes eigenvectors.
- Iterated classical Gram-Schmidt used to orthogonalise sets of vectors.
- User must supply subroutine to form $\mathbf{Ax}$.
- Uses some BLAS, EISPACK and LAPACK.

EB13

- Scott 1993.
- Explicitly restarted Arnoldi method.
- Computes r right-most eigenvalues or eigenvalues of largest modulus or eigenvalues of largest imaginary parts.
- Uses approximate Schur vectors to restart.
- Optionally computes eigenvectors.
- Optionally uses Chebychev polynomials.
- Iterated classical Gram-Schmidt used to orthogonalise sets of vectors.
- Reverse communication used for $\mathbf{Ax}$.
- Includes locking if $r > 1$.
- Exploits BLAS.
- Block option.

ARPACK (Release 2)

- Lehoucq, Sorensen, and Vu 1995.
- Implicitly restarted Arnoldi method.
- Solves $\mathbf{Ax} = \lambda\mathbf{x}$ or $\mathbf{Ax} = \lambda\mathbf{Mx}$ ($\mathbf{M}$ symmetric positive definite or symmetric semi-definite).
- Computes r right-most eigenvalues, or eigenvalues of largest or smallest modulus or eigenvalues of largest or smallest imaginary parts.
- Uses approximate Schur vectors to restart.
- Iterated classical Gram-Schmidt used to orthogonalise sets of vectors.
- Reverse communication used for $\mathbf{Ax}$.
- Spectral transformations also available.
- Allows user to define a polynomial preconditioner on $\mathbf{A}$ through its roots.
- Exploits BLAS and LAPACK (release 2).
- No locking and purging (release 3?).

User interface

- We got all the codes to run.
- But not all were user-friendly.
- Not all had been fully tested.

Our experience was:

- Comprehensive documentation supplied with SRRIT, EB12, EB13, and ARPACK.
- Sample programs included in documentation of EB12 and EB13.
- LOPSI and ARNCHEB documentation provides no assistance in the event of an error.
- ARNCHEB does not document all input & output parameters fully and the code itself does not explain what parameters are used for.
- Our experiences suggest LOPSI and ARNCHEB have not been comprehensively tested. Both codes contained bugs.

Matrix-vector products

- Fixed-format: LOPSI.
- User-supplied subroutine: SRRIT, ARNCHEB.
- Reverse communication: EB12, EB13, ARPACK.

Advantages of reverse communication

- Gives user control.
- Sparsity and structure can be exploited.
- Parallelism or vectorisation can be used fully (no common blocks).
- Preconditioning can be incorporated (e.g. shift-and-invert)
- More than one matrix-vector product subroutine can be used.
- Existing software can be used.

Error handling

- LOPSI - success or failure indicated. Can be wrong! Input data not checked.
- ARNCHEB - No error checking.
- SRRIT, ARPACK - error flag but no printed message. Input checked.
- EB12, EB13 - error flag and error/warning message optionally printed. Input checked.

Printing options

- LOPSI - warning if pivot increased in Cholesky factorization.
- ARNCHEB - monitoring printing (stream 7 and default).
- EB12, EB13 - optional printing of errors and/or warnings.
- EB13, SRRIT, ARPACK - optional monitoring printing.

Input parameters.

Good software should make most decisions automatically.

The user **must** supply:

r	=	number of required eigenvalues
m	=	subspace dimension/ Arnoldi steps
ϵ	=	convergence tolerance
$which$	=	which portion of the spectrum is required

Input parameters (continued).

In addition, parameters for which no default is supplied:

`LOPSI`
- matrix $\mathbf{A}$. (A, IRN, ICN)
- maximum number matrix-vector products
- maximum premultiplications between reorientations ($\mathbf{A}^p$)

`SRRIT`
- matrix-vector product routine
- maximum number of iterations

`EB12`
- none

`ARNCHEB`
- matrix-vector product routine
- ellipse parameter
- degree of Chebychev polynomial
- orthogonalization
- deflation

`EB13`
- block size

`ARPACK`
- maximum number of iterations

Storage requirements

- n = order of $\mathbf{A}$
- m = subspace dimension/ Arnoldi steps
- n_b = block size

Code	Storage
LOPSI	$3n \times m + 4m^2 + O(m)$
SRRIT	$2n \times m + 2m^2 + O(m)$
EB12	$3n \times m + 2m^2 + O(m)$
ARNCHEB	$3n \times (m+5) + 2m^2 + O(m)$
EB13	$3n \times m \times n_b + 2m^2 + O(m)$
ARPACK	$n \times (m+4) + 3m^2 + O(m)$

Conclude:
For a given m, `ARPACK` is most efficient on storage.

Stopping criteria : Subspace iteration codes

- The dominant eigenvalues converge first.
- Test $(j+1)$st eigenvalue only if jth has converged.
- `LOPSI` accepts an eigenvector when approximations become almost stationary
  $$\|(\mathbf{X_m}^{(k)} - \mathbf{X_m}^{(k-1)})_j)\|_\infty < \epsilon$$
  Problems if multiple eigenvalues.
- `SRRIT` and `EB12` demand
  $$\mathbf{AX_r} \approx \mathbf{X_r T_r},$$
  where $\mathbf{T_r} \approx \mathbf{X_r}^T \mathbf{AX_r}$ and $\mathbf{I_r} \approx \mathbf{X_r}^T \mathbf{X_r}$.
- `SRRIT`:
  $$\|(\mathbf{AX_m} - \mathbf{X_m T_m})_j\|_2 < |\theta_j| \approx \epsilon,$$
  where θ_j is the jth eigenvalue of $\mathbf{T_m}$.
  - Nearly equimodular eigenvalues are grouped and tested for convergence.
- `EB12`:
  $$\|(\mathbf{AX_m} - \mathbf{X_m T_m})_j\|_2 < \|(\mathbf{AX_m})_j\|_2 * \epsilon.$$
  Monitors for unacceptably slow convergence.

Stopping criteria : Arnoldi codes

- `ARNCHEB`: uses backward error.
  Computes the Arnoldi and direct backward errors
  $$\frac{h_{m+1,m}|\mathbf{e}_m^T \mathbf{z}|}{\|\mathbf{A}\|_F \|\mathbf{y}\|_2}, \qquad \frac{\|\mathbf{Ay} - \lambda \mathbf{y}\|_2}{\|\mathbf{A}\|_F \|\mathbf{y}\|_2}.$$
  where $\mathbf{z}$ is an eigenvector of $\mathbf{H_m}$.
  Note: computes $\|\mathbf{A}\|_F$ with n matrix-vector products.
- `EB13`: demands
  $$\mathbf{AX_r} \approx \mathbf{X_r T_r},$$
  where $\mathbf{T_r} \approx \mathbf{X_r}^T \mathbf{AX_r}$ and $\mathbf{I_r} \approx \mathbf{X_r}^T \mathbf{X_r}$.
  User chooses to use
  $$\|(\mathbf{AX_m} - \mathbf{X_m T_m})_j\|_2 < \|\mathbf{A}\| \approx \epsilon$$
  (user may supply estimate of $\|\mathbf{A}\|$ ... use `MC41` if $\mathbf{A}^T\mathbf{x}$ available) or
  $$\|(\mathbf{AX_m} - \mathbf{X_m T_m})_j\|_2 < \|(\mathbf{AX_m})_j\|_2 \approx \epsilon.$$
- `ARPACK` uses Arnoldi residual
  $$(h_{m+1,m}|\mathbf{e}_m^T \mathbf{z}|)/|\theta|.$$

Numerical experiments

- Harwell-Boeing collection :
  - real problems.
  - not specifically eigenvalue problems.
- Collection of Bai, Barratt, Day and Dongarra:
  - designed for testing software for sparse unsymmetric eigenvalue problems
  - collection still being developed
  - will be available by ftp
  - matrix-vector product routines supplied

Large real examples are still needed.

Test matrices

Identifier	Order	Number of entries	Description/discipline
PORES2	1224	9613	Oil reservoir simulation.
PORES3	532	3474	Oil reservoir simulation.
GRE1107	1107	5664	Simulation studies in computer systems.
HOR131	434	4710	Flow network problem.
IMPCOLC	137	411	Ethylene plant model.
IMPCOLD	425	1339	Nitric acid plant model.
NNC666	666	4044	Nuclear reactor core modelling.
NNC1374	1374	8606	Nuclear reactor core modelling.
WEST0156	156	371	Chemical engineering plant model.
WEST0167	167	507	Chemical engineering plant model.
WEST02021	2021	7353	Chemical engineering plant model.
CK400*	400	2860	Not available.
CK656*	656	3884	Not available.
PDE2961*	2961	14585	Model PDE eigenvalue problem.
RW5151*	5151	20199	Markov chain modelling: random walk.
BW2000*	2000	7996	Chemical engineering model.
CDDE1*			2-D convection diffusion problem.
TOLOSA*			Stability of aircraft in flight.

* indicates matrix included in the collection of Bai *et al.*

Test environment

- Tests performed on RS/6000 3BT using double precision.
- Vendor-supplied BLAS used.
- Convergence tolerances ranging from $u * 10$ to 10^{-4} were tried.
- Default parameter values used.
  1. An exception was stopping criteria for `EB13`.
  2. `ARNCHEB` had **no** default parameters.
- Codes tested with same subspace dimension.

Verification

Results checked as follows:

- Compute the relative accuracy of the computed eigenvalues (those computed by QR algorithm used if exact ones not known).
- eigenvector residuals
$$\|\mathbf{Ay} - \theta\mathbf{y}\|_2$$
- Rayleigh quotient errors
$$\|\mathbf{y}^T\mathbf{Ay} - \theta\mathbf{y}^T\mathbf{y}\|_2$$
- orthogonality (`SRRIT`, `EB12`, `EB13`, `ARPACK`)
$$\|\mathbf{X}^T\mathbf{X} - \mathbf{I}\|_\infty$$
- Schur form (`SRRIT`, `EB12`, `EB13`, `ARPACK`)
$$\|\mathbf{X}^T\mathbf{AX} - \mathbf{T}\|_\infty$$

First test gives an indication of the forward stability of the software.
Last four tests are for determining backward stability of the software.

First numerical experiment
(from test matrix collection)

TOLOSA matrix arises from the stability analysis of a model of an airplane in flight. Its eigenvalues lie on a parabola in the left half plane that opens to the left. The matrix is non-normal and it's departure from normality increases with the order of the matrix.

We computed the eigenpair of largest imaginary part of the TOLOSA matrix with order up to 2000.
Note: The eigenvalues of largest imaginary part are also those of largest modulus.

Timings for TOLOSA matrix of order 2000.

Computed residuals are of $0(10^{-12})$ and the accuracy of the computed eigenvalues is $\sqrt{u}$.

Table entries give time (seconds) and number of matrix vector products.

Algorithm	Subspace dimension m		
	8	16	32
`SRRIT`	*	*	*
`EB12`	117/26255	72/9919	341/26111
`ARNCHEB`	5.0/4201	6.3/4100	14.9/5484
`ARPACK`	16.3/2528	3.4/422	5.0/422
`EB13`	2.8/1451	6.4/2468	7.1/1657

Note:

- * denotes that code did not converge within $4000m$ matrix-vector products.
- `ARNCHEB` used a degree $m - 1$ Chebyshev polynomial.

Second numerical experiment
(from test matrix collection)

Two dimensional model convection-diffusion problem.

$$-\mathbf{\Delta u}(\mathbf{x},\mathbf{y}) + \rho\nabla\cdot\nabla\mathbf{u}(\mathbf{x},\mathbf{y}) = \lambda\mathbf{u}(\mathbf{x},\mathbf{y})$$

on the unit square $[0,1]\times[0,1]$ with zero boundary data.

Discretize using centered finite differences. Eigenvalues and eigenvectors of the resulting matrix are known explicitly.

- Many eigenvalues have multiplicity two.
- As the mesh size decreases, the relative separation of all the eigenvalues decreases. All are contained within the interval $(\mathbf{0},\ \mathbf{8})$.
- As ρ increases the off-diagonals of the matrix increase in size—pseudo-spectrum becomes larger.

We computed up to $r = 6$ eigenpairs of largest real part for a range of values of ρ and for orders up to $10,000$.
Note: The eigenvalues of largest real part are also those of largest absolute value.

Timings for 2-D Laplacian($\rho = 0$) matrix of order $10,000$ (**symmetric** problem).

Computed residuals are of $0(10^{-12})$ and the accuracy of the computed eigenvalues is $\sqrt{u}$. Table entries give time (seconds) and number of matrix vector products.

Algorithm	Subspace dimension m	
	18	36
`SRRIT`	*	2326/92196
`EB12`	62/8631	153/20099
`ARNCHEB`	*	*
`ARPACK`	171/2625	79/1081
`EB13`	81/4781	115/4263

Note:

- * denotes that code did not converge within $4000m$ matrix-vector products.
- `ARNCHEB` used a degree $m - r$ Chebyshev polynomial.
- `ARPACK, EB13` needed a smaller convergence tolerance to achieve the same accuracy as the subspace codes.

Timings for 2-D convection-diffusion ($\rho = 15$) matrix of order $10,000$.

Computed residuals are of $0(10^{-12})$ and the accuracy of the computed eigenvalues is $\sqrt{u}$. Table entries give time (seconds) and number of matrix vector products.

Algorithm	Subspace dimension m	
	18	36
SRRIT	*	*
EB12	284/41857	292/34268
ARNCHEB	*	*
ARPACK	61/991	80/1095
EB13	727/20004	436/7263

Note:

- * denotes that code did not converge within $4000m$ matrix vector products.
- ARNCHEB used a degree $m - r$ Chebyshev polynomial.
- ARPACK, EB13 needed a smaller convergence tolerance to achieve the same accuracy as the subspace codes.

Other findings for 2-D convection diffusion matrices.

- Results for $n = 10,000$ and $\rho = 40$ were poor (computation time long and inaccurate Ritz values returned).
- Results for $n = 1024, 2500$ were similar **except** that, for these n, SRRIT did converge and found all multiplicities. Suggests that SRRIT is slow but sure!

General findings ... subspace iteration

- SRRIT can be much slower than EB12 and LOPSI.
- SRRIT uses small number of iterations but many reorthogonalisations.
- SRRIT displays monotonic consistency.
- LOPSI can be fastest for $r = 1$ but can return spurious eigenvalues for $r > 1$. **BUG?**
- SRRIT recognises clusters... may compute more eigenvalues than requested.
- Chebychev acceleration can improve EB12.
- **But** Chebychev acceleration can give worse results.
- Codes very slow for most difficult test problems.

General findings ... Arnoldi

- ARPACK general fastest and most dependable, especially for small tolerances.
- ARPACK displays monotonic consistency.
- EB13 offers many options, enabling user to experiment.
- ARNCHEB gave reasonable results for computing a single eigenpair.
  NOTE: The Wielandt deflation strategy used does not return Ritz vectors of **A**.
- On some examples, ARPACK uses dramatically less matrix-vector products but its restarting strategy is more expensive.
- EB13's blocking option needs further work.

General findings

- None of the codes succeeded on **all** the test problems.
- All the codes are sensitive to the choice of input parameters.
- Generally, the best Arnoldi result was better than the best subspace result.
- There is a tendency of the Arnoldi methods to miss multiplicities.

Next generation software

Particular features that should be included:

- Grouping of clustered eigenvalues (cf. `SRRIT`).
- Detecting stagnating convergence (cf. `EB12, EB13`).
- Reverse communication (for Fortran codes) (cf. `EB12, EB13, ARPACK`).
- Improved polynomial restarting (eg exploiting past information) (cf. `EB12, EB13`).
- Improved stopping criteria.
- Improved reliability ... verification procedures (Meerbergen).
- Automated parameter selection (eg. subspace dimension).
- Modern programming language.
- Implicitly restarted Arnoldi appears a promising way forward.

In conclusion...

Library quality implementations of ABLE and Jacobi–Davidson are needed!
(C or FORTRAN)

- All test results and test procedures will be made available.
- A report is in preparation and will be submitted for publication.

Acknowledgments

Thanks to Zhaojun Bai, Thierry Braconnier, David Day, Iain Duff, and Danny Sorensen.