SLEPc

Computing Smallest and Interior Eigenvalues in SLEPc

Jose E. Roman

Universidad Politécnica de Valencia, Spain

IWASEP 7

UNIVERSIDAD POLITECNICA DE VALENCIA

grycap
Grupo de Redes y Computación de Altas Prestaciones

Outline

Overview of SLEPc

Eigenvalue Problems

Standard Eigenproblem

$$Ax = \lambda x$$

Generalized Eigenproblem

$$Ax = \lambda Bx$$

where

- λ is a (complex) scalar: *eigenvalue*
- x is a (complex) vector: *eigenvector*
- Matrices A and B are large and sparse
- Matrices A and B can be real or complex
- Matrices A and B can be symmetric (Hermitian) or not

Solution of the Eigenvalue Problem

There are n eigenvalues (counted with their multiplicities)

Partial eigensolution: nev solutions

$$\lambda_0, \lambda_1, \ldots, \lambda_{nev-1} \in \mathbb{C}$$
$$x_0, x_1, \ldots, x_{nev-1} \in \mathbb{C}^n$$

nev = number of eigenvalues / eigenvectors (eigenpairs)

Solution of the Eigenvalue Problem

There are n eigenvalues (counted with their multiplicities)

Partial eigensolution: nev solutions

$$\lambda_0, \lambda_1, \ldots, \lambda_{nev-1} \in \mathbb{C}$$
$$x_0, x_1, \ldots, x_{nev-1} \in \mathbb{C}^n$$

nev = number of eigenvalues / eigenvectors (eigenpairs)

Different requirements:

- Compute a few λ_i's with largest or smallest magnitude
- Compute a few rightmost or leftmost λ_i's
- Compute λ_i's closest to a given point in the complex plane
- Compute all λ_i's in a given interval

Spectral Transformation

A general technique that can be used in many methods

$$Ax = \lambda x \quad \Longrightarrow \quad Tx = \theta x$$

Spectral Transformation

A general technique that can be used in many methods

$$Ax = \lambda x \quad \Longrightarrow \quad Tx = \theta x$$

In the transformed problem

- The eigenvectors are not altered
- The eigenvalues are mapped to a different position
- Convergence is usually improved (better separation)

Spectral Transformation

A general technique that can be used in many methods

$$Ax = \lambda x \quad \Longrightarrow \quad Tx = \theta x$$

In the transformed problem

- The eigenvectors are not altered
- The eigenvalues are mapped to a different position
- Convergence is usually improved (better separation)

Shift of Origin

$$T_S = A + \sigma I$$

Shift-and-invert

$$T_{SI} = (A - \sigma I)^{-1}$$

Cayley

$$T_C = (A - \sigma I)^{-1}(A - \tau I)$$

Drawback: T not computed explicitly, linear solves instead

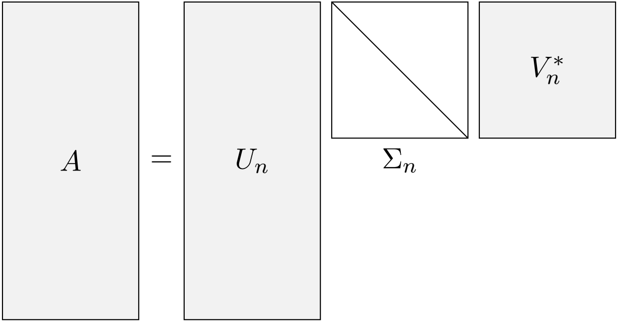
Singular Value Decomposition (SVD)

$$A = U\Sigma V^*$$

where

- A is an $m \times n$ rectangular matrix
- $U = [u_1, u_2, \ldots, u_m]$ is a $m \times m$ unitary matrix
- $V = [v_1, v_2, \ldots, v_n]$ is a $n \times n$ unitary matrix
- Σ is a $m \times n$ diagonal matrix with entries $\Sigma_{ii} = \sigma_i$
- $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$
- If A is real, U and V are real and orthogonal
- Each (σ_i, u_i, v_i) is called a singular triplet

Thin SVD



In SLEPc we compute a *partial* SVD, that is, only a subset of the singular triplets

What Users Need

- Abstraction of mathematical objects: vectors and matrices
- Efficient linear solvers (direct or iterative), preconditioners
- Easy programming interface
- Run-time flexibility, full control over the solution process
- Parallel computing, mostly transparent to the user

- State-of-the-art eigensolvers and SVD solvers
- Spectral transformations

What Users Need

Provided by PETSc

- Abstraction of mathematical objects: vectors and matrices
- Efficient linear solvers (direct or iterative), preconditioners
- Easy programming interface
- Run-time flexibility, full control over the solution process
- Parallel computing, mostly transparent to the user

Provided by SLEPc

- State-of-the-art eigensolvers and SVD solvers
- Spectral transformations

PETSc/SLEPc Numerical Components

PETSc

Nonlinear Systems		
Line Search	Trust Region	Other

Time Steppers			
Euler	Backward Euler	Pseudo Time Step	Other

Krylov Subspace Methods							
GMRES	CG	CGS	Bi-CGStab	TFQMR	Richardson	Chebychev	Other

Preconditioners						
Additive Schwarz	Block Jacobi	Jacobi	ILU	ICC	LU	Other

Matrices				
Compressed Sparse Row	Block Compressed Sparse Row	Block Diagonal	Dense	Other

Vectors

Index Sets			
Indices	Block Indices	Stride	Other

PETSc/SLEPc Numerical Components

PETSc

Nonlinear Systems

Line Search	Trust Region	Other

Time Steppers

Euler	Backward Euler	Pseudo Time Step	Other

Krylov Subspace Methods

GMRES	CG	CGS	Bi-CGStab	TFQMR	Richardson	Chebychev	Other

Preconditioners

Additive Schwarz	Block Jacobi	Jacobi	ILU	ICC	LU	Other

Matrices

Compressed Sparse Row	Block Compressed Sparse Row	Block Diagonal	Dense	Other

Vectors

Index Sets

Indices	Block Indices	Stride	Other

SLEPc

SVD Solvers

Cross Product	Cyclic Matrix	Lanczos	Thick Res. Lanczos

Eigensolvers

Krylov-Schur	Arnoldi	Lanczos	Other

Spectral Transform

Shift	Shift-and-invert	Cayley	Fold

Summary

SLEPc: Scalable Library for Eigenvalue Problem Computations

A *general* library for solving large-scale sparse eigenproblems on parallel computers

- For standard and generalized eigenproblems
- For real and complex arithmetic
- For Hermitian or non-Hermitian problems

Current version: 2.3.3 (released June 2007)

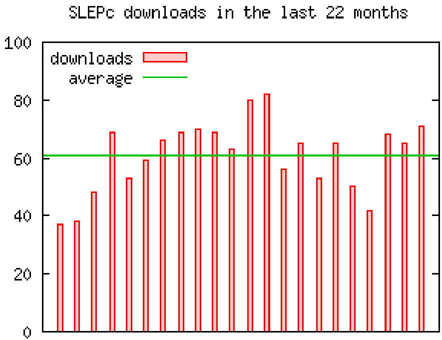
`http://www.grycap.upv.es/slepc`

The SLEPc Project

Project started in 2001

8 releases since 2002

More than 2,000 downloads (average 60 downloads per month)



SLEPc Applications

Nuclear Engineering

Computing the Lambda Modes of a Nuclear Reactor with SLEPc and PETSc, V. Hernandez et al., Intl. Conference on Computational Methods for Mathematical Science and Engineering, 183-192, Alicante (Spain), September 2002.

Simulating control rod and fuel assembly motion using moving meshes, D. Gilbert et al., Annals of Nuclear Energy, 35(2), 291-303, 2008.

Computational Electomagnetics

Computation of the Smallest Positive Eigenfrequencies of Resonant Cavities with Freely Available Parallel Solvers, V. Hernandez et al., 7th Intl. Workshop on Finite Elements for Microwave Engineering, Madrid (Spain), May 2004.

Plasma Physics

Equilibrium and Stability of Tokamak Plasmas with Arbitrary Flow, L. Guazzotto, PhD thesis.

Exceptional points in linear gyrokinetics, M. Kammerer et al., Physics of Plasmas 15, 2008.

Computational Physics, Materials Science, Electronic Structure

Rapidly Rotating Boson Molecules with Long or Short Range Repulsion: an Exact Diagonalization Study, L. O. Baksmaty et al., Phys. Rev. A 75, 2007.

Sparse Grids for the Schrödinger Equation, M. Griebel and J. Hamaekers, Math. Model. and Numer. Anal. 41(2), 215-247, 2007.

TiberCAD: A New Multiscale Simulator for Electronic and Optoelectronic Devices, M. Auf der Maur et al., Superlattices and Microstructures 41, 381-385, 2007.

Spectrum of the non-abelian phase in Kitaev's honeycomb lattice model, V. Lahtinen et al., Annals of Physics, 2008.

A stabilized stochastic finite element second-order projection method for modeling natural convection in random porous media, X. Ma and N. Zabaras, preprint.

The Computation of Resonances in Open Systems Using a Perfectly Matched Layer, S. Kim and J. E. Pasciak, preprint.

Computational Chemistry

Advanced Software for the Calculation of Thermochemistry, Kinetics, and Dynamics, D. M. Medvedev et al., J. Phys.: Conf. Ser. 16, 247-251, 2005.

Other Applications

Hessian-based model reduction for large-scale systems with initial-condition inputs, O. Bashir et al., Int. J. Numer. Meth. Engrg. 73(6), 844-868, 2007.

Spectrum of a non-self-adjoint operator associated with the periodic heat equation, M. Chugunova et al., J. Math. Anal. Appl. 342(2), 970-988, 2008.

Simple Example

```
EPS          eps;      /* eigensolver context  */
Mat          A, B;     /* matrices of Ax=kBx   */
Vec          xr, xi;   /* eigenvector, x       */
PetscScalar kr, ki;    /* eigenvalue, k        */

EPSCreate(PETSC_COMM_WORLD, &eps);
EPSSetOperators(eps, A, B);
EPSSetProblemType(eps, EPS_GNHEP);
EPSSetFromOptions(eps);

EPSSolve(eps);

EPSGetConverged(eps, &nconv);
for (i=0; i<nconv; i++) {
  EPSGetEigenpair(eps, i, &kr, &ki, xr, xi);
}

EPSDestroy(eps);
```

Available Eigensolvers

Currently available eigensolvers:

- Power Iteration and RQI
- Subspace Iteration with Rayleigh-Ritz projection and locking
- Arnoldi method with explicit restart and deflation
- Lanczos method with explicit restart and deflation
  - Reorthogonalization: Local, Partial, Periodic, Selective, Full
- Krylov-Schur (default)

Also interfaces to external software: ARPACK, PRIMME, BLOPEX

Run-Time Examples

```
% program -eps_type krylovschur -eps_nev 6 -eps_ncv 24

% program -eps_type krylovschur -eps_largest_real

% program -eps_type arnoldi -eps_tol 1e-8 -eps_max_it 2000

% program -eps_harmonic -eps_target 1.0

% program -eps_type arpack

% program -eps_type lapack

% program -eps_view -eps_monitor
```

Viewing Current Options

Sample output of -eps_view

```
EPS Object:
  problem type: symmetric eigenvalue problem
  method: lanczos
  reorthogonalization: selective
  selected portion of spectrum: largest eigenvalues in magnitude
  number of eigenvalues (nev): 1
  number of column vectors (ncv): 16
  maximum number of iterations: 100
  tolerance: 1e-07
  orthogonalization method: classical Gram-Schmidt
  orthogonalization refinement: if needed (eta: 0.500000)
  dimension of user-provided deflation space: 0
  ST Object:
    type: shift
    shift: 0
```

Built-in Support Tools

- Plotting computed eigenvalues

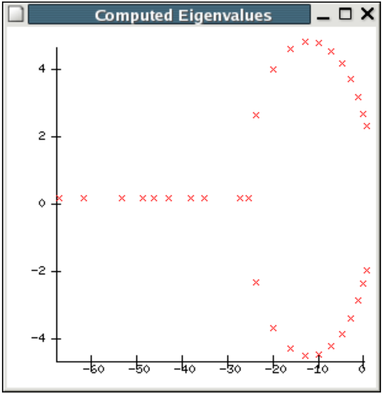
  `% program -eps_plot_eigs`

- Printing profiling information

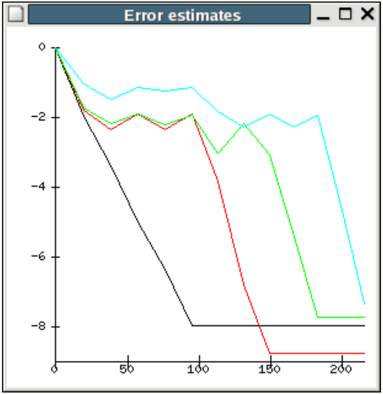
  `% program -log_summary`

- Debugging

  `% program -start_in_debugger`
  `% program -malloc_dump`



Built-in Support Tools



- Monitoring convergence (textually)

  ```
  % program -eps_monitor
  ```

- Monitoring convergence (graphically)

  ```
  % program -draw_pause 1
        -eps_monitor_draw
  ```

Simple SVD Example

```
SVD        svd;      /*  singular value solver context  */
Mat        A;        /*  matrix                          */
Vec        u, v;     /*  singular vectors                */
PetscReal  sigma;    /*  singular value                  */

SVDCreate(PETSC_COMM_WORLD, &svd);
SVDSetOperators(svd, A);
SVDSetFromOptions(svd);

SVDSolve(svd);

SVDGetConverged(svd, &nconv);
for (i=0; i<nconv; i++) {
  SVDGetSingularTriplet(svd, i, &sigma, u, v);
}

SVDDestroy(svd);
```

SVD Solvers

1. Solvers based on EPS:

Cross product matrix

$$A^*Ax = \lambda x, \quad AA^*y = \lambda y$$

Eigenvalues are $\lambda_i = \sigma_i^2$ and eigenvectors $x_i = v_i$ or $y_i = u_i$

Cyclic matrix

$$H(A)x = \lambda x, \quad H(A) = \begin{bmatrix} 0 & A \\ A^* & 0 \end{bmatrix}$$

Eigenvalues are $\pm\sigma_i$ and eigenvectors $\frac{1}{\sqrt{2}} \begin{bmatrix} \pm u_i \\ v_i \end{bmatrix}$

SVD Solvers

1. Solvers based on EPS:

Cross product matrix

$$A^*Ax = \lambda x, \quad AA^*y = \lambda y$$

Eigenvalues are $\lambda_i = \sigma_i^2$ and eigenvectors $x_i = v_i$ or $y_i = u_i$

Cyclic matrix

$$H(A)x = \lambda x, \quad H(A) = \begin{bmatrix} 0 & A \\ A^* & 0 \end{bmatrix}$$

Eigenvalues are $\pm\sigma_i$ and eigenvectors $\frac{1}{\sqrt{2}} \begin{bmatrix} \pm u_i \\ v_i \end{bmatrix}$

2. Specific solvers:
   - Explicit restart Lanczos bidiagonalization
   - Thick-restart Lanczos bidiagonalization

Run-Time Examples

```
% program -svd_type trlanczos -svd_nsv 4

% program -svd_type cross -svd_eps_type krylovschur
          -svd_ncv 30 -svd_smallest
          -svd_monitor_draw
```

Krylov Eigensolvers

Krylov Eigensolvers

Orthogonal Rayleigh-Ritz Procedure

1. Build orthogonal basis of a subspace, $V_m^* V_m = I$
2. Compute the projection onto the subspace, $H_m = V_m^* A V_m$
3. Compute eigenpairs of reduced problem, $H_m y_i = \tilde{\lambda}_i y_i$

Obtains $m \ll n$ Ritz pairs, $(\tilde{\lambda}_i, V_m y_i)$

In Krylov eigensolvers, V_m is a basis of the Krylov subspace
$\mathcal{K}_m(A, v_1) \equiv \text{span}\{v_1, Av_1, \ldots, A^{m-1}v_1\}$

Best convergence for extreme and well separated eigenvalues

Arnoldi Method

initial vector v_1 of norm 1
for $j = 1, 2, \ldots, m$
    $w = Av_j$
    for $i = 1, 2, \ldots, j$
        $h_{i,j} = v_i^* w$
        $w = w - h_{i,j} v_i$
    end
    $h_{j+1,j} = \|w\|_2$
    $v_{j+1} = w / h_{j+1,j}$
end

Orthogonalization via CGS with selective refinement and estimated norm [Hernandez, R., Tomas, 2007]

Arnoldi Decomposition

$$AV_m = V_m H_m + f e_m^*$$

Arnoldi Decomposition

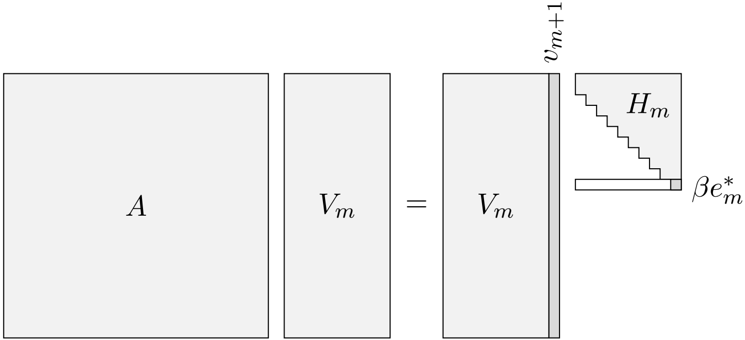
$$AV_m = V_m H_m + f e_m^*$$

$$AV_m = \left[\begin{array}{cc} V_m & v_{m+1} \end{array} \right] \left[\begin{array}{c} H_m \\ \beta e_m^* \end{array} \right]$$

Arnoldi Decomposition

$$AV_m = V_m H_m + f e_m^*$$

$$AV_m = \left[\begin{array}{cc} V_m & v_{m+1} \end{array} \right] \left[\begin{array}{c} H_m \\ \beta e_m^* \end{array} \right]$$



Restarted Arnoldi Methods

Cost grows with m - Restarting is required

1. Build an initial Arnoldi decomposition
2. Extract spectral information
3. Use information to build a new Arnoldi decomposition
4. If not satisfied go to step 2

Restarted Arnoldi Methods

Cost grows with m - Restarting is required

1. Build an initial Arnoldi decomposition
2. Extract spectral information
3. Use information to build a new Arnoldi decomposition
4. If not satisfied go to step 2

Alternatives:

Explicit restart: build new decomposition from a single vector

Implicit restart: compact the decomposition to dimension $m - p$, then extend it again (implemented in ARPACK)

The Krylov-Schur method [Stewart, 2001, 2002] is an alternative to the implicit restart implemented in ARPACK

Krylov Decomposition

Krylov decomposition

$$AU = UB + ub^*$$

Generalization of Arnoldi decomposition, B is not upper Hessenberg

Similarity transformation:

$$A(UW^{-1}) = (UW^{-1})(WBW^{-1}) + u(b^*W^{-1})$$

Translation:

$$AU = U(B + gb^*) + (u - Ug)b^*$$

Krylov-Schur Decomposition

B can be reduced to upper quasi-triangular form by orthogonal similarity transformations

Krylov-Schur decomposition

$$AU = US + ub^*$$

Krylov-Schur Decomposition

B can be reduced to upper quasi-triangular form by orthogonal similarity transformations

Krylov-Schur decomposition

$$AU = US + ub^*$$

A Krylov-Schur decomposition can be truncated at any point

$$A \begin{bmatrix} U_1 & U_2 \end{bmatrix} = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{bmatrix} + u \begin{bmatrix} b_1^* & b_2^* \end{bmatrix}$$

$$AU_1 = U_1 S_{11} + ub_1^*$$

Krylov-Schur Method

Input: Matrix A, initial vector x_1, and number of steps m
Output: $k \leq p$ Ritz pairs

1. Build an orthonormal Krylov decomposition of order m
2. Reduce to Krylov-Schur form by orthogonal similarity
3. Sort diagonal blocks of the Rayleigh quotient
4. Truncate to a Krylov-Schur decomposition of order p
5. Extend to a Krylov decomposition of order m
6. If not satisfied, go to step 2

Krylov-Schur Method

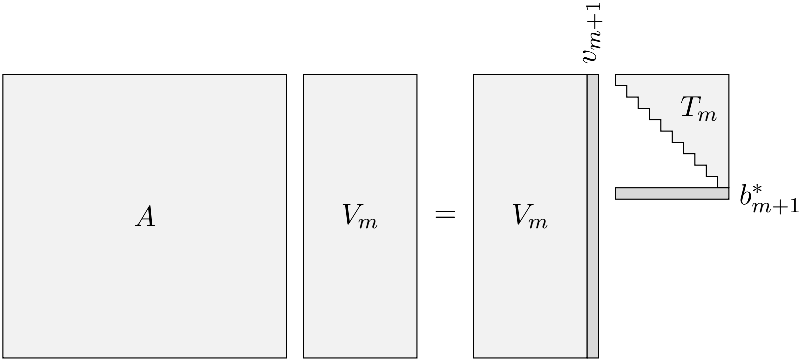
Input: Matrix A, initial vector x_1, and number of steps m
Output: $k \leq p$ Ritz pairs

1. Build an orthonormal Krylov decomposition of order m
2. Reduce to Krylov-Schur form by orthogonal similarity
3. Sort diagonal blocks of the Rayleigh quotient
4. Truncate to a Krylov-Schur decomposition of order p
5. Extend to a Krylov decomposition of order m
6. If not satisfied, go to step 2

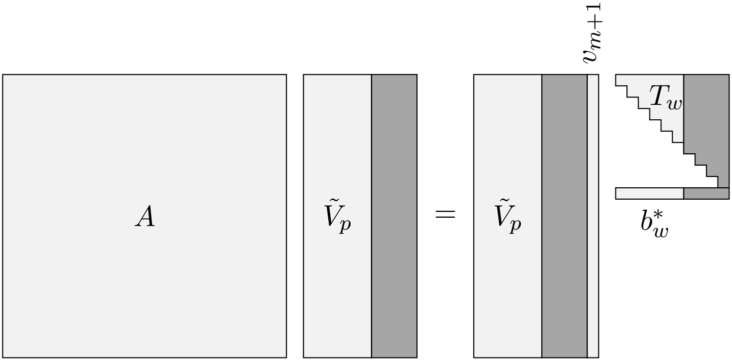
Before step 5, lock converged eigenpairs according to residual norm

$$\|A\hat{U}y - \theta\hat{U}y\| = \|(\hat{U}\hat{S} + u\hat{b}^*)y - \theta\hat{U}y\| = \|u\hat{b}^*y\| = |\hat{b}^*y|$$

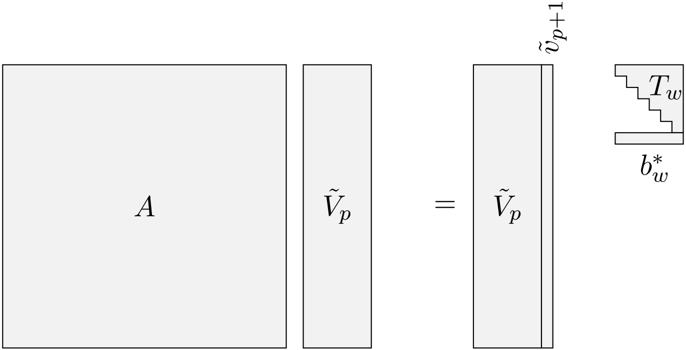
Build Krylov-Schur Decomposition



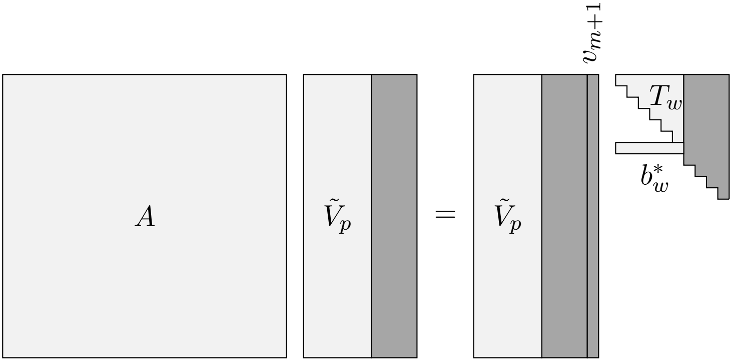
Sort and Select p Wanted Eigenvalues



Truncate the Decomposition

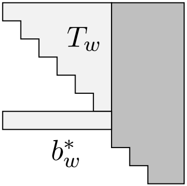


Extend the Decomposition with Arnoldi Steps



Structure of Projected Eigenproblem

The projected eigenproblem has no longer Hessenberg form



Structure of Projected Eigenproblem

The projected eigenproblem has no longer Hessenberg form

In symmetric problems the projected matrix is also symmetric - Thick-restart Lanczos method

Spectral Transformation

Spectral Transformation in SLEPc

An ST object is always associated to any EPS object

$$Ax = \lambda x \quad \Longrightarrow \quad Tx = \theta x$$

Spectral Transformation in SLEPc

An ST object is always associated to any EPS object

$$Ax = \lambda x \quad \Longrightarrow \quad Tx = \theta x$$

- The user need not manage the ST object directly
- Internally, the eigensolver works with the operator T
- At the end, eigenvalues are transformed back automatically

Spectral Transformation in SLEPc

An ST object is always associated to any EPS object

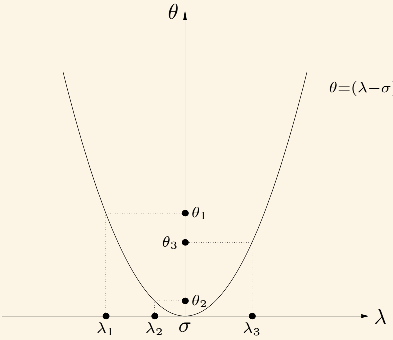
$$Ax = \lambda x \quad \Longrightarrow \quad Tx = \theta x$$

- The user need not manage the ST object directly
- Internally, the eigensolver works with the operator T
- At the end, eigenvalues are transformed back automatically

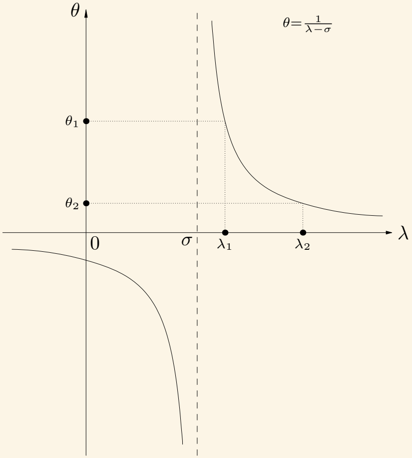
ST	Standard problem	Generalized problem
`shift`	$A + \sigma I$	$B^{-1}A + \sigma I$
`fold`	$(A + \sigma I)^2$	$(B^{-1}A + \sigma I)^2$
`sinvert`	$(A - \sigma I)^{-1}$	$(A - \sigma B)^{-1}B$
`cayley`	$(A - \sigma I)^{-1}(A - \tau I)$	$(A - \sigma B)^{-1}(A - \tau B)$

Illustration of Spectral Transformation

Spectrum folding



Shift-and-invert



Shift-and-Invert

$$T_{SI} = (A - \sigma I)^{-1}$$

Enhance convergence of eigenvalues closest to σ [Ericsson, Ruhe, 1980], [Nour-Omid et al, 1987], [Grimes et al, 1994]

Shift-and-Invert

$$T_{SI} = (A - \sigma I)^{-1}$$

Enhance convergence of eigenvalues closest to σ [Ericsson, Ruhe, 1980], [Nour-Omid et al, 1987], [Grimes et al, 1994]

Problem: convergence criterion operates on transformed problem

- Improve x [Ericsson, Ruhe, 1980], [Hetmaniuk, Lehoucq, 2006]
- Compute explicit residuals of original problem

Shift-and-Invert

$$T_{SI} = (A - \sigma I)^{-1}$$

Enhance convergence of eigenvalues closest to σ [Ericsson, Ruhe, 1980], [Nour-Omid et al, 1987], [Grimes et al, 1994]

Problem: convergence criterion operates on transformed problem

- Improve x [Ericsson, Ruhe, 1980], [Hetmaniuk, Lehoucq, 2006]
- Compute explicit residuals of original problem

Linear systems must be solved in each Arnoldi iteration

- Direct solver recommended: exact shift-and-invert
- Iterative solver possible: inexact shift-and-invert

Far eigenvalues mapped to zero: possible bad convergence

Cayley Transform

$$T_C = (A - \sigma I)^{-1}(A - \tau I)$$

Mathematically equivalent to shift-and-invert, $T_C = I + (\sigma - \tau)T_{SI}$

Also enhance convergence of eigenvalues closest to σ

Linear solver also required

Far eigenvalues mapped to one: better for iterative solvers

Has been used for applications that compute rightmost eigenvalues [Meerbergen, Roose, 1996, 1997], [Lehoucq, Salinger, 2001]

ST Run-Time Examples

```
% program -eps_type power -st_type shift -st_shift 1.5

% program -eps_type power -st_type sinvert -st_shift 1.5

% program -eps_type power -st_type sinvert
          -eps_power_shift_type rayleigh

% program -st_type sinvert   -st_shift 1
          -st_ksp_type bcgsl -st_ksp_rtol 1e-8
          -st_pc_type sor    -st_pc_sor_omega 1.3

% program -st_type cayley -st_shift 1 -st_antishift -1
```

Harmonic Projection

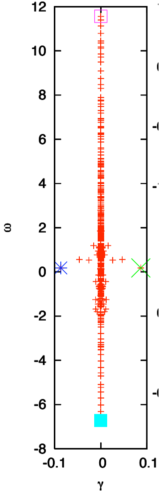
Application: Plasma Turbulence

Study of plasma turbulence inside a tokamak

- Nonlinear gyrokinetic equations

SLEPc is used to compute the rightmost eigenvalues of a complex, non-Hermitian matrix

Sizes ranging from a few millions to even a billion



Application: Plasma Turbulence

Study of plasma turbulence inside a tokamak

- Nonlinear gyrokinetic equations

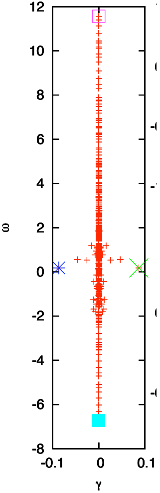
SLEPc is used to compute the rightmost eigenvalues of a complex, non-Hermitian matrix

Sizes ranging from a few millions to even a billion

The matrix is not built explicitly

- Spectral transformation impractical (no preconditioner available)

Harmonic Krylov-Schur is about 5 times faster than inexact spectral transform



Harmonic Projection

Harmonic Ritz values [Morgan, 1991], [Paige, Parlett, van der Vorst, 1995] are the reciprocals of Ritz values of A^{-1} with respect to the subspace $A\mathcal{K}_m(A, v_1)$

- The idea is to compute them without building A^{-1}
- Harmonic Ritz values are not shift-invariant $\rightarrow (A - \tau I)^{-1}$

Can also be defined in terms of Rayleigh quotients [Beattie, 1998]

Another interpretation possible in terms of an oblique projection

- The space of approximats is $\mathcal{K}_m(A, v_1)$
- The residuals are orthogonal to $A\mathcal{K}_m(A, v_1)$

Harmonic Projection - Practical Implementation

In a practical implementation, we want to

- Avoid building $(A - \tau I)^{-1}$
- Avoid building two bases, one for $\mathcal{K}_m$ and the other for $A\mathcal{K}_m$

Main idea:

- Define $V := (A - \tau I)U$, where U is a (orthogonal) basis of $\mathcal{K}_m$
- Manipulate the relations, trying to cancel $(A - \tau I)^{-1}$

Harmonic Krylov-Schur - Naive Approach

Suppose we have built a Krylov decomposition $AU = UB + ub^*$

After shifting

$$(A - \tau I)U = U(B - \tau I) + ub^*$$

Since $V := (A - \tau I)U$, we have $V = U(B - \tau I) + ub^*$, or

$$\boxed{U = V(B - \tau I)^{-1} - ug^*} \qquad g := (B - \tau I)^{-*}b$$

Harmonic Krylov-Schur - Naive Approach

Suppose we have built a Krylov decomposition $AU = UB + ub^*$

After shifting

$$(A - \tau I)U = U(B - \tau I) + ub^*$$

Since $V := (A - \tau I)U$, we have $V = U(B - \tau I) + ub^*$, or

$$\boxed{U = V(B - \tau I)^{-1} - ug^*} \qquad g := (B - \tau I)^{-*}b$$

This is a Krylov decomposition of $(A - \tau I)^{-1}$, so if θ is an eigenvalue of $(B - \tau I)^{-1}$ then $\theta^{-1} + \tau$ is an harmonic Ritz value of A

Problem: this would require computing V and orthogonalizing

Harmonic Krylov-Schur - Galerkin View

Orthogonal projection for $(A-\tau I)^{-1}$ with respect to V

Galerkin condition
$$V^*\left[(A-\tau I)^{-1}Vz-\tilde{\theta}Vz\right]=0$$
$$V^*(A-\tau I)^{-1}Vz=\tilde{\theta}V^*Vz$$

Harmonic Krylov-Schur - Galerkin View

Orthogonal projection for $(A-\tau I)^{-1}$ with respect to V

Galerkin condition
$$V^*\left[(A-\tau I)^{-1}Vz-\tilde{\theta}Vz\right]=0$$
$$V^*(A-\tau I)^{-1}Vz=\tilde{\theta}V^*Vz$$

In terms of U:

$$U^*(A-\tau I)^*Uz=\tilde{\theta}U^*(A-\tau I)^*(A-\tau I)Uz$$

Using $(A-\tau I)U=U(B-\tau I)+ub^*$:

$$(B-\tau I)^*z=\tilde{\theta}\left[U(B-\tau I)+ub^*\right]^*\left[U(B-\tau I)+ub^*\right]z$$

$$\boxed{(B-\tau I)^*z=\tilde{\theta}\left[(B-\tau I)^*(B-\tau I)+bb^*\right]z}$$

Harmonic Krylov-Schur - Projected Problem

The projected problem

$$(B-\tau I)^* z = \tilde{\theta}\left[(B-\tau I)^*(B-\tau I) + bb^*\right] z$$

can be solved as a generalized eigenproblem or reduced to a standard eigenproblem:

1. Compute Cholesky factor of right matrix, as $QR = \begin{bmatrix} B-\tau I \\ b^* \end{bmatrix}$
2. Pre-multiply by $(B-\tau I)^{-*}$ (if well conditioned)

$$z = \tilde{\theta}\left[(B-\tau I) + gb^*\right] z$$

$$(B + gb^*)z = (\tilde{\theta}^{-1} + \tau)z$$

$$g := (B-\tau I)^{-*} b$$

Harmonic Krylov-Schur - Petrov-Galerkin View

Oblique projection for $(A - \tau I)$ onto $\mathcal{K}_m$ orthogonal to $A\mathcal{K}_m$

Petrov-Galerkin condition

$$V^* \left[(A - \tau I)Uy - \tilde{\lambda} Uy \right] = 0$$

$$V^*(A - \tau I)Uy = \tilde{\lambda} V^* Uy$$

Harmonic Krylov-Schur - Petrov-Galerkin View

Oblique projection for $(A - \tau I)$ onto $\mathcal{K}_m$ orthogonal to $A\mathcal{K}_m$

Petrov-Galerkin condition
$$V^* \left[(A - \tau I)Uy - \tilde{\lambda} U y \right] = 0$$
$$V^*(A - \tau I)Uy = \tilde{\lambda} V^* U y$$

Using the Krylov decomposition:

$$V^* \left[U(B - \tau I) + ub^* \right] y = \tilde{\lambda} V^* U y$$
$$\left[V^* U(B - \tau I) + V^* ub^* \right] y = \tilde{\lambda} V^* U y$$
$$\left[B - \tau I + (V^* U)^{-1} V^* ub^* \right] y = \tilde{\lambda} y$$

$V^*U = U^*(A - \tau I)^* U = (B - \tau I)^*$ and $V^*u = U^*(A - \tau I)^* u = b$

$$\boxed{\left[B + (B - \tau I)^{-*} bb^* \right] y = (\tilde{\lambda} + \tau) y}$$

Harmonic Krylov-Schur - Summary

1. Build a Krylov-Schur decomposition for A

$$AU = UB + ub^*$$

2. Solve projected eigenproblem

$$(B + gb^*)z = (\tilde{\theta}^{-1} + \tau)z$$

3. The eigenvalues are harmonic Ritz values

4. The harmonic Ritz vectors are Vz but Uz can be taken as well

Harmonic Krylov-Schur - Summary

1. Build a Krylov-Schur decomposition for A

$$AU = UB + ub^*$$

2. Solve projected eigenproblem

$$(B + gb^*)z = (\tilde{\theta}^{-1} + \tau)z$$

3. The eigenvalues are harmonic Ritz values

4. The harmonic Ritz vectors are Vz but Uz can be taken as well

---

Practical considerations: restart and convergence test
Implementation based on translations [Stewart, 2002]

Harmonic Krylov-Schur Method

1. Build an orthonormal Krylov decomposition of order m

3. Reduce to Krylov-Schur form by orthogonal similarity
4. Sort diagonal blocks of the Rayleigh quotient
5. Truncate to a Krylov-Schur decomposition of order p

7. Extend to a Krylov decomposition of order m
8. If not satisfied, go to step 2

SLEPc

Harmonic Krylov-Schur Method

1. Build an orthonormal Krylov decomposition of order m
2. Translate

$$AU = U\tilde{B} + \tilde{u}b^*, \quad \tilde{B} = B + gb^*, \quad \tilde{u} = u - Ug$$

3. Reduce to Krylov-Schur form by orthogonal similarity
4. Sort diagonal blocks of the Rayleigh quotient
5. Truncate to a Krylov-Schur decomposition of order p

7. Extend to a Krylov decomposition of order m
8. If not satisfied, go to step 2

Harmonic Krylov-Schur Method

1. Build an orthonormal Krylov decomposition of order m
2. Translate

$$AU = U\tilde{B} + \tilde{u}b^*, \quad \tilde{B} = B + gb^*, \quad \tilde{u} = u - Ug$$

3. Reduce to Krylov-Schur form by orthogonal similarity
4. Sort diagonal blocks of the Rayleigh quotient (largest θ)
5. Truncate to a Krylov-Schur decomposition of order p

7. Extend to a Krylov decomposition of order m
8. If not satisfied, go to step 2

Harmonic Krylov-Schur Method

1. Build an orthonormal Krylov decomposition of order m
2. Translate

$$AU = U\tilde{B} + \tilde{u}b^*, \quad \tilde{B} = B + gb^*, \quad \tilde{u} = u - Ug$$

3. Reduce to Krylov-Schur form by orthogonal similarity
4. Sort diagonal blocks of the Rayleigh quotient (largest θ)
5. Truncate to a Krylov-Schur decomposition of order p
6. Recover orthonormality with another translation

$$\hat{B} = \hat{S} + \hat{g}\hat{b}^*, \quad \gamma\hat{u} = \tilde{u} - \hat{U}\hat{g}, \quad \hat{g} = -Q_{1:\ell}^* g$$

7. Extend to a Krylov decomposition of order m
8. If not satisfied, go to step 2

Harmonic Krylov-Schur Method - Practical Aspects

$\tilde{u}$ is not really necessary, because

$$\gamma\hat{u} = (u - Ug) - \hat{U}\hat{g} = u - Ug + UQ_{1:\ell}Q_{1:\ell}^{*}g = u - U\tilde{g}$$

where $\tilde{g} = (I - Q_{1:\ell}Q_{1:\ell}^{*})g$

Harmonic Krylov-Schur Method - Practical Aspects

$\tilde{u}$ is not really necessary, because

$$\gamma\hat{u} = (u - Ug) - \hat{U}\hat{g} = u - Ug + UQ_{1:\ell}Q_{1:\ell}^{*}g = u - U\tilde{g}$$

where $\tilde{g} = (I - Q_{1:\ell}Q_{1:\ell}^{*})g$

Its norm is required for the computation of the residual norm estimates,

$$\|A\hat{U}y - \theta\hat{U}y\| = \|(\hat{U}\hat{S} + \tilde{u}\hat{b}^{*})y - \theta\hat{U}y\| = \|\tilde{u}\hat{b}^{*}y\| = \|\tilde{u}\||\hat{b}^{*}y|$$

But since $\tilde{u} = u - Ug$ with $u \perp U$,

$$\|\tilde{u}\| = \sqrt{1 + \|g\|^2}$$

Harmonic Krylov-Schur Method - Compact Implementation

Input: $AU = UB + ub^*$ Output: $A\hat{U} = \hat{U}\hat{B} + \hat{u}(\gamma\hat{b})^*$

$g = (B - \tau I)^{-*}b$
$\tilde{B} = B + gb^*$
$[Q, \tilde{S}] = \texttt{schur\_sorted}(B, \tau)$
$\hat{b} = Q_{1:\ell}^* b$
$\hat{g} = -Q_{1:\ell}^* g$
$\|\tilde{u}\| = \sqrt{1 + \|g\|^2}$
Test convergence, using $\|\tilde{u}\|$ and $\hat{b}$
$\hat{B} = \tilde{S}_{1:\ell,1:\ell} + \hat{g}\hat{b}^*$
$\tilde{g} = (I - Q_{1:\ell}Q_{1:\ell}^*)g$
$\hat{u} = u - U\tilde{g}$
$\gamma = \sqrt{1 + \|\tilde{g}\|^2}$
$\hat{U} = UQ_{1:\ell}$

Refined Harmonic Extraction

For the refined harmonic extraction, we change the way the harmonic Ritz vector and the residual estimate are computed:

$$\tilde{x}_i = U z_i,$$

$$\|(A - \tilde{\lambda}_i I)\tilde{x}_i\| = \sigma_{min}(C),$$

where

$$C = \begin{bmatrix} B - \tau I \\ b^* \end{bmatrix},$$

and z_i is the right singular vector of C associated with its smallest singular value.

Conclusion

SLEPc Highlights

- Growing number of eigensolvers
- Seamlessly integrated spectral transformation
- Easy programming with PETSc's object-oriented style
- Data-structure neutral implementation
- Run-time flexibility, giving full control over the solution process
- Portability to a wide range of parallel platforms
- Usable from code written in C, C++ and Fortran
- Extensive documentation

Harmonic Krylov-Schur - Work in Progress

Usable prototype of harmonic Krylov-Schur

- Check if it is safe to invert $(B - \tau I)^*$
- Keep projected eigenproblem in generalized form to preserve symmetry
- Discard harmonic Ritz value, keep vector

Harmonic Krylov-Schur - Work in Progress

Usable prototype of harmonic Krylov-Schur

- Check if it is safe to invert $(B - \tau I)^*$
- Keep projected eigenproblem in generalized form to preserve symmetry
- Discard harmonic Ritz value, keep vector

Extensions:

- Explicitly restarted solvers (Arnoldi and Lanczos)
- Lanczos bidiagonalization for the SVD [Baglama, Reichel, 2005]
- Use in Cyclic SVD solver for small singular values
- Generalizations of harmonic Ritz values [Hochstenbach, 2005]

More Work in Progress

Refined extraction – need to know:

- How locking is done in explicitly restarted Arnoldi
- How complex conjugate pairs can be handled
- How refined Ritz vectors fit in the Krylov-Schur scheme

Extend for harmonic projection

More Work in Progress

Refined extraction – need to know:

- How locking is done in explicitly restarted Arnoldi
- How complex conjugate pairs can be handled
- How refined Ritz vectors fit in the Krylov-Schur scheme

Extend for harmonic projection

Preconditioned eigensolvers

- First attempt: Davidson-type trace minimization (to be presented in Vecpar 08)
- Next goal: Generalized Davidson and various Jacobi-Davidson

Thanks!

SLEPc

`http://www.grycap.upv.es/slepc`

`slepc-maint@grycap.upv.es`