SLEPc: overview, applications and some implementation details

Jose E. Roman

Universidad Politécnica de Valencia, Spain

ETH Zürich - May, 2010

UNIVERSIDAD POLITECNICA DE VALENCIA

Arnoldi Method

Computes a basis V_m of $\mathcal{K}_m(A, v_1)$ and $H_m = V_m^* A V_m$

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

Very elegant algorithm, BUT what else is required for addressing the needs of real applications?

Outline

1. Sample Applications
2. Overview of SLEPc
   - Eigenvalue Solvers
   - Spectral Transformation
3. Some Implementation Details of Krylov Solvers
   - Orthogonalization, restart
   - Symmetry, deflation, parallelization

Eigenproblems

Large-scale eigenvalue problems are among the most demanding calculations in scientific computing

Example application areas:

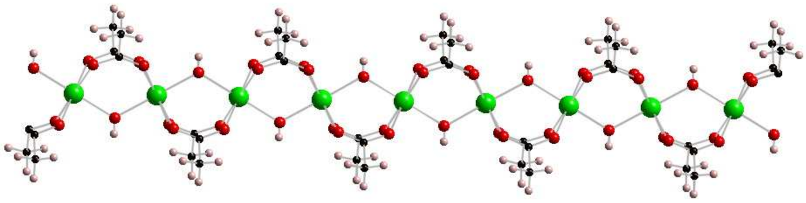
- Dynamic structural analysis (e.g. civil engineering)
- Stability analysis (e.g. control engineering)
- Eigenfunction determination (e.g. electromagnetics)
- Bifurcation analysis (e.g. fluid dynamics)
- Information retrieval (e.g. latent semantic indexing)

Application 1: Molecular Clusters

Goal: Analysis of high-nuclearity spin clusters

- Bulk magnetic properties (magnetic susceptibility and magnetization)
- Spectroscopic properties (inelastic neutron scattering spectra)

Example: Chain of Ni atoms with antiferromagnetic interaction (use closed ring to simulate infinite chain; for 10 ions n=59,049)



[Ramos, R., Cardona-Serra, Clemente-Juan, 2010] (submitted)

Application 2: FE in Schrödinger Equation

Goal: Facilitate the finite-element analysis of the Schrödinger Equation $H\Psi = \varepsilon\Psi$

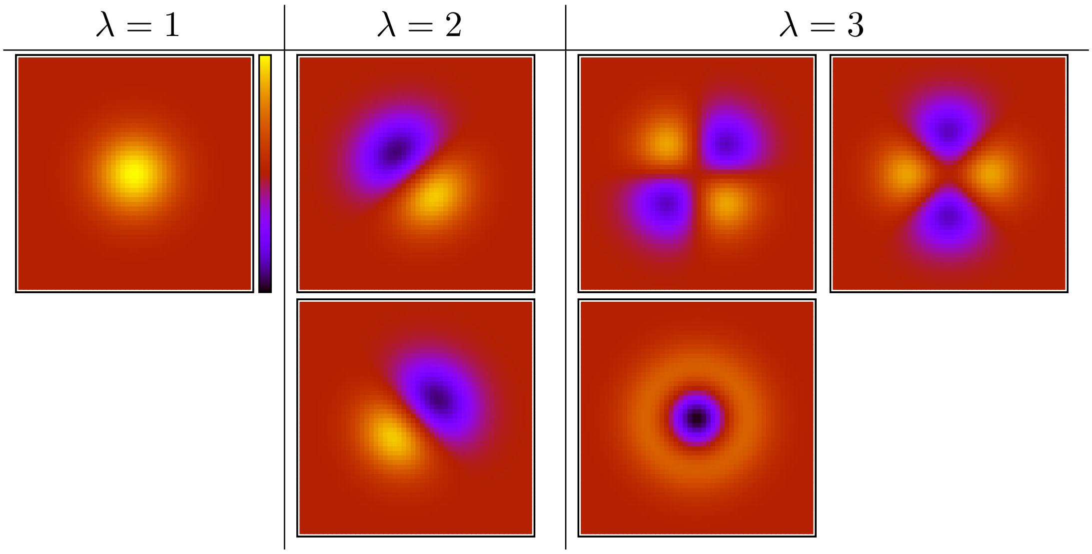
- SLEPc together with deal.II finite-element library

Features:

- Automatic mesh creation and adaptive refinement
- Computation of local matrices from a library of element types
- Automatic assembly of system matrices A, B
- Robust computation of energies/eigenstates with SLEPc $(A - \varepsilon_n B)\tilde{\psi}_n = 0$

Initial work: simple harmonic oscillator [Young, Romero, R., 2009]
Long term: Hartree-Fock self-consistent field formalism

Application 2: FE in Schrödinger Equation (cont'd)



Square domain, discretized with quadrangular Lagrange elements

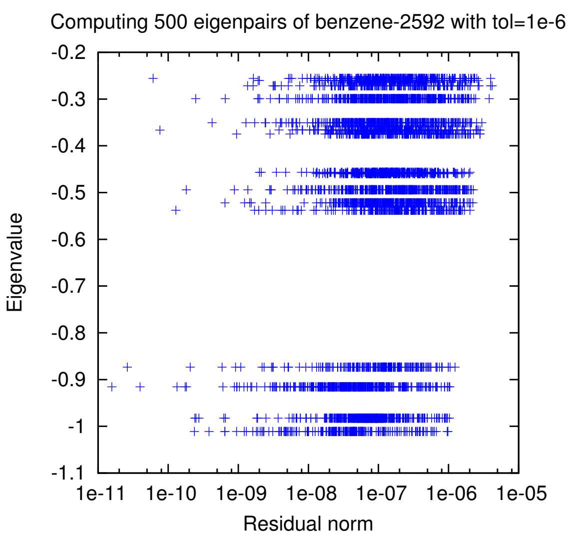
Application 3: SIESTA

SIESTA: a parallel code for self-consistent DFT calculations

Generalized symmetric-definite problem:
$(A - \varepsilon_n B)\tilde{\psi}_n = 0$

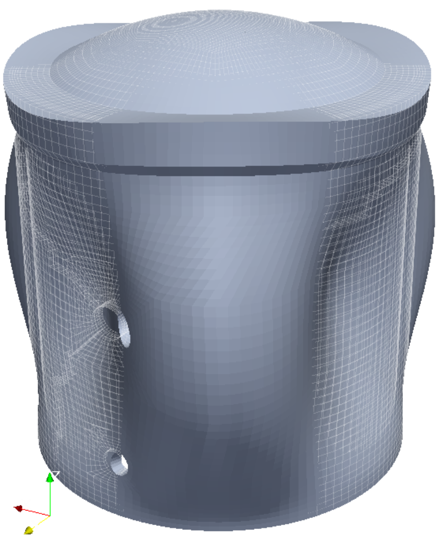
Challenges:

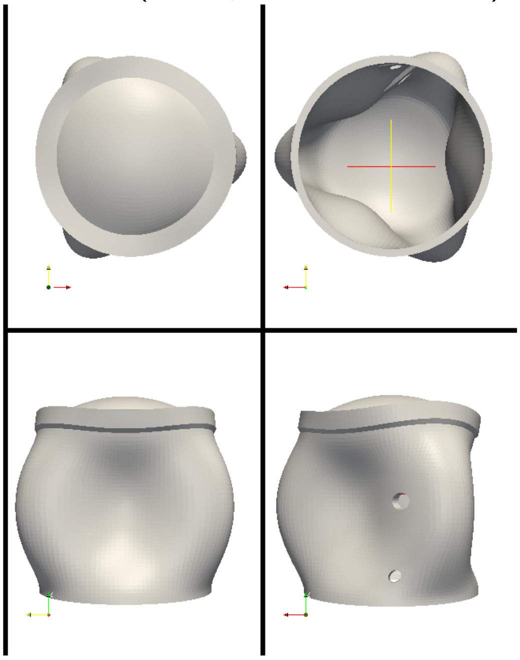
- Large number of eigenpairs
- High multiplicity



Application 4: Structural Dynamics

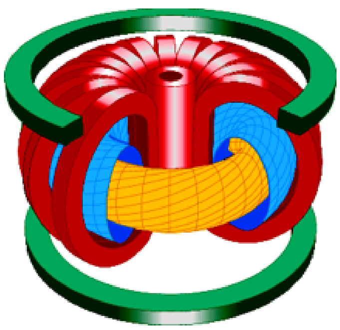
Smallest eigenfrequencies of $Kx = \lambda Mx$ (real symmetric-definite)

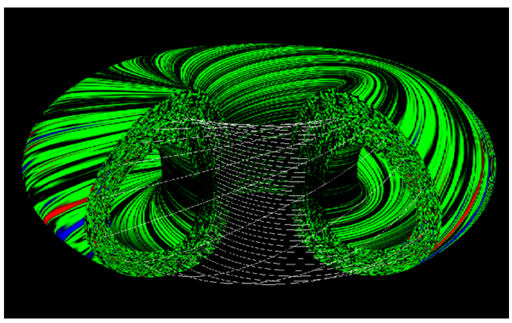




Application 5: Plasma Physics

Plasma turbulence in a tokamak determines its energy confinement





Simulation based on the nonlinear gyrokinetic equations

- GENE parallel code, scalable to 1000's of processors
- Numerical solution with initial value solver

Application 5: Plasma Physics (cont'd)

Knowledge of the operator's spectrum can be useful

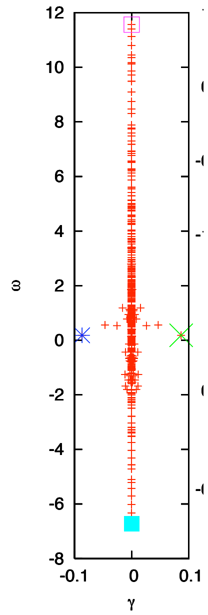
$$Ax = \lambda x$$

- Complex, non-Hermitian eigenproblem
- The matrix is not built explicitly
- Sizes ranging from a few millions to a billion

Uses:

1. Largest magnitude eigenvalue to estimate optimal timestep
2. Track sub-dominant instabilities (rightmost eigenvalues)

[R., Kammerer, Merz, Jenko, 2010]



Application 6: Nuclear Engineering

Modal analysis of nuclear reactor cores

Objectives:

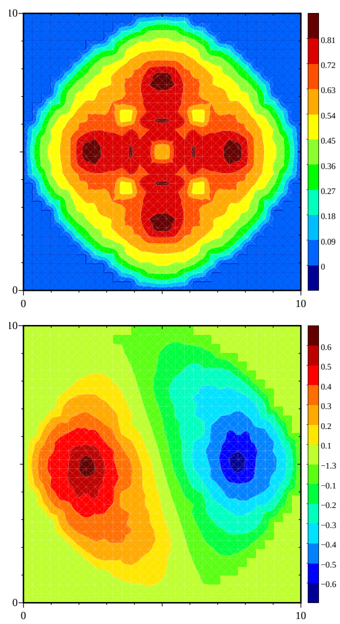
- Improve safety
- Reduce operation costs

Lambda Modes Equation

$$\mathcal{L}\phi = \frac{1}{\lambda}\mathcal{M}\phi$$

Want modes associated to largest λ

- Criticality (eigenvalues)
- Prediction of instabilities and transient analysis (eigenvectors)



Application 6: Nuclear Engineering (cont'd)

Discretized eigenproblem (two energy groups):

$$\begin{bmatrix} L_{11} & 0 \\ -L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix} = \frac{1}{\lambda} \begin{bmatrix} M_{11} & M_{12} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix}$$

Can be formulated as

$$G\psi_1 = \lambda\psi_1 \ , \quad G = L_{11}^{-1}(M_{11} + M_{12}L_{22}^{-1}L_{21})$$

- Matrix should not be computed explicitly
- In some applications, many successive problems are solved

Other modes may be of interest [Verdú, Ginestar, R., Vidal, 2010]

Application 7: Computational Electromagnetics

Analysis of resonant cavities

Source-free wave equations

$$\begin{aligned} \nabla \times (\hat{\mu}_r^{-1}\nabla \times \vec{E}) - \kappa_0^2\hat{\varepsilon}_r\vec{E} = 0 \\ \nabla \times (\hat{\varepsilon}_r^{-1}\nabla \times \vec{H}) - \kappa_0^2\hat{\mu}_r\vec{H} = 0 \end{aligned}$$

FEM discretization leads to $Ax = \kappa_0^2 Bx$
Target: smallest nonzero eigenfrequencies (large nullspace)

$$\underbrace{\lambda_1, \lambda_2, \ldots, \lambda_k}_{=0}, \underbrace{\lambda_{k+1}, \lambda_{k+2}}_{\text{Target}}, \ldots, \lambda_n$$

Eigenfunctions associated to 0 are irrotational electric fields, $\vec{E} = -\nabla\Phi$. This allows the computation of a basis of $\mathcal{N}(A)$

Constrained Eigenvalue Problem

$$\left.\begin{aligned} Ax = \kappa_0^2 Bx \\ C^T Bx = 0 \end{aligned}\right\}$$

Facts Observed from the Examples

- Various problem characteristics
  - Real/complex, Hermitian/non-Hermitian
  - Need to support complex in real arithmetic
- Many formulations
  - Standard, generalized, quadratic, non-linear, SVD
  - Special cases: implicit matrix, block-structured problems, constrained problems, structured spectrum
- Wanted solutions
  - Usually only a few eigenpairs, but may be many
  - Any part of the spectrum (exterior, interior), intervals
- Robustness and usability
  - Singular B and other special cases, high multiplicity
  - Interoperability (linear solvers, FE), flexibility

Eigenvalue Problems

Consider the following 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 can be real or complex
- Matrices A and B can be symmetric (Hermitian) or not
- Typically, B is symmetric positive (semi-) definite

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:

- A few of the dominant eigenvalues (largest magnitude)
- A few λ_i's with smallest or largest real parts
- A few λ_i's closest to a target value in the complex plane

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 modified by a simple relation
- 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

What Users Need

Provided by PETSc

- Abstraction of mathematical objects: vectors and matrices
- Efficient linear solvers (direct or iterative)
- 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 (or SVD solvers)
- Spectral transformations

Summary

PETSc: Portable, Extensible Toolkit for Scientific Computation

Software for the scalable (parallel) solution of algebraic systems arising from partial differential equation (PDE) simulations

- Developed at Argonne National Lab since 1991
- Usable from C, C++, Fortran77/90
- Focus on abstraction, portability, interoperability
- Extensive documentation and examples
- Freely available and supported through email

`http://www.mcs.anl.gov/petsc`

Current version: 3.1 (released March 2010)

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

Also support for the partial SVD decomposition

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

Current version: 3.0.0 (released Feb 2009)

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

EPS: Basic Usage

Usual steps for solving an eigenvalue problem with SLEPc:

1. Create an EPS object
2. Define the eigenvalue problem
3. (Optionally) Specify options for the solution
4. Run the eigensolver
5. Retrieve the computed solution
6. Destroy the EPS object

All these operations are done via a generic interface, common to all the eigensolvers

EPS: 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);
```

EPS: Run-Time Examples

```
% program -eps_view -eps_monitor

% program -eps_type krylovschur -eps_nev 6 -eps_ncv 24

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

% program -eps_type subspace -eps_hermitian -log_summary

% program -eps_type lapack

% program -eps_type arpack -eps_plot_eigs -draw_pause -1

% program -eps_type primme -eps_smallest_real
```

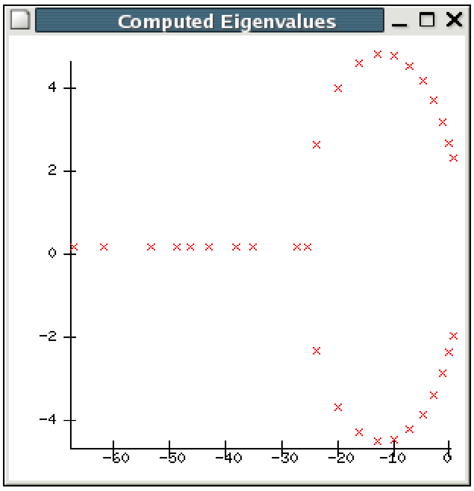

EPS: Viewing Current Options

Sample output of -eps_view

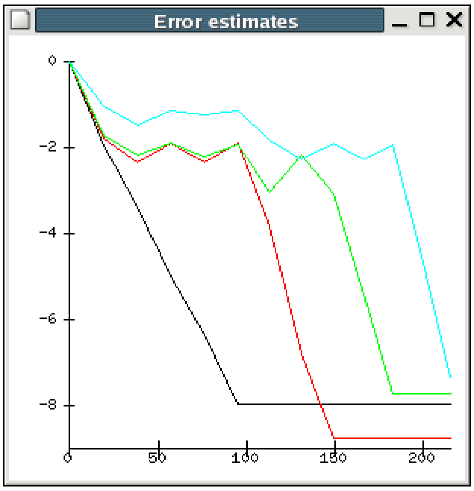
```
EPS Object:
  problem type: symmetric eigenvalue problem
  method: krylovschur
  selected portion of spectrum: largest eigenvalues in magnitude
  number of eigenvalues (nev): 1
  number of column vectors (ncv): 16
  maximum dimension of projected problem (mpd): 16
  maximum number of iterations: 100
  tolerance: 1e-07
  dimension of user-provided deflation space: 0
  IP Object:
    orthogonalization method:  classical Gram-Schmidt
    orthogonalization refinement:  if needed (eta: 0.707100)
  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`

Spectral Transformation in SLEPc

An ST object is always associated to any EPS object

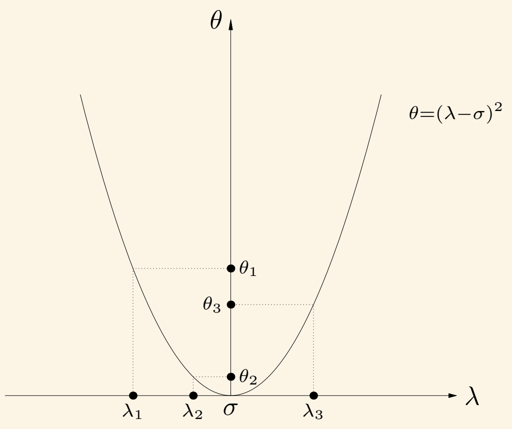
$$Ax = \lambda x \quad \Longrightarrow \quad Tx = \theta x$$

- 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

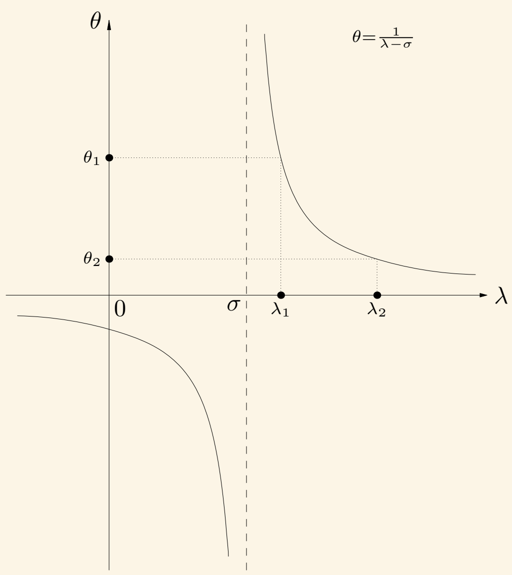
ST	Standard problem	Generalized problem
`shift`	$A + \sigma I$	$B^{-1}A + \sigma I$
`fold`	$(A + \sigma I)^2$	$(B^{-1}A + \sigma I)^2$
`sinvert`	$(A - \sigma I)^{-1}$	$(A - \sigma B)^{-1}B$
`cayley`	$(A - \sigma I)^{-1}(A + \tau I)$	$(A - \sigma B)^{-1}(A + \tau B)$

Illustration of Spectral Transformation

Spectrum folding



Shift-and-invert



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 -eps_type arpack -eps_tol 1e-6
          -st_type sinvert -st_shift 1
          -st_ksp_type cgs -st_ksp_rtol 1e-8
          -st_pc_type sor  -st_pc_sor_omega 1.3
```


SLEPc Technical Reports

Contain technical description of actual implementation

- STR-1 Orthogonalization Routines in SLEPc
- STR-2 Single Vector Iteration Methods in SLEPc
- STR-3 Subspace Iteration in SLEPc
- STR-4 Arnoldi Methods in SLEPc
- STR-5 Lanczos Methods in SLEPc
- STR-6 A Survey of Software for Sparse Eigenvalue Problems
- STR-7 Krylov-Schur Methods in SLEPc
- STR-8 Restarted Lanczos Bidiagonalization for the SVD in SLEPc
- STR-9 Practical Implementation of Harmonic Krylov-Schur

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

Orthogonalization

In Krylov methods, we need good quality of orthogonalization

- Classical GS is not numerically robust
- Modified GS is bad for parallel computing
- Modified GS can also be unstable in some cases

Solution: iterative Gram-Schmidt

Default in SLEPc: classical GS with selective reorthogonalization [Hernandez, Tomas, R., 2007]

$h_{1:j,j} = 0$
repeat
  $\rho = \|w\|_2$
  $c_{1:j,j} = V_j^* w$
  $w = w - V_j c_{1:j,j}$
  $h_{1:j,j} = h_{1:j,j} + c_{1:j,j}$
  $h_{j+1,j} = \sqrt{\rho^2 - \sum_{i=1}^{j} c_{i,j}^2}$
until $h_{j+1,j} > \eta\, \rho$

Restart

In most applications, restart is essential

Implicit restart

- Explicit restart is not powerful enough
- Implicit/thick restart keeps a lot of useful information...
- ... and purges unwanted eigenpairs
- Krylov-Schur is much easier to implement

Also

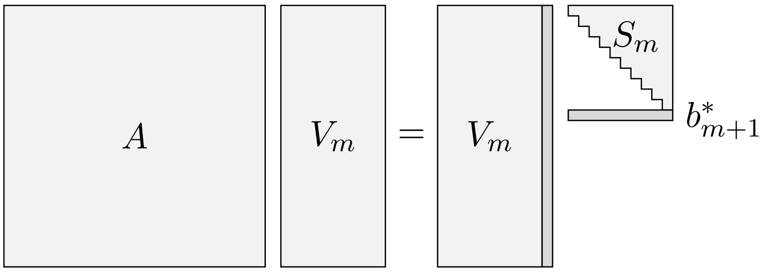
- Restart with locking is necessary in Lanczos for problems with multiple eigenvalues
- Semi-orthogonalization for Lanczos may not be worth with restart

Computation of Many Eigenpairs

By default, a subspace of dimension $2 \cdot nev$ is used...

For large nev, this is not appropriate

- Excessive storage and inefficient computation



Strategy: restrict the dimension of the projected problem

```
% program -eps_nev 2000 -eps_mpd 300
```


Stopping Criterion

Krylov methods provide an estimate of the residual norm

$$\|r_i\| = \beta_{m+1}|s_{m,i}|$$

The stopping criterion based on the normwise backward errors

$$\frac{\|r_i\|}{(a + |\theta_i| \cdot b)\|x_i\|} < \text{tol}$$

where

- $a = \|A\|, b = \|B\|$, or
- $a = 1, b = 1$

Warning: in shift-and-invert the above residual estimate is for $\|(A - \sigma B)^{-1}Bx_i - \theta_i x_i\|$
In SLEPc we allow for explicit computation of the residual

Preserving the Symmetry

In symmetric-definite generalized eigenproblems symmetry is lost because, e.g., $(A - \sigma B)^{-1}B$ is not symmetric

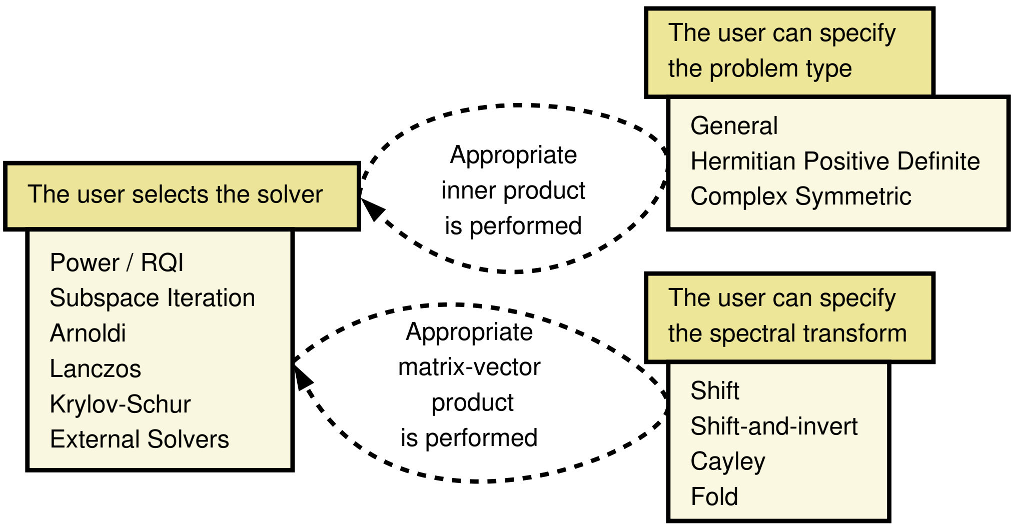
Choice of Inner Product

- Standard Hermitian inner product: $\langle x, y \rangle = y^* x$
- B-inner product: $\langle x, y \rangle_B = y^* B\, x$

Observations:

- $\langle \cdot, \cdot \rangle_B$ is a genuine inner product only if B is symmetric positive definite
- $\mathbb{R}^n$ with $\langle \cdot, \cdot \rangle_B$ is isomorphic to the Euclidean n-space $\mathbb{R}^n$ with the standard Hermitian inner product
- $(A - \sigma B)^{-1}B$ is self-adjoint with respect to $\langle \cdot, \cdot \rangle_B$

SLEPc Abstraction



These operations are virtual functions: STInnerProduct and STApply

Purification of Eigenvectors

When B is singular some additional precautions are required.
If $T = (A - \sigma B)^{-1} B$, then all finite eigenvectors belong to $\mathcal{R}(T)$

- $\langle x, y \rangle_B$ is a semi-inner product
- $\|x\|_B$ is a semi-norm
- $\langle x, y \rangle_B$ is a true inner product on $\mathcal{R}(T)$

Strategy:

- Force initial vector to lie in $\mathcal{R}(T)$
- Use Krylov method with $\langle \cdot, \cdot \rangle_B$

In finite precision arithmetic we need purification: remove components in the nullspace of B (e.g., with $x_i \leftarrow T x_i$)

Problems with Implicit Matrix

Example: cross-product matrix A^*A (for the SVD)

Example: linearized QEP

$$\begin{bmatrix} 0 & I \\ -K & -C \end{bmatrix} x = \lambda \begin{bmatrix} I & 0 \\ 0 & M \end{bmatrix} x$$

In SLEPc it is very easy to work with implicit matrices

1. Create an empty matrix (*shell matrix*)
2. Register functions to be called for certain operations (matrix-vector product but maybe others)
3. Use the matrix as a regular matrix

Options for Subspace Expansion

Initial Subspace

- Provide an initial trial subspace, e.g., from a previous computation (EPSSetInitialSpace)
- Krylov methods can only use one initial vector

Deflation Subspace

- Provide a deflation space with EPSSetDeflationSpace
- The eigensolver operates in the restriction to the orthogonal complement
- Useful for constrained eigenproblems or problems with a known nullspace
- Currently implemented as an orthogonalization

Parallel Layout - Vectors

Each process locally owns a subvector of contiguously numbered global indices

- simple block-row division of vectors
- other orderings through permutation

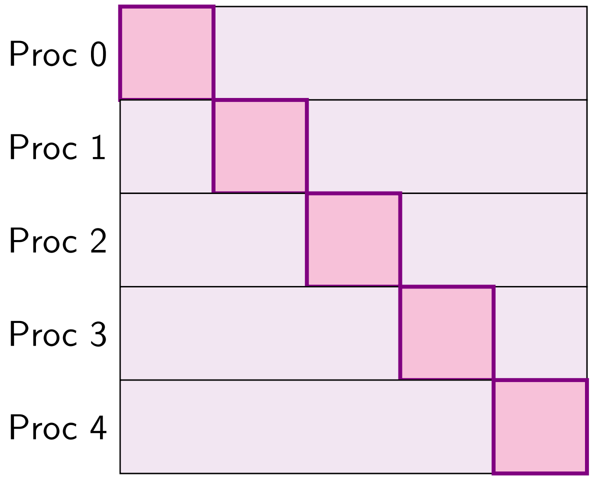
Vector dot products are inefficient (global reduction)

- avoid them if possible, or
- merge several together

Proc 0
Proc 1
Proc 2
Proc 3
Proc 4

Parallel Layout - Matrices

Each process locally owns a contiguous chunk of rows



Each processor stores the diagonal and off-diagonal parts separately

Sample Applications
Overview of SLEPc
Implementation Details

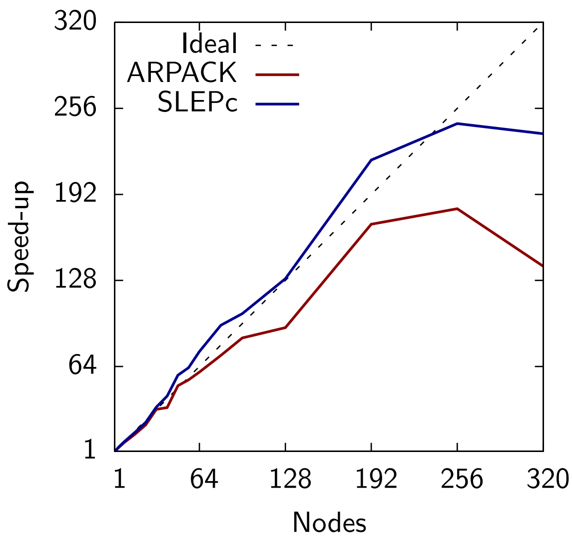
Parallel Performance on MareNostrum

Speed-up

$$S_p = \frac{T_s}{T_p}$$

Matrix PRE2

- University of Florida
- Dimension 659,033
- 5,834,044 nonzero elem.
- 10 eigenvalues with 30 basis vectors
- 1 processor per node



Wrap Up

SLEPc highlights:

- Free software
- Growing list of solvers
- Seamlessly integrated spectral transformation
- Easy programming with PETSc's object-oriented style
- Data-structure neutral implementation
- Run-time flexibility
- Portability to a wide range of parallel platforms
- Usable from code written in C, C++, Fortran, Python
- Extensive documentation

Next release: Jacobi-Davidson and QEP solvers

More Information

SLEPc

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

Hands-on Exercises:
`http://www.grycap.upv.es/slepc/handson`

Contact email:
`slepc-maint@grycap.upv.es`