Eigensolvers for Symmetric Generalized Eigenproblems in SLEPc

Jose E. Roman

D. Sistemes Informàtics i Computació
Universitat Politècnica de València, Spain

RES Engineering Seminar – 15th October 2014, Barcelona
Outline

1 Overview of SLEPc

2 Definite Problems
   - Exterior eigenvalues
   - Spectrum slicing

3 Indefinite Problems
   - Pseudo-Lanczos
   - Quadratic eigenproblems
Overview of SLEPc
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)
**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 SVD, QEP, and more

\[ Ax = \lambda x \quad Ax = \lambda Bx \quad Av_i = \sigma_i u_i \quad (\lambda^2 M + \lambda C + K)x = 0 \]

Authors: J. E. Roman, C. Campos, E. Romero, A. Tomas

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

Current version: 3.5 (released July 2014)
### Overview of SLEPc

#### Definite Problems

#### Indefinite Problems

## PETSc/SLEPc Numerical Components

### PETSc

<table>
<thead>
<tr>
<th>Nonlinear Systems</th>
<th>Time Steppers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line Search</td>
<td>Euler</td>
</tr>
<tr>
<td>Trust Region</td>
<td>Backward Euler</td>
</tr>
<tr>
<td>Other</td>
<td>Pseudo Euler</td>
</tr>
<tr>
<td>Other</td>
<td>Other</td>
</tr>
</tbody>
</table>

### Krylov Subspace Methods

<table>
<thead>
<tr>
<th>Method</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GMRES</td>
<td>CG</td>
<td>CGS</td>
<td>Bi-CGSTab</td>
<td>TFQMR</td>
</tr>
<tr>
<td>Richardson</td>
<td>Chebychev</td>
<td>Other</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Preconditioners

<table>
<thead>
<tr>
<th>Type</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Additive Schwarz</td>
<td>Block</td>
<td>Jacobi</td>
<td>ILU</td>
<td>ICC</td>
</tr>
<tr>
<td>Jacobi</td>
<td>ILU</td>
<td>ICC</td>
<td>LU</td>
<td>Other</td>
</tr>
</tbody>
</table>

### Matrices

<table>
<thead>
<tr>
<th>Type</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Compressed</td>
<td>Block</td>
<td>Symmetric</td>
<td>Dense</td>
<td>CUSP</td>
</tr>
<tr>
<td>Sparse Row</td>
<td>CSR</td>
<td>Block CSR</td>
<td>Other</td>
<td></td>
</tr>
</tbody>
</table>

### Vectors

<table>
<thead>
<tr>
<th>Type</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>CUSP</td>
</tr>
</tbody>
</table>

### Index Sets

<table>
<thead>
<tr>
<th>Type</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Indices</td>
<td>Block</td>
<td>Stride</td>
<td>Other</td>
</tr>
</tbody>
</table>
## PETSc/SLEPc Numerical Components

### PETSc

<table>
<thead>
<tr>
<th>Nonlinear Systems</th>
<th>Time Steppers</th>
<th>Krylov Subspace Methods</th>
<th>Preconditioners</th>
<th>Matrices</th>
<th>Vectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line Search</td>
<td>Euler</td>
<td>GMRES</td>
<td>Additive Schwarz</td>
<td>Compressed Sparse Row</td>
<td>Standard</td>
</tr>
<tr>
<td>Trust Region</td>
<td>Backward Euler</td>
<td>CG</td>
<td>Block Jacobi</td>
<td>Block CSR</td>
<td>CUSP</td>
</tr>
<tr>
<td>Other</td>
<td>Pseudo Time Step</td>
<td>CGS</td>
<td>Jacobi</td>
<td>Symmetric Block CSR</td>
<td>Indices</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bi-CGStab</td>
<td>ILU</td>
<td>Dense</td>
<td>Block</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TFQMR</td>
<td>ICC</td>
<td>CUSP</td>
<td>Stride</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Richardson</td>
<td>LU</td>
<td>Other</td>
<td>Other</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chebychev</td>
<td>Other</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### SLEPc

<table>
<thead>
<tr>
<th>Polynomial E-solver</th>
<th>Nonlinear Eigensolver</th>
<th>SVD Solver</th>
<th>Linear Eigensolver</th>
<th>Spectral Transformation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOAR</td>
<td>SLP</td>
<td>Cross Product</td>
<td>Krylov-Schur</td>
<td>Shift</td>
</tr>
<tr>
<td>Q-Arnoldi</td>
<td>RII</td>
<td>Cyclic Matrix</td>
<td>GD</td>
<td>Shift-and-invert</td>
</tr>
<tr>
<td>Linear-ization</td>
<td>N-Arnoldi</td>
<td>Lanczos</td>
<td>JD</td>
<td>Cayley</td>
</tr>
<tr>
<td>Interp.</td>
<td></td>
<td>Thick R. Lanczos</td>
<td>RQCG</td>
<td>Preconditioner</td>
</tr>
</tbody>
</table>

### M. Function

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Krylov</td>
</tr>
</tbody>
</table>

### Linear Systems

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Time Steppers

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Krylov Subspace Methods

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Preconditioners

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Matrices

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Vectors

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Index Sets

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Use Case: Neutron Diffusion Equation in Nuclear Eng.

Neutron power in nuclear reactor cores
- Commercial reactors such as PWR
- Both steady state and transient
- Goal: assure safety

\[ \mathcal{L}\phi = \frac{1}{\chi} M\phi \]

Current trends
- Complex geometries, unstructured meshes, FVM
- Coupled neutronic-thermalhydraulic calculations
Use Case: Gyrokinetic Equations in Plasma Physics

Plasma turbulence in a tokamak determines its energy confinement

- GENE code
- Initial value solver

Knowledge of the spectrum of the linearized equation

\[ Ax = \lambda x \]

- Complex, non-Hermitian, implicit \( A \)
- Sizes ranging from a few millions to a billion
- Estimate optimal timestep (largest eigenvalue); track sub-dominant instabilities (rightmost evals)
The SLEPc Project

Project started around 2000; 14 releases since 2002
- Software distribution with GNU LGPL license
- User support via email
- Collaborations in cases of special interest

13,000 downloads (since 2003)
\( \sim 200 \) per month (last year)

Main distributions (Linux, MacOSX, PyPI)

Supercomputing centers: BSC, NERSC, NCSA, Jülich
Applications

Google Scholar: 284 citations of main paper (ACM TOMS 2005)

<table>
<thead>
<tr>
<th>Application</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nuclear Engineering</td>
<td>6 %</td>
</tr>
<tr>
<td>Computational Electromagnetics, Electronics, Photonics</td>
<td>9 %</td>
</tr>
<tr>
<td>Plasma Physics</td>
<td>11 %</td>
</tr>
<tr>
<td>Astrophysics</td>
<td>1 %</td>
</tr>
<tr>
<td>Computational Physics, Materials Science, Electronic Structure</td>
<td>20 %</td>
</tr>
<tr>
<td>Acoustics</td>
<td>5 %</td>
</tr>
<tr>
<td>Computational Fluid Dynamics</td>
<td>12 %</td>
</tr>
<tr>
<td>Earth Sciences, Oceanology, Hydrology, Geophysics</td>
<td>4 %</td>
</tr>
<tr>
<td>Bioengineering, Computational Neuroscience</td>
<td>3 %</td>
</tr>
<tr>
<td>Structural Analysis, Mechanical Engineering</td>
<td>5 %</td>
</tr>
<tr>
<td>Information Retrieval, Machine Learning, Graph Algorithms</td>
<td>7 %</td>
</tr>
<tr>
<td>Visualization, Computer Graphics, Image Processing</td>
<td>3 %</td>
</tr>
<tr>
<td>PDE’s, Numerical Methods</td>
<td>10 %</td>
</tr>
<tr>
<td>Dynamical Systems, Model Reduction, Inverse Problems</td>
<td>4 %</td>
</tr>
</tbody>
</table>
Definite Problems
Linear Eigenvalue Problems

Compute a few eigenpairs \((\lambda, x)\) of

**Standard Eigenproblem**
\[ Ax = \lambda x \]

**Generalized Eigenproblem**
\[ Ax = \lambda Bx \]

where \(A, B\) can be real or complex, symmetric (Hermitian) or not

User can specify:
- Number of eigenpairs \((n_{ev})\), subspace dimension \((n_{cv})\)
- Selected part of spectrum
- Tolerance, maximum number of iterations
- Many solvers: Krylov-Schur, GD, JD, RQCG, CISS, ...
- Solver options: extraction type, balancing, ...
- Initial guesses, deflation space
**MAGPACK: Molecular Clusters**

Analysis of high-nuclearity spin clusters

- Collaboration with ICMol (Valencia)
- Goal: study magnetic susceptibility and magnetization as well as inelastic neutron scattering spectra

Example: ring of 10 Ni(II) ions with antiferromagnetic interaction

- Dimension: 59,049
- Nonzero elements: $\sim$ 35 million
- Real, symmetric, indefinite
- 600 leftmost eigenvalues
Molecular Clusters - Real Use Case: Fe8

Study: octanuclear iron, Fe8 (n=1,679,616)

Classical example of single molecule magnet, with interesting properties

- Slow magnetic moment relaxation at low temperatures
- Strong axial anisotropy

S. Cardona-Serra, J.M. Clemente-Juan, E. Coronado, E. Ramos and J. E. Roman, “Parallel implementation of the MAGpack, the Fe8 example”, in preparation.
Molecular Clusters - Results of Fe8 Simulations

Magnetic susceptibility

- Good match against experimental data
  - Results obtained with 2000 eigenvalues
  - Computations in MareNostrum (during a visit of E. Ramos) with 4096 processors
  - Low temperatures (<40K) computed with anisotropy (7 diagonalizations), high with isotropic analysis (21 decoupled matrices)
Molecular Clusters - Results of Fe8 Simulations

Magnetic susceptibility

- Good match against experimental data
  - Results obtained with 2000 eigenvalues
  - Computations in MareNostrum (during a visit of E. Ramos) with 4096 processors
  - Low temperatures (<40K) computed with anisotropy (7 diagonalizations), high with isotropic analysis (21 decoupled matrices)

4 magnetic field orientations, 10 magnetic fields
- 2 diagonalizations per point (80 in total)
- Only 400 low-energy states
- Mostly computed in Tirant w/1024 procs
- About 8 hours each diagonalization

Magnetization at 2K
Spectral Transformation

The shift-and-invert transformation enables Lanczos methods to compute interior eigenvalues

\[ Ax = \lambda Bx \quad \Rightarrow \quad (A - \sigma B)^{-1} Bx = \theta x \]

- Trivial mapping of eigenvalues: \( \theta = (\lambda - \sigma)^{-1} \)
- Eigenvectors are not modified
- Very fast convergence close to \( \sigma \)
Spectral Transformation

The shift-and-invert transformation enables Lanczos methods to compute interior eigenvalues

\[ Ax = \lambda Bx \implies (A - \sigma B)^{-1} Bx = \theta x \]

- Trivial mapping of eigenvalues: \( \theta = (\lambda - \sigma)^{-1} \)
- Eigenvectors are not modified
- Very fast convergence close to \( \sigma \)

Things to consider:
- Implicit inverse \( (A - \sigma B)^{-1} \) via linear solves
- Direct linear solver for robustness
- Less effective for eigenvalues far away from \( \sigma \)
Spectrum Slicing

Indefinite (block-)triangular factorization:

\[ A - \sigma B = LDL^T \]

By Sylvester’s law of inertia, we get as a byproduct the number of eigenvalues on the left of \( \sigma \)

\[ \nu(A - \sigma B) = \nu(D) \]
Spectrum Slicing

Indefinite (block-)triangular factorization:

\[ A - \sigma B = LDL^T \]

By Sylvester’s law of inertia, we get as a byproduct the number of eigenvalues on the left of \( \sigma \)

\[ \nu(A - \sigma B) = \nu(D) \]

Strategy:

- Multi-shift scheme that sweeps all the interval
- Compute eigenvalues by chunks
- Use inertia to validate sub-intervals

Spectrum Slicing

Indefinite (block-)triangular factorization:

\[ A - \sigma B = LDL^T \]

By Sylvester’s law of inertia, we get as a byproduct the number of eigenvalues on the left of \( \sigma \)

\[ \nu(A - \sigma B) = \nu(D) \]

Strategy:
- Multi-shift scheme that sweeps all the interval
- Compute eigenvalues by chunks
- Use inertia to validate sub-intervals

Spectrum Slicing

Indefinite (block-)triangular factorization:

\[ A - \sigma B = LDL^T \]

By Sylvester’s law of inertia, we get as a byproduct the number of eigenvalues on the left of \( \sigma \)

\[ \nu(A - \sigma B) = \nu(D) \]

Strategy:

- Multi-shift scheme that sweeps all the interval
- Compute eigenvalues by chunks
- Use inertia to validate sub-intervals

Spectrum Slicing

Indefinite (block-)triangular factorization:

\[ A - \sigma B = LDL^T \]

By Sylvester’s law of inertia, we get as a byproduct the number of eigenvalues on the left of \( \sigma \)

\[ \nu(A - \sigma B) = \nu(D) \]

Strategy:

- Multi-shift scheme that sweeps all the interval
- Compute eigenvalues by chunks
- Use inertia to validate sub-intervals

Spectrum Slicing

Indefinite (block-)triangular factorization:

\[ A - \sigma B = LDL^T \]

By Sylvester’s law of inertia, we get as a byproduct the number of eigenvalues on the left of \( \sigma \)

\[ \nu(A - \sigma B) = \nu(D) \]

Strategy:
- Multi-shift scheme that sweeps all the interval
- Compute eigenvalues by chunks
- Use inertia to validate sub-intervals

Spectrum Slicing

Indefinite (block-)triangular factorization:

\[ A - \sigma B = LDL^T \]

By Sylvester’s law of inertia, we get as a byproduct the number of eigenvalues on the left of \( \sigma \)

\[ \nu(A - \sigma B) = \nu(D) \]

Strategy:

- Multi-shift scheme that sweeps all the interval
- Compute eigenvalues by chunks
- Use inertia to validate sub-intervals

Spectrum Slicing

Indefinite (block-)triangular factorization:

\[ A - \sigma B = LDL^T \]

By Sylvester’s law of inertia, we get as a byproduct the number of eigenvalues on the left of \( \sigma \)

\[ \nu(A - \sigma B) = \nu(D) \]

Strategy:

- Multi-shift scheme that sweeps all the interval
- Compute eigenvalues by chunks
- Use inertia to validate sub-intervals

Test Case: Aircraft Fuselage

Simplified but realistic model: cylinder with skin, frames, and stringers

Parametric, “scalable”

First vibration mode (5.34 Hz)
Test Case: Matrix Properties

Analysis of frequency range [0–60] Hz

1 million dof’s
- Dimension: 1,036,698
- Nonzeros: ~29 million
- Eigenvalues in interval: 1989

2 million dof’s
- Dimension: 2,141,646
- Nonzeros: ~59 million
- Eigenvalues in interval: 2039

Maximum multiplicity: 2

$B$ is singular
## Evaluation: Parallel Performance

### 1 million, \( \text{nev}=120 \)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>12</td>
<td>21</td>
<td>4,318</td>
<td>16,399</td>
<td>3,153</td>
<td>594</td>
<td>5,270</td>
<td>6,218</td>
</tr>
<tr>
<td>16</td>
<td>13</td>
<td>27</td>
<td>4,565</td>
<td>10,125</td>
<td>1,739</td>
<td>595</td>
<td>4,277</td>
<td>3,364</td>
</tr>
<tr>
<td>32</td>
<td>10</td>
<td>21</td>
<td>4,101</td>
<td>5,500</td>
<td>519</td>
<td>653</td>
<td>2,839</td>
<td>1,428</td>
</tr>
<tr>
<td>64</td>
<td>12</td>
<td>22</td>
<td>4,338</td>
<td>4,064</td>
<td>375</td>
<td>654</td>
<td>2,482</td>
<td>580</td>
</tr>
<tr>
<td>128</td>
<td>12</td>
<td>20</td>
<td>4,155</td>
<td>3,394</td>
<td>273</td>
<td>596</td>
<td>2,265</td>
<td>245</td>
</tr>
</tbody>
</table>

### 2 million, \( \text{nev}=120 \)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>13</td>
<td>24</td>
<td>4,873</td>
<td>14,429</td>
<td>2,012</td>
<td>1,691</td>
<td>6,536</td>
<td>3,902</td>
</tr>
<tr>
<td>64</td>
<td>12</td>
<td>22</td>
<td>4,472</td>
<td>9,120</td>
<td>922</td>
<td>1,697</td>
<td>4,886</td>
<td>1,263</td>
</tr>
<tr>
<td>128</td>
<td>12</td>
<td>23</td>
<td>4,501</td>
<td>7,817</td>
<td>707</td>
<td>1,692</td>
<td>4,709</td>
<td>668</td>
</tr>
</tbody>
</table>

**CaesarAugusta** (256 JS20 nodes, 2 PowerPC 970+, 4 GB/node, Myrinet)
Overview of SLEPc

Definite Problems

Indefinite Problems

Multi-Communicator Spectrum Slicing

$p$ processes, $\ell$ partitions with $p/\ell$ processes each

- Each group processes a subinterval, with a parallel linear solver
- Goal: improve scalability
- Needs more memory: store $A, B$ redundantly

Computational work will likely differ in each subinterval

- A priori knowledge of eigenvalue distribution can be used to improve load balance
- User can set subinterval boundaries
Indefinite Problems
Quadratic Eigenvalue Problem

QEP:

\[(\lambda^2 M + \lambda C + K)x = 0\]

where \(M, C, K\) are real symmetric (or complex Hermitian)

Solution strategies:

1. Symmetric linearization with pseudo-Lanczos [Parlett, Chen 1990]

\[
\begin{bmatrix}
0 & -K \\
-K & -C \\
\end{bmatrix} - \lambda \begin{bmatrix}
-K & 0 \\
0 & M \\
\end{bmatrix}
\]

2. Disregard symmetry, use Arnoldi solver
   - Memory-efficient variant: Q-Arnoldi [Meerbergen 2008]
   - Stabilized variant: TOAR [Su et al. 2008]

3. Combine both

Pseudo-Lanczos

Adapt $B$-Lanczos to the case of indefinite $\langle \cdot, \cdot \rangle_B$

\[
\begin{align*}
\text{for } j &= 1, 2, \ldots, k \\
u &= S v_j \\
t_{1:j,j} &= V_j^* B u \\
u &= u - V_j \Omega_j^{-1} t_{1:j,j} \\
t_{j+1,j} &= \sqrt{|u^* B u|} \\
\omega_{j+1} &= \text{sign}(u^* B u) \\
v_{j+1} &= u \omega_{j+1} / t_{j+1,j}
\end{align*}
\]

end

$S := B^{-1} A$

Orthogonalization:
\begin{itemize}
\item Full $B$-orthogonalization (twice)
\item Must insert $\Omega_j^{-1}$ in the projector used in Gram-Schmidt
\end{itemize}

Computes Lanczos relation $SV_k = V_k \Omega_k^{-1} T_k + t_{k+1,k} \omega_{k+1}^{-1} v_{k+1} e_k^*$
\begin{itemize}
\item Lanczos vectors $V_k$ satisfy $V_k^* B V_k = \Omega_k$
\end{itemize}
Quadratic Arnoldi

Consider the linearization of the QEP

\[
A - \lambda B = \begin{bmatrix}
0 & N \\
-K & -C
\end{bmatrix} - \lambda \begin{bmatrix}
N & 0 \\
0 & M
\end{bmatrix}
\]

with either \( N = I \) or \( N = -K \). The eigenvector is \([x^*, \lambda x^*]^*\)
Quadratic Arnoldi

Consider the linearization of the QEP

\[ A - \lambda B = \begin{bmatrix} 0 & N \\ -K & -C \end{bmatrix} - \lambda \begin{bmatrix} N & 0 \\ 0 & M \end{bmatrix} \]

with either \( N = I \) or \( N = -K \). The eigenvector is \([x^*, \lambda x^*]^*\)

Q-Arnoldi implicitly computes an Arnoldi relation for \( S = B^{-1}A \)

\[
\begin{bmatrix}
0 & I \\
-M^{-1}K & -M^{-1}C
\end{bmatrix}
\begin{bmatrix}
V_k^0 \\
V_k^1
\end{bmatrix}
=
\begin{bmatrix}
V_k^0 \\
V_k^1
\end{bmatrix}
\begin{bmatrix}
v^0 \\
v^1
\end{bmatrix}
H_k
\]

From the 1st row block:

\[
V_k^1 = [V_k^0, v^0]H_k
\]

Hence, only \( \{V_k^0, H_k, v\} \) needs to be stored
Q-Arnoldi / Q-Lanczos

For each $j = 1, \ldots, k$

1. Expand: $w^0 = v^1$, $w^1 = -M^{-1}(Kv^0 + Cv^1)$

2. Gram-Schmidt coefficients
   \[ t = (V^1_{j-1})^* w^1 = H^*_{j-1} \left[ V^0_{j-1} \ v^0 \right]^* w^1 \]
   \[ h_j = \begin{bmatrix} V^0_{j-1} & v^0 \\ V^1_{j-1} & v^1 \end{bmatrix}^* w = \begin{bmatrix} (V^0_{j-1})^* w^0 + t \\ v^0 * w^0 + v^1 * w^1 \end{bmatrix} \]

3. Gram-Schmidt update
   \[ \tilde{w}^0 = w^0 - \begin{bmatrix} V^0_{j-1} & v^0 \end{bmatrix} h_j \]
   \[ \tilde{w}^1 = w^1 - \begin{bmatrix} V^0_{j-1} & v^0 \end{bmatrix} H_{j-1} \begin{bmatrix} v^0 \\ v^1 \end{bmatrix} h_j \]

4. Normalize
   \[ h_{j+1,j} = \sqrt{||\tilde{w}^0||^2 + ||\tilde{w}^1||^2} \]
   \[ v^0 = \tilde{w}^0 / h_{j+1,j}, \quad v^1 = \tilde{w}^1 / h_{j+1,j} \]
Q-Arnoldi / Q-Lanczos

For each $j = 1, \ldots, k$

1. Expand: $w^0 = v^1$, $w^1 = -M^{-1}(Kv^0 + Cv^1)$

2. Gram-Schmidt coefficients
   
   
   
   $t = (V^1_{j-1})^* M w^1 = H^*_{j-1} [V^0_{j-1} v^0]^* M w^1$
   
   $h_j = \left[ \begin{array}{c} V^0_{j-1} v^0 \\ V^1_{j-1} v^1 \end{array} \right]^* B w = \left[ \begin{array}{c} -(V^0_{j-1})^* Kw^0 + t \\ -v^0*Kw^0 + v^1*Mw^1 \end{array} \right]$, $h_j = \Omega_{j+1}^{-1} h_j$

3. Gram-Schmidt update
   
   $\tilde{w}^0 = w^0 - [V^0_{j-1} v^0] h_j$
   
   $\tilde{w}^1 = w^1 - [V^0_{j-1} v^0] H_{j-1} v^1] h_j$

4. Normalize
   
   $h_{j+1,j} = \|\tilde{w}\|_B$, $\omega_{j+1,j+1} = \text{sgn}(\|\tilde{w}\|_B)$
   
   $v^0 = \tilde{w}^0 / h_{j+1,j}$, $v^1 = \tilde{w}^1 / h_{j+1,j}$
Some Experiments with SLEPc Implementation

Maximum residual error for tolerance $10^{-9}$, 20 eigenpairs with maximum basis size $k = 40$ (times for 64 processes)

<table>
<thead>
<tr>
<th>name</th>
<th>method</th>
<th>nconv</th>
<th>its</th>
<th>$\max_i{|r_i|}$</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>acoustic-wave_2d</td>
<td>Q-Arnoldi</td>
<td>66</td>
<td>3</td>
<td>2.0e-10</td>
<td>172.8</td>
</tr>
<tr>
<td>$n = 6, 247, 500$</td>
<td>Q-Lanczos</td>
<td>66</td>
<td>2</td>
<td>1.8e-06</td>
<td>178.0</td>
</tr>
<tr>
<td>$\sigma = 0$</td>
<td>TOAR</td>
<td>66</td>
<td>3</td>
<td>1.4e-10</td>
<td>170.3</td>
</tr>
<tr>
<td>not scaled</td>
<td>STOAR</td>
<td>60</td>
<td>3</td>
<td>8.6e-10</td>
<td>176.3</td>
</tr>
<tr>
<td>sleeper</td>
<td>Krylov-Schur</td>
<td>40</td>
<td>2</td>
<td>1.8e-12</td>
<td>60.5</td>
</tr>
<tr>
<td>$n = 5, 000, 000$</td>
<td>Pseudo-Lanczos</td>
<td>37</td>
<td>2</td>
<td>5.2e-15</td>
<td>80.4</td>
</tr>
<tr>
<td>$\sigma = -0.9$</td>
<td>Q-Arnoldi</td>
<td>41</td>
<td>2</td>
<td>7.0e-09</td>
<td>70.8</td>
</tr>
<tr>
<td>not scaled</td>
<td>Q-Lanczos</td>
<td>46</td>
<td>2</td>
<td>0.017</td>
<td>82.6</td>
</tr>
<tr>
<td>TOAR</td>
<td></td>
<td>41</td>
<td>2</td>
<td>4.2e-07</td>
<td>70.3</td>
</tr>
<tr>
<td>STOAR</td>
<td></td>
<td>37</td>
<td>2</td>
<td>9.4e-04</td>
<td>71.2</td>
</tr>
</tbody>
</table>
Parallel Performance

Parallel computing times on MareNostrum III from Barcelona Supercomputing Center

acoustic_wave_2d

sleeper
Wrap Up

SLEPc is continually growing

- Increasing number of users, application areas
- New releases bring new features and optimizations

MareNostrum and friends very valuable for scalability analysis

- RES users can benefit immediately from new developments

Latest additions and things to come soon

- Nonlinear eigenproblems: polynomial, rational, general
- Contour integral for selecting a region of the complex plane
- Hybrid parallelism: multicore, GPU