Filtering techniques for eigenvalue problems
Yousef Saad

Department of Computer Science and Engineering

University of Minnesota

Emory University - Math-CS colloquium
Feb. 22, 2019
Background. Origins of Eigenvalue Problems

- Structural Engineering \[ Ku = \lambda Mu \] (Goal: frequency response)
- Electronic structure calculations [Schrödinger equation..]
- Stability analysis [e.g., electrical networks, mechanical system,..]
- Bifurcation analysis [e.g., in fluid flow]

▷ Large eigenvalue problems in quantum chemistry use up biggest portion of the time in supercomputer centers
Background. New applications in data analytics

- Machine learning problems often require a (partial) Singular Value Decomposition -

- Somewhat different issues in this case:
  - Very large matrices, update the SVD
  - Compute dominant singular values/vectors
  - Many problems of approximating a matrix (or a tensor) by one of lower rank (Dimension reduction, ...)

- But: Methods for computing SVD often based on those for standard eigenvalue problems
**Background. The Problem(s)**

- **Standard eigenvalue problem:**
  \[ Ax = \lambda x \]

  Often: \( A \) is symmetric real (or Hermitian complex)

- **Generalized problem**
  \[ Ax = \lambda Bx \]

  Often: \( B \) is symmetric positive definite, \( A \) is symmetric or nonsymmetric

- **Quadratic problems:**
  \[ (A + \lambda B + \lambda^2 C)u = 0 \]

- **Nonlinear eigenvalue problems (NEVP)**
  \[ \left[ A_0 + \lambda B_0 + \sum_{i=1}^{n} f_i(\lambda) A_i \right] u = 0 \]
General form of NEVP

\[ A(\lambda)x = 0 \]

Nonlinear eigenvector problems:

\[ [A + \lambda B + F(u_1, u_2, \cdots, u_k)]u = 0 \]

What to compute:

- A few \( \lambda_i \)'s with smallest or largest real parts;
- All \( \lambda_i \)'s in a certain region of \( \mathbb{C} \);
- A few of the dominant eigenvalues;
- All \( \lambda_i \)'s (rare).
Large eigenvalue problems in applications

Some applications require the computation of a large number of eigenvalues and vectors of very large matrices.

Density Functional Theory in electronic structure calculations: ‘ground states’

Excited states involve transitions and invariably lead to much more complex computations. → Large matrices, *many* eigenpairs to compute
Computing earth normal modes (J. Shi & M. V. De Hoop)

- FEM model leads to a generalized eigenvalue problem
- Compute (a large number of) eigenvalues in an interval
- More on this later
**Background: The main tools**

*Projection process:*

(a) Build a ‘good’ subspace $K = \text{span}(V)$;

(b) get approximate eigenpairs by a Rayleigh-Ritz process: $	ilde{\lambda}, \tilde{u} \in K$ satisfy: $(A - \tilde{\lambda}I)\tilde{u} \perp K \implies V^H (A - \tilde{\lambda}I) Vy = 0$

- $\tilde{\lambda} =$ Ritz value, $\tilde{u} = Vy =$ Ritz vector

- Two common choices for $K$:
  1) Power subspace $K = \text{span}\{A^kX_0\}$; or $\text{span}\{P_k(A)X_0\}$;
  2) Krylov subspace $K = \text{span}\{v, Av, \ldots, A^{k-1}v\}$
Shift-and-invert:

If we want eigenvalues near $\sigma$, replace $A$ by $(A - \sigma I)^{-1}$.

Example: power method: $v_j = A v_{j-1} / \text{scaling}$ replaced by

$$v_j = \frac{(A - \sigma I)^{-1} v_{j-1}}{\text{scaling}}$$

Works well for computing a few eigenvalues near $\sigma$.

Used in commercial package NASTRAN (for decades!)

Requires factoring $(A - \sigma I)$ (or $(A - \sigma B)$ in generalized case.) But convergence will be much faster.

A solve each time - Factorization done once (ideally).
Deflation:

➢ Once eigenvectors converge remove them from the picture

Restarting Strategies:

➢ Restart projection process by using information gathered in previous steps

➢ ALL available methods use some combination of these ingredients.

[e.g. ARPACK: Arnoldi/Lanczos + ‘implicit restarts’ + shift-and-invert (option).]
Solving large eigenvalue problems: Current state-of-the art

- Eigenvalues at one end of the spectrum:
  - Subspace iteration + filtering [e.g. FEAST, Cheb,...]
  - Lanczos+variants (no restart, thick restart, implicit restart, Davidson,..), e.g., ARPACK code, PRIMME.
  - Block Algorithms [Block Lanczos, TraceMin, LOBPCG, SlePSc,...]
  - + Many others - more or less related to above

- ‘Interior’ eigenvalue problems (middle of spectrum):
  - Combine shift-and-invert + Lanczos/block Lanczos. Used in, e.g., NASTRAN
  - Rational filtering [FEAST, Sakurai et al.,... ]
Solving large interior eigenvalue problems

Three broad approaches:

1. Shift-invert: $A \rightarrow (A - \sigma I)^{-1}$

2. Polynomial filtering: $A \rightarrow p(A)$

3. Rational filtering: $A \rightarrow \sum \alpha_i (A - \sigma_i I)^{-1}$

Issues with shift-and invert (and related approaches)

- Direct methods for the solves may be too expensive
  - Why not use iterative methods?

- Iterative techniques often fail –
  - Reason: Highly indefinite problems.
Filtering and “Spectrum Slicing”

- Context: very large number of eigenvalues to be computed
- Goal: compute spectrum by slices by applying filtering
- Apply Lanczos or Sub-space iteration to problem:

\[ \phi(A)u = \mu u \]

\( \phi(t) \equiv \) a polynomial or rational function that enhances wanted eigenvalues
Compute slices separately

For each slice Do:

\[[\text{get *all* eigenpairs in a slice}]\]

EndDo

Goal: Compute each slice independently from the others.
**Rationale.** Eigenvectors associated with different slices need not be orthogonalized against each other:

- Can get the spectrum by ‘slices’ or ‘windows’ [e.g., a few hundreds or thousands of pairs at a time]
- Note: Orthogonalization + RR cost can be very high if we do not slice the spectrum
Illustration: All eigenvalues in [0, 1] of a $49^3$ Laplacean

Note: This is a small pb. in a scalar environment. Effect likely much more pronounced in a fully parallel case.
POLYNOMIAL FILTERS
Polynomial filtering

- Apply Lanczos or Subspace iteration to:

\[ M = \phi(A) \]

where \( \phi(t) \) is a polynomial

- Each matvec \( y = Av \) is replaced by \( y = \phi(A)v \)

- Eigenvalues in high part of filter will be computed first

- Old (forgotten) idea. But new context is *very* favorable
What polynomials?

➢ For end-intervals: use standard Chebyshev polynomials (1st kind)

➢ For ‘interior case’ we need a polynomial that has large values for $\lambda \in [a, b]$ small values elsewhere
**Simplest technique: δ-Dirac function**

- Obtain the LS approximation to the δ—Dirac function — Centered at some point (TBD) inside the interval.

- Can use same damping: Jackson, Lanczos σ damping, or none.
The Chebyshev expansion of $\delta_\gamma$ is

$$\rho_k(t) = \sum_{j=0}^{k} \mu_j T_j(t) \quad \text{with} \quad \mu_j = \begin{cases} \frac{1}{2} & j = 0 \\ \cos(j \cos^{-1}(\gamma)) & j > 0 \end{cases}$$

Recall: The delta Dirac function is not a function – we can’t properly approximate it in least-squares sense. However:

**Proposition** Let $\hat{\rho}_k(t)$ be the polynomial that minimizes $\|r(t)\|_w$ over all polynomials $r$ of degree $\leq k$, such that $r(\gamma) = 1$, where $\|\cdot\|_w$ represents the Chebyshev $L^2$-norm. Then $\hat{\rho}_k(t) = \rho_k(t)/\rho_k(\gamma)$. 
‘The soul of a new filter’ – A few technical details

\[ p_m(t) = \sum_{j=0}^{m} \gamma_j^{(m)} \mu_j T_j(t) \]

\[ \mu_k = \begin{cases} 
1/2 & \text{if } k = 0 \\
\cos(k \cos^{-1}(\gamma)) & \text{otherwise}
\end{cases} \]

\[ \gamma_j^{(m)} = \text{Damping coefficients.} \]

- quite simple...
- .. provided we handle a few practical issues
**Issue # one:** ‘balance the filter’

- To facilitate the selection of ‘**wanted**’ eigenvalues [Select $\lambda$’s such that $\phi(\lambda) > \text{bar}$] we need to ...
- … find $\gamma$ so that $\phi(\xi) = = \phi(\eta)$

**Procedure:** Solve the equation $\phi_{\gamma}(\xi) - \phi_{\gamma}(\eta) = 0$ with respect to $\gamma$, accurately. Use Newton or eigenvalue formulation.
**Issue # two:** Determine degree & polynomial (automatically)

1) Start low (e.g. 2); 2) Increase degree until value (s) at the boundary (ies) become small enough –

Can also use criterion based on derivatives at $\xi$ & $\eta$
A zoom on the final polynomial found
Issue # Three: Gibbs oscillations

- Discontinuous ‘function’ approximated → Gibbs oscillations

- Three options:
  - No damping
  - Jackson damping
  - Lanczos $\sigma$ damping

- Good compromise: Lanczos $\sigma$ damping
COMBINING FILTERING WITH A PROJECTION METHOD
Algorithm builds orthonormal basis $V_m = [v_1, v_2, \cdots, v_m]$ for the Krylov subspace: $\text{span}\{v_1, Av_1, \cdots, A^{m-1}v_1\}$

... such that: $V_m^H AV_m = T_m$ - with $T_m = 
\begin{pmatrix}
\alpha_1 & \beta_2 \\
\beta_2 & \alpha_2 & \beta_3 \\
\beta_3 & \alpha_3 & \beta_4 \\
\vdots & \ddots & \ddots & \ddots \\
\beta_m & \alpha_m \\
\end{pmatrix}$

Note: three term recurrence:

$\beta_{j+1}v_{j+1} = Av_j - \alpha_jv_j - \beta_jv_{j-1}$

Eigenvalues of $A$ on both ends of spectrum are well approximated by eigenvalues of $T_m$ (Ritz values).
Which Projection: Lanczos, w/o restarts, Subspace iteration,..

**Options:**

- Subspace iteration: quite appealing in some applications (e.g., electronic structure): Can re-use previous subspace.
- Simplest: (+ most efficient) Lanczos without restarts
- Lanczos with Thick-Restarting [TR Lanczos, Stathopoulos et al ’98, Wu & Simon’00]
- Crucial tool in TR Lanczos: deflation (’Locking’)

**Main idea:** Keep extracting eigenvalues in interval $[\xi, \eta]$ until none are left.

- If filter is good: Can catch all eigenvalues in interval
Polynomial filtered Lanczos: No-Restart version

Use Lanczos with full reorthogonalization on $\rho(A)$. Eigenvalues of $\rho(A)$: $\rho(\lambda_i)$

- Accept if $\rho(\lambda_i) \geq \bar{\rho}$
- Ignore if $\rho(\lambda_i) < \bar{\rho}$
How do I slice a spectrum?

- Tools: Density of States (used in EVSL) or eigenvalue counts (used in FEAST)
  - L. Lin, YS, Chao Yang [Siam review ’16] – E. Di Napoli, E. Polizzi, YS [’16]
  - KPM method – see, e.g.,: [Weisse, Wellein, Alvermann, Fehske, ’06]
  - Interesting instance of a tool from physics used in linear algebra.

- Misconception: ‘load balancing will be assured by just having slices with roughly equal numbers of eigenvalues’

- In fact - will help mainly in balancing memory usage.
Slice spectrum into 8 with the DOS

We must have:

\[
\int_{t_i}^{t_{i+1}} \phi(t) \, dt = \frac{1}{n_{slices}} \int_{a}^{b} \phi(t) \, dt
\]
RATIONAL FILTERS
Why use rational filters?

- Consider a spectrum like this one:

- Polynomial filtering utterly ineffective for this case

- Second issue: situation when Matrix-vector products are expensive

- Generalized eigenvalue problems.
Alternative is to use rational filters:

\[ \phi(z) = \sum_j \frac{\alpha_j}{z - \sigma_j} \]

\[ \phi(A) = \sum_j \alpha_j (A - \sigma_j I)^{-1} \]

We now need to solve linear systems

Tool: Cauchy integral representations of spectral projectors

\[ P = \frac{-1}{2i\pi} \int_{\Gamma} (A - sI)^{-1} ds \]

- Numer. integr. \( P \rightarrow \tilde{P} \)
- Use Krylov or S.I. on \( \tilde{P} \)

Sakurai-Sugiura approach [Krylov]

Polizzi [FEAST, Subsp. Iter. ]
What makes a good filter

Assume subspace iteration is used with above filters. Which filter will give better convergence?

Simplest and best indicator of performance of a filter is the magnitude of its derivative at -1 (or 1)
The Gauss viewpoint: Least-squares rational filters

- Given: poles $\sigma_1, \sigma_2, \ldots, \sigma_p$
- Related basis functions $\phi_j(z) = \frac{1}{z-\sigma_j}$

Find $\phi(z) = \sum_{j=1}^{p} \alpha_j \phi_j(z)$ that minimizes

$$\int_{-\infty}^{\infty} w(t)|h(t) - \phi(t)|^2 dt$$

- $h(t) = \text{step function } \chi_{[-1,1]}$.
- $w(t) = \text{weight function}$. For example $a = 10$, $\beta = 0.2$

$$w(t) = \begin{cases} 0 & \text{if } |t| > a \\ \beta & \text{if } |t| \leq 1 \\ 1 & \text{else} \end{cases}$$
How does this work?

- Small example: Laplacean on a $43 \times 53$ grid. ($n = 2279$)
- 4 poles obtained from mid-point rule
- Want: all ($nev = 31$) eigenvalues in $[0, 0.2]$
- Use 1) standard subspace iteration + Cauchy (FEAST) then 2) subspace iteration + LS Rat. Appox.
LS Uses the same poles + same factorizations as Cauchy but

... much faster as expected from a look at the curves of the functions
Other advantages:

- Can select poles far away from real axis → faster iterative solvers
- Very flexible – can be adapted to many situations
- Can repeat poles (!)

Implemented in EVSL.. [Interfaced to UMFPACK as a solver]
Spectrum Slicing and the EVSL project

- EVSL package now at version 1.1.x
- Uses polynomial and rational filtering: Each can be appealing in different situations.

*Spectrum slicing: Invokes Kernel Polynomial Method or Lanczos quadrature to cut the overall interval containing the spectrum into small sub-intervals.*

![Graph of spectrum slicing](image-url)
Levels of parallelism

The two main levels of parallelism in EVSL
EVSL Main Contributors (version 1.1.0+) & Support

- Ruipeng Li
  LLNL
- Yuanzhe Xi
  Asst. Prof. Emory
- Luke Erlandson
  PhD Student, GTech.

➤ Work supported by NSF (past work: DOE)

➤ See web-site for details:

http://www-users.cs.umn.edu/~saad/software/EVSL/
EVSL: current status & plans

Version _1.0_ Released in Sept. 2016

- Matrices in CSR format (only)
- Standard Hermitian problems (no generalized)
- Spectrum slicing with KPM (Kernel Polynomial Meth.)
- Trivial parallelism across slices with OpenMP

Methods:
- Non-restart Lanczos – polynomial & rational filters
- Thick-Restart Lanczos – polynomial & rational filters
- Subspace iteration – polynomial & rational filters
**Version _1.1.x_**

V_1.1.0 Released back in August 2017.

- **general matvec [passed as function pointer]**
- \( Ax = \lambda Bx \)
- Fortran (03) interface.
- Spectrum slicing by Lanczos and KPM
- Efficient Spectrum slicing for \( Ax = \lambda Bx \) (no solves with \( B \)).

**Version _1.2.x_**

pEVSL – In progress

- Fully parallel version [MPI + openMP]
**Spectrum slicing and the EVSL package**

- All eigenvalues in $[0, 1]$ of a $49^3$ discretized Laplacian
- `eigs(A,1971,'sa')`: 14830.66 sec
- Solution: Use DOS to partition $[0, 1]$ into 5 slices
- Polynomial filtering from EVSL on Mesabi MSI, 23 threads/slice

<table>
<thead>
<tr>
<th>$[a_i, a_{i+1}]$</th>
<th># eigs</th>
<th>CPU time (sec)</th>
<th>max residual</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>matvec</td>
<td>orth.</td>
</tr>
<tr>
<td>[0.00000, 0.37688]</td>
<td>386</td>
<td>1.31</td>
<td>18.26</td>
</tr>
<tr>
<td>[0.37688, 0.57428]</td>
<td>401</td>
<td>3.28</td>
<td>38.25</td>
</tr>
<tr>
<td>[0.57428, 0.73422]</td>
<td>399</td>
<td>4.69</td>
<td>36.47</td>
</tr>
<tr>
<td>[0.73422, 0.87389]</td>
<td>400</td>
<td>5.97</td>
<td>38.60</td>
</tr>
<tr>
<td>[0.87389, 1.00000]</td>
<td>385</td>
<td>6.84</td>
<td>36.16</td>
</tr>
</tbody>
</table>

➢ Grand tot. = 263 s. Time for slicing the spectrum: 1.22 sec.
Computing the Earth normal modes

- Collaborative effort: Rice-UMN:
  J. Shi, R. Li, Y. Xi, YS, and M. V. De Hoop

- FEM model leads to a generalized eigenvalue problem:
\[
\begin{bmatrix}
A_s & E_{fs} \\
\quad 0 & A_d \\
E_{fs}^T & A_d^T & A_p
\end{bmatrix}
\begin{bmatrix}
u^s \\
u^f \\
p^e
\end{bmatrix}
= \omega^2
\begin{bmatrix}
M_s & \\
M_f & 0
\end{bmatrix}
\begin{bmatrix}
v^s \\
v^f \\
p^e
\end{bmatrix}
\]

- Want all eigen-values/vectors inside a given interval
- Issue 1: ‘mass’ matrix has a large null space..
- Issue 2: interior eigenvalue problem
- Solution for 1: change formulation of matrix problem [eliminate \(p^e\) ...]
New formulation:

\[
\left\{ \begin{pmatrix} A_s & 0 \\ 0 & 0 \end{pmatrix} - \begin{pmatrix} E_{fs} \\ A_d \end{pmatrix} A_p^{-1} \begin{pmatrix} E_{fs}^T & A_d^T \end{pmatrix} \right\} \begin{pmatrix} u_s \\ u_f \end{pmatrix} = \hat{A} \omega^2 \begin{pmatrix} M_s & 0 \\ 0 & M_f \end{pmatrix} \begin{pmatrix} u_s \\ u_f \end{pmatrix}
\]

Use polynomial filtering – need to solve with \( \hat{M} \) but ...

- ... severe scaling problems if direct solvers are used

Hence:

- Replace action of \( M^{-1} \) by a low-deg. polynomial in \( M \) [to avoid direct solvers]
Memory: parallel shift-invert and polynomial filtering

Machine: Comet, SDSC

<table>
<thead>
<tr>
<th>Matrix size</th>
<th># Proc.s</th>
</tr>
</thead>
<tbody>
<tr>
<td>591, 303</td>
<td>32</td>
</tr>
<tr>
<td>1, 157, 131</td>
<td>64</td>
</tr>
<tr>
<td>2, 425, 349</td>
<td>128</td>
</tr>
<tr>
<td>4, 778, 004</td>
<td>256</td>
</tr>
<tr>
<td>9, 037, 671</td>
<td>512</td>
</tr>
</tbody>
</table>

![Graph showing memory consumption for different matrix sizes and number of processors. The graph includes bars for shift-invert (peak), shift-invert (avg), and poly. filtering.]
Recent: weak calability test for different solid (Mars-like) models on TACC Stampede2

<table>
<thead>
<tr>
<th>nn/np</th>
<th>Mat-size</th>
<th>$A\nu$ ($ms$)</th>
<th>← Eff.</th>
<th>$M\nu$ ($ms$)</th>
<th>← Eff.</th>
<th>$M^{-1}\nu$ ($\mu s$)</th>
<th>← Eff.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2/96</td>
<td>1,038,084</td>
<td>1760</td>
<td>1.0</td>
<td>495</td>
<td>1.0</td>
<td>0.01044</td>
<td>1.0</td>
</tr>
<tr>
<td>4/192</td>
<td>2,060,190</td>
<td>1819</td>
<td>0.960</td>
<td>568</td>
<td>0.865</td>
<td>0.0119</td>
<td>0.870</td>
</tr>
<tr>
<td>8/384</td>
<td>3,894,783</td>
<td>1741</td>
<td>0.948</td>
<td>571</td>
<td>0.813</td>
<td>0.0119</td>
<td>0.825</td>
</tr>
<tr>
<td>16/768</td>
<td>7,954,392</td>
<td>1758</td>
<td>0.959</td>
<td>621</td>
<td>0.763</td>
<td>0.0129</td>
<td>0.774</td>
</tr>
<tr>
<td>32/1536</td>
<td>15,809,076</td>
<td>1660</td>
<td>1.009</td>
<td>572</td>
<td>0.824</td>
<td>0.0119</td>
<td>0.834</td>
</tr>
<tr>
<td>64/3072</td>
<td>31,138,518</td>
<td>1582</td>
<td>1.043</td>
<td>566</td>
<td>0.820</td>
<td>0.0117</td>
<td>0.837</td>
</tr>
<tr>
<td>128/6144</td>
<td>61,381,362</td>
<td>1435</td>
<td>1.133</td>
<td>546</td>
<td>0.838</td>
<td>0.0113</td>
<td>0.851</td>
</tr>
<tr>
<td>256/12288</td>
<td>120,336,519</td>
<td>1359</td>
<td>1.173</td>
<td>592</td>
<td>0.757</td>
<td>0.01221</td>
<td>0.774</td>
</tr>
</tbody>
</table>
Nonlinear eigenvalue problems

- Joint work with A. Miedlar and M. Elguide

\[ T(z)u = 0 \quad \text{\( z \rightarrow T(z) \) maps } \mathbb{C} \text{ to } \mathbb{C}^{n \times n} \]

- Classical (well-understood) case: Polynomial:
  \[ T(z) = A_0 + zA_1 + \cdots + z^p A_p \]

- Often treated with linearization, e.g., when \( p = 2 \)
  \[ (A_0 + zA_1 + z^2 A_2)u = 0 \quad \rightarrow \text{ (among other forms) } \]
  \[ \begin{bmatrix} 0 & I \\ -A_0 & -A_1 \end{bmatrix} - z \begin{bmatrix} I & 0 \\ 0 & A_2 \end{bmatrix} \begin{bmatrix} u \\ zu \end{bmatrix} = 0 \]

- General case can be very different from linear case.
Restrict slightly the class of problems we consider:

\[ T(z) = -B_0 + zA_0 + f_1(z)A_1 + \ldots + f_p(z)A_p \]

- Main assumption: each of the analytic functions \( f_j : \Omega \to \mathbb{C} \) well approximated by a rational function.

- Write (Cauchy integral representation of \( f_j \)):
  \[
  f_j(z) = -\frac{1}{2i\pi} \int_{\Gamma} \frac{f_j(t)}{z - t} \, dt, \quad z \in \Omega.
  \]

- Then use numerical quadrature with quadrature points \( \sigma_i \)'s on contour \( \Gamma \):
  \[
  f_j(z) \approx r_j(z) \equiv \sum_{i=1}^{m} \frac{\alpha_{ij}}{z - \sigma_i}.
  \]
Consequence: $T(z)$ approximated by

$$
\tilde{T}(z) = -B_0 + zA_0 + \sum_{j=1}^{p} \sum_{i=1}^{m} \frac{\alpha_{ij}}{z - \sigma_i} A_j = \ldots
$$

$$
\equiv -B_0 + zA_0 + \sum_{i=1}^{m} \frac{B_i}{z - \sigma_i}, \quad \text{where:}
$$

$$
B_i = \sum_{j=0}^{p} \alpha_{ij} A_j, \quad i = 1, \ldots, m.
$$

$$
\left[ -B_0 + zA_0 + \sum_{i=1}^{m} \frac{B_i}{z - \sigma_i} \right] u = 0
$$

‘Surrogate’ for original problem $T(z)u = 0$
Linearization

\[ v_i = \frac{u}{\sigma_i - z} \rightarrow \tilde{T}(z)u = (-B_0 + zA_0)u - \sum_{i=1}^{m} B_i v_i, \]

\[ \tilde{T}(\lambda)u = 0 \text{ iff } Aw = \lambda Mw \]

where:

\[ M = \begin{bmatrix} I & \cdots & \cdots & I \\ I & \cdots & \cdots & I \\ \vdots & \ddots & \ddots & \vdots \\ \cdots & \cdots & \cdots & A_0 \end{bmatrix}, \quad A = \begin{bmatrix} \sigma_1 I & -I \\ \sigma_2 I & -I \\ \vdots & \vdots & \ddots & \vdots \\ B_1 & B_2 & \cdots & B_m & B_0 \end{bmatrix}. \]

- Eigenvalue problem of size \( n(m + 1) \)
- Special form: matrix need not be stored explicitly.
Approaches

1. Can use a shift-and-invert Arnoldi on whole system [Pb: memory when \( m \gg 1 \)]
   - Block structure exploited.

2. Can use a shift-and-invert Subspace iteration [memory: similar pb.]
   - Advantages: Less memory, ‘one-shot-method’ can be very efficient (memory)

3. Add restart to 2 but work only with vectors of length \( n \).
Reduced Subspace Iteration: (Case when $\mathcal{M} = I$)

1. While Convergence not yet reached
2. For $j = 1 : \nu$
3. Select $w = [v; u]$ // See below
4. Do $q$ steps of inverse iteration: $w := (A - \sigma I)^{-1}w$
5. If $w = [v; u] \equiv$ last iterate, set $U(:, j) = u$
6. EndFor
7. Use $U$ to perform Rayleigh-Ritz procedure
8. EndWhile

Step 2:
(1) Very first outer loop: take random vectors.
(2) Other outer iterations: If $(\lambda, u)$ is an eigenpair from step 7, define $v$-part as $v_i = u / (\sigma_i - \lambda)$ - then:

$$w = [v_1; v_2; \cdots; v_m; u] \quad \text{(Matlab notation)}$$
Accuracy of computed eigenvalues

**Proposition** Let us assume that \( \| f_j(z) - r_j(z) \|_{\Omega_1} \leq \varepsilon \) for \( j = 1, \cdots, p \) and let \((\tilde{\lambda}, \tilde{u})\) be an exact eigenpair of the surrogate problem with \( \tilde{\lambda} \) located inside \( \Omega_1 \) and \( \| \tilde{u} \| = 1 \) for a certain vector norm \( \| \cdot \| \). Let \( \mu = \sum_{j=1}^{p} \| A_j \| \). Then,

\[
\| T(\tilde{\lambda})\tilde{u} \| \leq \mu \varepsilon.
\]

**Proposition** Let us assume that \( \| f_j(z) - r_j(z) \|_{\Omega_1} \leq \varepsilon \) for \( j = 1, \cdots, p \) and let \((\lambda, u)\) be an exact eigenpair for \( T(z) \) with \( \lambda \) located inside \( \Omega_1 \) and \( \| u \| = 1 \). Then, \((\lambda, u)\) is an approximate eigenpair of the surrogate problem, i.e.,

\[
\| \tilde{T}(\lambda)u \| \leq \mu \varepsilon,
\]

where \( \mu \) is defined above.
The halo of extraneous eigenvalues

- Observed behavior: many ‘extraneous’ or ‘spurious’ eigenvalues congregate around the contour of integration.

**Example:** \( T(z) = -B_0 + \lambda A_0 + \lambda^2 A_2 \) where [Matlab] (n=4)

\[
B_0 = -2 \times \text{eye}(n) + \text{diag(ones(n-1,1),1)} + \text{diag(ones(n-1,1),-1)}; \\
A_0 = \text{eye}(n); \\
A_2 = 0.5 \times (n \times \text{eye}(n) - \text{eye}(n,1) \times \text{ones}(1,n) - \text{ones}(n,1) \times \text{eye}(1,n));
\]

- Spectrum inside rectangle with bottom-left and top-right corners \((-1, -1.5i), (0, 1.5i)\)

- Use this for integration contour.
**Left:** The 8 eigenvalues of original problem (circle); the 4 eigenvalues of the linear part (square); contour and quadrature points along it.

**Right:** Eigenvalues computed with $m = 20$ quadrature points (plus) along with contour, original eigenvalues (circle), and eigenvalues of linear part (square).
Using a total of $m = 32$ quadrature points (left) and $m = 60$ quadrature points (right).

(i) Spectrum of Linear part outside contour APPROXIMATED
(ii) Spectrum of Linear part inside contour IGNORED
(iii) Spectrum of $T(z)$ inside contour APPROXIMATED
(iv) Other eigenvalues populate the contour
Example

Hadeler problem of dimension $n = 200$:

$$T(\lambda) = (e^\lambda - 1)B_1 + \lambda^2 B_2 - B_0$$

with:

- $B_0 = b_0 I$, $b_0 = 100$
- $b_{jk}^{(1)} = (n + 1 - \max(j, k))j k$
- $b_{jk}^{(2)} = n\delta_{jk} + 1/(j + k)$,
Eigenvalues of Hadeler Pb. inside a circle of radius $r = 10$ and center $c = -30$ obtained by the reduced subspace iteration ('+'), and by Beyn’s method ('O'). Quadrature: Gauss-Legendre with 50 points.

Current work: Helmholtz equation (in 3-D):

$$\Delta u + k^2 u = 0 \quad + B.C.$$  

Using the Boundary Element Method (BEM) produces a nonlinear eigenvalue problem.
Conclusion

- **EVSL** code available here: [Current version: version 1.1.1]
  [www.cs.umn.edu/~saad/software/EVSL]

- **EVSL** Also on github (development)

**Plans:**
1. Release fully parallel code;
2. Block versions;
3. Iterative solvers for rational filt.;
4. Nonhermitian case;

- Earth modes calculations done with fully parallel code
- Scalability issues with parallel direct solvers ...
- ... Needed: iterative solvers for the highly indefinite case
- Frontier in eigenvalue problem: **Nonlinear** case