Divide and conquer algorithms for large eigenvalue problems

Yousef Saad

Department of Computer Science and Engineering

University of Minnesota

PMAA 14
Lugano, July 4, 2014
Collaborators:

- Joint work with: Haw-ren Fang and Vassileos Kalantzis
- Grady Schoefield and Jim Chelikowsky [UT Austin] [windowing into PARSEC]
- Work supported in part by NSF (to 2012) and now by DOE
How do you compute eigenvalues in the middle of the spectrum of a large Hermitian matrix?

Common practice: Shift and invert + some projection process (Lanczos, subspace iteration..)

Main steps:

1) Select a shift (or sequence of shifts) $\sigma$;
2) Factor $A - \sigma I$: $A - \sigma I = LDL^T$;
3) Apply Lanczos algorithm to $(A - \sigma I)^{-1}$

- Solves with $A - \sigma I$ carried out using factorization
- Limitation: factorization
Q: What if factoring $A$ is too expensive (e.g., Large 3-D simulation)?

A: Obvious answer: Use iterative solvers ...

➢ But: systems highly indefinite $\rightarrow$ Won’t work well.

➢ Other common issue: Need a very large number of eigenvalues and eigenvectors

➢ Applications: Excited states in quantum physics: TDDFT, GW, ... or just plain Density Functional Theory (DFT)

➢ Example: in real-space code (PARSEC), Hamiltonian can be of size a few Millions, and number of ev’s in the tens of thousands
I. Polynomial filtered Lanczos

Possible solution: Use Lanczos with polynomial filtering.

In short: just replace \((A - \sigma I)^{-1}\) in S.I. Lanczos by \(p_k(A)\) where \(p_k(t) = \text{polynomial of degree } k\)

Idea not new (and not too popular in the past)

What is new?

1. Very large problems;
2. (tens of) Thousands of eigenvalues;
3. Parallelism.

Important application: compute the spectrum by pieces ['spectrum slicing' a term coined by B. Parlett]

Main attraction: reduce cost of orthogonalization
**Low-pass, high-pass, & barrier (mid-pass) filters**

- **Base Filter**: $\psi(\lambda)$
- **Polynomial Filter**: $\rho(\lambda)$

**Left Diagram**:
- $[a,b]=[0,3]$, $[\xi,\eta]=[0,1]$, $d=10$
- Plot of $f(\lambda)$ vs $\lambda$

**Right Diagram**:
- $[a,b]=[0,3]$, $[\xi,\eta]=[0.5,1]$, $d=20$
- Plot of $f(\lambda)$ vs $\lambda$

- See Reference on Lanczos + pol. filtering: Bekas, Kokiopoulou, YS (2008) for motivation, etc.

- H.-r Fang and YS “Filtlan” paper [SISC,2012] and code
Misconception: High degree polynomials are bad.

Degree 1000 (zoom)

Degree 1600 (zoom)
‘Spectrum slicing’ or ‘windowing’

**Rationale.** Eigenvectors on both ends of wanted spectrum need not be orthogonalized against each other:

- Idea: Get the spectrum by ‘slices’ or ’windows’
- Can get a few hundreds or thousands of vectors at a time.
Compute eigenpairs one slice at a time

Deceivingly simple looking idea.

Issues:
- Deal with interfaces: duplicate/missing eigenvalues
- Window size [need estimate of eigenvalues]
- Polynomial degree
Spectrum slicing in PARSEC

- Implemented in our code:

Pseudopotential Algorithm for Real-Space Electronic Calculations (PARSEC)

- See:


- Refer to this paper for details on windowing and ‘initial proof of concept’
Computing the polynomials: Jackson-Chebyshev

Chebyshev-Jackson approximation of a function $f$:

$$f(x) \approx \sum_{i=0}^{k} g_i^k \gamma_i T_i(x)$$

$$\gamma_i = \frac{2 - \delta_{i0}}{\pi} \int_{-1}^{1} \frac{1}{\sqrt{1 - x^2}} f(x) \, dx \quad \delta_{i0} = \text{Kronecker symbol}$$

$\gamma_i = \text{explicitly known}$

$\rightarrow$

- The $g_i^k$'s for $k=50,100,150$
- The $g_i^k$'s dampen high order terms
- Alternative: Lanczos $\sigma$-damping
A mid-pass (barrier) filter - 3 damping methods

Mid-pass polynom. filter \([-1 .3 .6 1]\); Degree = 30

- Standard Cheb.
- Jackson–Cheb.
- σ Lanczos
Experiments on two dual-core AMD Opteron(tm) Processors 2214 @ 2.2GHz and 16GB memory.

**Test matrices:**

* Five Hamiltonians from electronic structure calculations,

* **Andrews matrix** $N = 60,000$, $nnz \approx 760 K$, interval $[4, 5]$; nev=1,844 eigenvalues, (3,751 to the left of $\eta$)

* A discretized Laplacian (FD) $n = 10^6$, interval = $[1, 1.01]$, nev= 276, (>17,000 on the left of $\eta$)

Here : report only on Andrews and Laplacean
### Results for *Andrews* - set 1 of stats

<table>
<thead>
<tr>
<th>method</th>
<th>degree</th>
<th># iter</th>
<th># matvecs</th>
<th>memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>filt. Lan. (mid-pass)</td>
<td>$d = 20$</td>
<td>9,440</td>
<td>188,800</td>
<td>4,829</td>
</tr>
<tr>
<td>filt. Lan.</td>
<td>$d = 30$</td>
<td>6,040</td>
<td>180,120</td>
<td>2,799</td>
</tr>
<tr>
<td>filt. Lan. (high-pass)</td>
<td>$d = 50$</td>
<td>3,800</td>
<td>190,000</td>
<td>1,947</td>
</tr>
<tr>
<td>filt. Lan.</td>
<td>$d = 100$</td>
<td>2,360</td>
<td>236,000</td>
<td>1,131</td>
</tr>
<tr>
<td>filt. Lan. (high-pass)</td>
<td>$d = 10$</td>
<td>5,990</td>
<td>59,900</td>
<td>2,799</td>
</tr>
<tr>
<td>filt. Lan.</td>
<td>$d = 20$</td>
<td>4,780</td>
<td>95,600</td>
<td>2,334</td>
</tr>
<tr>
<td>filt. Lan. (high-pass)</td>
<td>$d = 30$</td>
<td>4,360</td>
<td>130,800</td>
<td>2,334</td>
</tr>
<tr>
<td>filt. Lan.</td>
<td>$d = 50$</td>
<td>4,690</td>
<td>234,500</td>
<td>2,334</td>
</tr>
<tr>
<td>Part. ⊥ Lanczos</td>
<td></td>
<td>22,345</td>
<td>22,345</td>
<td>10,312</td>
</tr>
<tr>
<td>ARPACK</td>
<td></td>
<td>30,716</td>
<td>30,716</td>
<td>6,129</td>
</tr>
</tbody>
</table>
### Results for Andrews - CPU times (sec.)

<table>
<thead>
<tr>
<th>method</th>
<th>degree</th>
<th>$\rho(A)v$</th>
<th>reorth</th>
<th>eigvec</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>filt. Lan.</td>
<td>$d = 20$</td>
<td>2,797</td>
<td>192</td>
<td>4,834</td>
<td>9,840</td>
</tr>
<tr>
<td>(mid-pass)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$d = 30$</td>
<td>2,429</td>
<td>115</td>
<td>2,151</td>
<td>5,279</td>
</tr>
<tr>
<td></td>
<td>$d = 50$</td>
<td>3,040</td>
<td>65</td>
<td>521</td>
<td>3,810</td>
</tr>
<tr>
<td></td>
<td>$d = 100$</td>
<td>3,757</td>
<td>93</td>
<td>220</td>
<td>4,147</td>
</tr>
<tr>
<td>filt. Lan.</td>
<td>$d = 10$</td>
<td>1,152</td>
<td>2,911</td>
<td>2,391</td>
<td>7,050</td>
</tr>
<tr>
<td>(high-pass)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$d = 20$</td>
<td>1,335</td>
<td>1,718</td>
<td>1,472</td>
<td>4,874</td>
</tr>
<tr>
<td></td>
<td>$d = 30$</td>
<td>1,806</td>
<td>1,218</td>
<td>1,274</td>
<td>4,576</td>
</tr>
<tr>
<td></td>
<td>$d = 50$</td>
<td>3,187</td>
<td>1,032</td>
<td>1,383</td>
<td>5,918</td>
</tr>
<tr>
<td>Part. ⊥ Lanczos</td>
<td></td>
<td>217</td>
<td>30,455</td>
<td>64,223</td>
<td>112,664</td>
</tr>
<tr>
<td>ARPACK</td>
<td></td>
<td>345</td>
<td>†423,492</td>
<td>†18,094</td>
<td>441,934</td>
</tr>
</tbody>
</table>
## Results for Laplacian – Matvecs and Memory

<table>
<thead>
<tr>
<th>method</th>
<th>degree</th>
<th># iter</th>
<th># matvecs</th>
<th>memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>mid-pass filter</td>
<td>600</td>
<td>1,400</td>
<td>840,000</td>
<td>10,913</td>
</tr>
<tr>
<td></td>
<td>1,000</td>
<td>950</td>
<td>950,000</td>
<td>7,640</td>
</tr>
<tr>
<td></td>
<td>1,600</td>
<td>710</td>
<td>1,136,000</td>
<td>6,358</td>
</tr>
</tbody>
</table>

## Results for Laplacian – CPU times

<table>
<thead>
<tr>
<th>method</th>
<th>degree</th>
<th>$\rho(A)v$</th>
<th>reorth</th>
<th>eigvec</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>mid-pass filter</td>
<td>600</td>
<td>97,817</td>
<td>927</td>
<td>241</td>
<td>99,279</td>
</tr>
<tr>
<td></td>
<td>1,000</td>
<td>119,242</td>
<td>773</td>
<td>162</td>
<td>120,384</td>
</tr>
<tr>
<td></td>
<td>1,600</td>
<td>169,741</td>
<td>722</td>
<td>119</td>
<td>170,856</td>
</tr>
</tbody>
</table>
II. Domain decomposition ideas

- Main idea: Cauchy integral-based method [e.g. FEAST]
- ... within a domain-decomposition framework:

Two classical ways of partitioning a graph.

- We use edge-separators (vertex-based partitioning)
Distributed graph and its matrix representation

- Stack all interior variables $u_1, u_2, \ldots, u_p$ into a vector $u$, then interface variables $y$

- Result:

$$
\begin{pmatrix}
B_1 & \cdots & E_1 \\
B_2 & \cdots & E_2 \\
\vdots & \ddots & \vdots \\
E_1^T & E_2^T & \cdots & E_p^T \\
E_1^T & E_2^T & \cdots & E_p^T \\
\end{pmatrix}
\begin{pmatrix}
B_p & E_p \\
1 & \vdots & \vdots & \vdots & \vdots \\
C & 1 & \vdots & \vdots & \vdots \\
\end{pmatrix}
\begin{pmatrix}
B_1 & \cdots & E_1 \\
B_2 & \cdots & E_2 \\
\vdots & \ddots & \vdots \\
E_1^T & E_2^T & \cdots & E_p^T \\
E_1^T & E_2^T & \cdots & E_p^T \\
\end{pmatrix}
\begin{pmatrix}
u_1 \\
u_2 \\
\vdots \\
u_p \\
y \\
\end{pmatrix}
= \lambda
\begin{pmatrix}
u_1 \\
u_2 \\
\vdots \\
u_p \\
y \\
\end{pmatrix}
Notation:

Write as:

\[ A = \begin{pmatrix} B & E \\ E^T & C \end{pmatrix} \]
First idea: Schur complement techniques (On-going work)

- Eliminate interior variables \( u_i \) – Result:

\[
\begin{pmatrix}
S_1(\lambda) & E_{12} & \cdots & E_{1p} \\
E_{21} & S_2(\lambda) & \cdots & E_{2p} \\
\vdots & \ddots & \ddots & \vdots \\
E_{p1} & E_{p,2} & \cdots & S_p(\lambda)
\end{pmatrix}
\begin{pmatrix}
y_1 \\
y_2 \\
\vdots \\
y_p
\end{pmatrix} = 0
\]

\( S_i(\lambda) = C_i - E_i^T (B - \lambda I)^{-1} E_i \equiv \text{Local Schur Complement} \)

- Nonlinear eigen-value problem.

- Related to AMLS – see also Bekas and YS (2005)

- Involves only interface variables.
Next: Schur complements + FEAST

\[
A - sI = \begin{pmatrix} B - sI & E \\ ET & C - sI \end{pmatrix} \rightarrow \]

\[
(A - sI)^{-1} = \begin{bmatrix} * & -(B - sI)^{-1}ES(s)^{-1} \\ * & S(s)^{-1} \end{bmatrix}
\]

Then, Cauchy integral formula for spectral projector yields:

\[
P = \frac{-1}{2i\pi} \int_{\Gamma} R(s)ds \equiv \begin{bmatrix} * & -W \\ * & G \end{bmatrix}
\]

with

\[
G = \frac{-1}{2i\pi} \int_{\Gamma} S(s)^{-1}ds, \quad W = \frac{-1}{2i\pi} \int_{\Gamma} (B - sI)^{-1}ES(s)^{-1}ds
\]

Advantage: Does not involve inverse of whole matrix
Let
\[ P = [P_1, P_2] \equiv \begin{bmatrix} * & -W \\ * & G \end{bmatrix} \]

We know how to compute \( P_2 \) or \( P_2 \times \text{randn}(s, ns) \)

Q: How can we recover eigenvectors of \( A \) from \( P_2 \)?

A: Write \( P \) as \( P = VV^T \), and \( V = \begin{pmatrix} V_u \\ V_s \end{pmatrix} \) then note:
\[ P_2 = VV_s^T \]

Just capture the range of \( P_2 \)

Can use Lanczos on \( P_2P_2^T \) or just a random \( X \in \mathbb{R}^{s \times ns} \)

Advantage of Lanczos: stops when dimension reached

Drawbacks: 1) sequential; 2) \( \approx \) Doubles the work

So far: Both idea tested in matlab
Need better poles

- Approach is a one-shot method [no easy way to iterate]

**Q:** How can we improve accuracy?

**A:** Select poles carefully.

- Current choices: trapezoidal rule, Gauss, Zolotarev, ...

- None of these allows for repeated ('multiple') poles e.g.,

\[
 r(z) = \frac{\alpha_1}{z - \sigma} + \frac{\alpha_2}{(z - \sigma)^2} + \cdots + \frac{\alpha_k}{(z - \sigma)^k}
\]

- This can be useful for any rational filtering approach

- Next: See what we can do with one double pole
double pole
Gauss 1−pole
Gauss 2−poles
double pole
Gauss 1-pole
Gauss 2-poles
Two double poles + comparison with compounding

Sigma = +/-0.6 +/-1i; pow = 2 2

- Multiple pole
- Single pole
- Gauss 2-poles
- Gauss 2-poles²
Who needs a circle? Two poles squared far from the origin

Sigma = +/-0.05 +/-1.5i; pow = 2 2
Conclusion

Part I:

- FiltLan is appealing when number of eigenvectors to be computed is large and when Matvecs are not too expensive
- Will not work too well for generalized eigenvalue problem
- Code available here [www.cs.umn.edu/~saad/software/filtlan](http://www.cs.umn.edu/~saad/software/filtlan)

Part II:

- Many ideas still to explore in Domain Decomposition for interior eigenvalue problems
- Viewpoint: look at rational filtering from angle of approx. theory