Large-scale Eigenvalue Calculations in Scientific Problems

Esmond G. Ng
Lawrence Berkeley National Laboratory
The team

- Metin Aktulga
- Lin Lin
- Esmond Ng
- Eugene Vecharynski
- Chao Yang

- Summer student - Christopher Haine

- Funding source: DOE SciDAC Program
Eigenvalue problems

- Sources of eigenvalue problems
  - Accelerator physics (e.g., modeling of accelerator cavities)
  - Nuclear physics (e.g., understanding of nuclear structures)
  - Chemical sciences (e.g., understanding excited state electronic structure)
  - Materials science (e.g., understanding properties of materials)

- Types of eigenvalue problems
  - Linear problems
    \[Ax = \lambda Bx\]
  - Nonlinear problems
    \[H(X)X = X\Lambda\]
    \[F(\lambda)x = 0\]
Focus on large-scale linear eigenvalue problems

- Computing a few smallest/largest/interior eigenvalues
  - Examples: the ground or low excited states of a many-body Schrödinger’s equation, structure analysis (resonant frequency)
  - Methods
    - Krylov subspace based methods (Lanczos, Krylov-Schur, etc.)
    - Jacobi-Davidson
    - Optimization based method (trace minimization)
Large-scale symmetric linear eigenvalue problems

- Computing a relatively “large” number of eigenpairs
  - What is “large”?  
    - 1% of one million = 10,000
  - Kohn-Sham density functional theory based electronic structure calculation. The number of eigenpairs is proportional to the number of electrons (hundred to thousands to hundreds of thousands depending on the system)
  - Excited state calculation through Green’s function formalism of many-body perturbation theory (the GW approximation)
Methods for computing many eigenpairs

- LAPACK or ScaLAPACK if the NEV is a significant portion of the matrix dimension (50%, 30% or maybe even 20%) unless the eigenvectors are structured (e.g., block diagonal)
- Compute all at once using block methods
  - EigPen
  - LOBPCG
  - Chebyshev-Davidson
  - Block Krylov-Schur
- Spectrum slicing
  - Divide the spectrum into subintervals
  - Compute interior eigenvalues within each interval
Why spectrum slicing?

- More concurrency; eigenvalues belonging to different intervals computed (almost) independently

- Reduced Rayleigh-Ritz/orthogonalization cost
What to use in each interval?

- **Shift-invert Lanczos (MSIL)**
  \[(A - \sigma I)^{-1}x = \frac{1}{\lambda - \sigma} x\]

- **Contour integral projection method (MCISPM)**
  \[P = \frac{1}{2\pi i} \oint_{\Gamma} (A - zI)^{-1} dz \approx \sum_{1}^{n_p} \omega_i (A - z_i I)^{-1}\]

- **Polynomial transformation**
Optimal partition?

- **Objective:**
  - Load balancing
  - Rapid convergence within each interval
  - Minimize the number of factorizations (hence the total cost)

- **Dynamic (Boeing code, SIPs) vs static**

- **Estimate of the density of state**
  - Inertia count (Sylvester’s inertia theorem)
  - Other methods that do not require matrix factorization

- **Estimate the cost of computing** $k$ **eigenvalues within each interval**
Estimate the (cumulative) density of state
Cost model

- **Assumption:**
  - Matrix dimension: $n$
  - $q$ processors per interval
  - Uniform eigenvalue distribution
  - Factorization and triangular solution costs: $c_f n^{\alpha_f}, c_s n^{\alpha_s}$
  - Parallel efficiencies: $\eta_f, \eta_s \in (0, 1)$
  - $p = c_p n$, $nev = c_n n$
  - Rayleigh-Ritz and orthogonalization costs are negligible

- Wall clock time for computing $k$ eigenpairs per interval:

\[
W(k) = \frac{c_f n^{\alpha_f}}{q^{\eta_f} q^{\eta_s}} + \frac{c_s n^{\alpha_s} k}{q^{\eta_f} q^{\eta_s}}
\]

\[
k_{opt} = \frac{\eta_f c_f}{(1 - \eta_s) c_s} \left( \frac{c_n}{c_p} \right)^{\frac{\eta_f - \eta_s}{\eta_f - \eta_s + 1}} \frac{n^{\alpha_f - \alpha_s}}{n^{\eta_f - \eta_s + 1}}
\]
Observations

- When $c_n / c_p$ is constant, optimal $k$ depends on problem size and relative cost of factorization and triangular solution

$$k_{opt} = \left( \frac{\eta_f c_f}{(1-\eta_s)c_s} \right)^{\frac{1}{\eta_f-\eta_s+1}} \left( \frac{c_n}{c_p} \right)^{\frac{\eta_f-\eta_s}{\eta_f-\eta_s+1}} n$$

- When $\eta_f, \eta_s$ are close to 1, we should place more eigenvalues in each interval without increasing Rayleigh-Ritz and orthogonalization cost/slowing down convergence

- When $\eta_f = \eta_s = \eta$
  - $k_{opt} = \left( \frac{\eta c_f}{(1-\eta)c_s} \right) n^{\alpha_f-\alpha_s}$ independent of the total # of processors
  - $W(k_{opt}) = \frac{c_f n^{\alpha_f}}{u} + \frac{\eta}{1-\eta} \frac{c_s n^{\alpha_s}}{u}, \quad u = q^\eta$
The effect of interval size (MSIL)

Graphene 512

# intervals x # processors per interval = 512
Implementation of multiple shift-invert Lanczos

- Place the target shift in the middle of the interval
- Set $k$ to be slight larger than the number of eigenvalues estimated to be in this interval
- Use the implicit restart to limit the size of the Krylov subspace (hence the cost of orthogonalization and Rayleigh-Ritz calculation)
- Set maximum number of restarts to limit the total cost for this interval

![Diagram](image-url)
Implementation of contour integral projection method

- Use the FEAST package (Polizzi)
- In most cases, 16 quadrature points (poles) are sufficient for constructing $P = \sum_i (A - z_i I)^{-1} \omega_i$
- Apply the approximate spectral projector $P$ to an orthonormal basis of a subspace $S$ within a subspace iterative (2-3 iterations often sufficient)

Pick an orthonormal basis $V$ for $S$
While no convergence
- $W \leftarrow PV$
- $V \leftarrow qr(W)$
- Check convergence
- $\dim(S) = 1.5k$
### MSIL vs MCISPM

**MSIL:**
- One factorization per interval
- Real arithmetic
- One solve at a time, a sequential process

**MICISPM:**
- 8-16 factorizations per interval
- Complex arithmetic
- Multiple right-hand sides.
  - However, if the factor is distributed, the solves cannot be performed completely in parallel
  - Some efficiency (2-3x) can be gained from blocking (BLAS3)
Why is MCIPM less efficient than MSIL on distributed-memory systems?

- Because MCISPM requires multiple complex factorizations, one would like to include as many eigenvalues as possible in an interval to amortize the factorization cost
  - Ideally, the number of eigenvalues should be 8 or 16x those in an MSIL
- But having too many eigenvalues in an interval will increase the cost of triangular substitution
- The number of eigenvalues is also limited by Rayleigh-Ritz/orthogonalization cost and convergence rate of the subspace iteration

- The conclusion may be different if each linear system is solved iteratively or if shared-memory parallelism is used
MSIL vs MCISPM

<table>
<thead>
<tr>
<th>problem</th>
<th>$n$</th>
<th>$nev$</th>
<th>$p$</th>
<th>$t_{\text{msil}}$</th>
<th>$t_{\text{mcispm}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphene512</td>
<td>20,480</td>
<td>2,048</td>
<td>128</td>
<td>16</td>
<td>125</td>
</tr>
<tr>
<td>Graphene2048</td>
<td>81,920</td>
<td>8,192</td>
<td>512</td>
<td>31</td>
<td>208</td>
</tr>
<tr>
<td>C60</td>
<td>17,576</td>
<td>1,757</td>
<td>8,192</td>
<td>5.5</td>
<td>46</td>
</tr>
<tr>
<td>SiO</td>
<td>33,401</td>
<td>3,340</td>
<td>2,048</td>
<td>27</td>
<td>98</td>
</tr>
<tr>
<td>GaAsH6</td>
<td>61,349</td>
<td>6,134</td>
<td>16,384</td>
<td>50</td>
<td>198</td>
</tr>
</tbody>
</table>

- Use MUMPS for factorization and triangular solution
- FEAST for MCISPM
- PARPACK for MSIL
- Experiments performed on Hopper at NERSC. Each node has two 12-core AMD Magny Cours 2.1GHz processors, 32GB shared memory
- Convergence tolerance (for relative residual norm) set to $10^{-10}$
Weak and strong scaling study

<table>
<thead>
<tr>
<th>Matrix name</th>
<th>dimension</th>
<th>nnz</th>
<th>$L_{nnz}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphene128</td>
<td>5120</td>
<td>1M</td>
<td>3.1M</td>
</tr>
<tr>
<td>Graphene512</td>
<td>20480</td>
<td>4.1M</td>
<td>43.2M</td>
</tr>
<tr>
<td>Graphene2048</td>
<td>81920</td>
<td>16.4M</td>
<td>135.4M</td>
</tr>
<tr>
<td>Graphene8192</td>
<td>327680</td>
<td>65.7M</td>
<td>727.5M</td>
</tr>
</tbody>
</table>

- Matrices generated from DGDFDT
- Use MUMPS for factorization and triangular solution
- FEAST for MCISPM
- PARPACK for MSIL
- Experiments performed on Hopper at NERSC. Each node has two 12-core AMD Magny Cours 2.1GHz processors, 32GB shared memory
- Convergence tolerance (for relative residual norm) set to $10^{-10}$
MSIL strong scaling
The expected scaling factors are calculated from the cost model and the actual time measured for factorization and triangular solutions, and the assumption that $\eta_f = \eta_s = 0.5$.
MCISPM Strong scaling

- **Graphene2048, \( n = 81,920, nev = 8,192 \)**

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q \times l )</th>
<th>( k )</th>
<th>( t_f )</th>
<th>( t_s )</th>
<th>( t_{\text{other}} )</th>
<th>( t_{\text{wall}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>8,192</td>
<td>256 x 32</td>
<td>256</td>
<td>65</td>
<td>104</td>
<td>37</td>
<td>208</td>
</tr>
<tr>
<td>16,384</td>
<td>256 x 64</td>
<td>128</td>
<td>65</td>
<td>52</td>
<td>18</td>
<td>137</td>
</tr>
<tr>
<td>32,768</td>
<td>256 x 128</td>
<td>64</td>
<td>65</td>
<td>26</td>
<td>18</td>
<td>104</td>
</tr>
<tr>
<td>65,536</td>
<td>256 x 256</td>
<td>32</td>
<td>65</td>
<td>12</td>
<td>27</td>
<td>89</td>
</tr>
<tr>
<td>131,072</td>
<td>256 x 512</td>
<td>16</td>
<td>65</td>
<td>6</td>
<td>12</td>
<td>87</td>
</tr>
<tr>
<td>262,144</td>
<td>256 x 1024</td>
<td>8</td>
<td>65</td>
<td>3.2</td>
<td>10</td>
<td>83</td>
</tr>
</tbody>
</table>

- **Optimal choice of \( q \times l \)**

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q \times l )</th>
<th>( k )</th>
<th>( t_f )</th>
<th>( t_s )</th>
<th>( t_{\text{other}} )</th>
<th>( t_{\text{wall}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>8,192</td>
<td>256 x 32</td>
<td>256</td>
<td>65</td>
<td>104</td>
<td>37</td>
<td>208</td>
</tr>
<tr>
<td>16,384</td>
<td>512 x 32</td>
<td>256</td>
<td>33</td>
<td>68</td>
<td>30</td>
<td>131</td>
</tr>
<tr>
<td>32,768</td>
<td>512 x 64</td>
<td>128</td>
<td>33</td>
<td>36</td>
<td>16</td>
<td>85</td>
</tr>
<tr>
<td>131,072</td>
<td>512 x 256</td>
<td>32</td>
<td>33</td>
<td>9</td>
<td>5</td>
<td>47</td>
</tr>
<tr>
<td>262,144</td>
<td>1024 x 256</td>
<td>16</td>
<td>24</td>
<td>9</td>
<td>4</td>
<td>38</td>
</tr>
</tbody>
</table>
MCISPM weak scaling

<table>
<thead>
<tr>
<th>problem</th>
<th>$p$</th>
<th>$q \times l$</th>
<th>$k$</th>
<th>$t_{wall}$</th>
<th>expect/actual scaling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphene128</td>
<td>512</td>
<td>64 $\times$ 8</td>
<td>64</td>
<td>3.6</td>
<td></td>
</tr>
<tr>
<td>Graphene512</td>
<td>2048</td>
<td>128 $\times$ 16</td>
<td>128</td>
<td>55</td>
<td>5.6/16</td>
</tr>
<tr>
<td>Graphene2048</td>
<td>8192</td>
<td>256 $\times$ 32</td>
<td>256</td>
<td>208</td>
<td>5.6/3.8</td>
</tr>
</tbody>
</table>

$l = \#$ intervals; $q = \#$ processors/interval

- The expected scaling factors are calculated from the cost model and the actual time measured for factorization and triangular solutions, and the assumption that $\eta_f = \eta_s = 0.5$
Conclusion

- Spectrum slicing introduces additional levels of concurrency
- Optimal partition depends on the relative cost of factorization and triangular substitution in addition to several other constraints

- MSIL appears to be more efficient than MCISPM on distributed-memory parallel machines when direct methods are used to factor the matrix and solve triangular systems

- It will be interesting to see how these methods perform when preconditioned iterative solvers are used to solve the linear systems

- Performance on nonsymmetric eigenvalue problems???