

# 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 eignvalue problems

- Computing a few smallest/largest/interior eignvalues
  - 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 eignvalue 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 manybody 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?



□ Contour integral projection method (MCISPM)

$$P = \frac{1}{2\pi i} \oint_{\Gamma} (A - zI)^{-1} dz \approx \sum_{1}^{n_p} \omega_i (A - z_iI)^{-1}$$

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

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





### **Observations**

When c<sub>n</sub> / c<sub>p</sub> is constant, optimal k depends on problem size and relative cost of factorization and triangular solution

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

□ When  $\eta_f, \eta_s$  are close to 1, we should place more eigenvaues 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





## 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







# 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 distributedmemory 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





problem	n	nev	р	<b>t</b> <sub>msil</sub>	<b>t</b> <sub>mcispm</sub>
Graphene512	20,480	2,048	128	16	125
Graphene2048	81,920	8,192	512	31	208
C60	17,576	1,757	8,192	5.5	46
SiO	33,401	3,340	2,048	27	98
GaAsH6	61,349	6,134	16,384	50	198

- Use MUMPS for factorization and triangular solution
- □ FEAST for MCISPM
- PARPACK for MSIL
- Experiments performed on Hopper at NERSC. Each node has two 12core AMD Magny Cours 2.1GHz processors, 32GB shared memory
- □ Convergence tolerance (for relative residual norm) set to 10<sup>-10</sup>





# Weak and strong scaling study

Matrix name	dimension	nnz	L <sub>nnz</sub>
Graphene128	5120	1M	3.1M
Graphene512	20480	4.1M	43.2M
Graphene2048	81920	16.4M	135.4M
Graphene8192	327680	65.7M	727.5M

- Matrices generated from DGDFT
- Use MUMPS for factorization and triangular solution
- □ FEAST for MCISPM
- PARPACK for MSIL
- Experiments performed on Hopper at NERSC. Each node has two 12core AMD Magny Cours 2.1GHz processors, 32GB shared memory
- □ Convergence tolerance (for relative residual norm) set to 10<sup>-10</sup>





### **MSIL strong scaling**

Science



..... BERKELEY LAB

### MSIL weak scaling

problem	P	q x l	k	<b>t</b> <sub>wall</sub>	expect/ actual scaling
Graphene512	512	20 x 25	82	16.4	
Graphene2048	2048	20 x 102	80	50.6	3.2/3.1
Graphene8192	8192	20 x 108	96	353	5.2/7

*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  $\eta_f = \eta_s = 0.5$ 





## **MCISPM Strong scaling**

### Graphene2048, *n* = 81,920, *nev* = 8,192

р	q x l	k	t <sub>f</sub>	t <sub>s</sub>	<b>t</b> <sub>other</sub>	$t_{\sf wall}$
8,192	256 x 32	256	65	104	37	208
16,384	256 x 64	128	65	52	18	137
32,768	256 x 128	64	65	26	18	104
65,536	256 x 256	32	65	12	27	89
131,072	256 x 512	16	65	6	12	87
262,144	256 x 1024	8	65	3.2	10	83

### $\Box$ Optimal choice of $q \ge l$

р	q x l	k	t <sub>f</sub>	t <sub>s</sub>	<b>t</b> <sub>other</sub>	<b>t</b> <sub>wall</sub>
8,192	256 x 32	256	65	104	37	208
16,384	512 x 32	256	33	68	30	131
32,768	512 x 64	128	33	36	16	85
131,072	512 x 256	32	33	9	5	47
262,144	1024 x 256	16	24	9	4	38



Office of Science



# **MCISPM weak scaling**

problem	Р	q x l	k	t <sub>wall</sub>	expect/actual scaling
Graphene128	512	64 x 8	64	3.6	
Graphene512	2048	128 x 16	128	55	5.6/16
Graphene2048	8192	256 x 32	256	208	5.6/3.8

*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 distributedmemory 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???



