A Hybrid Parallel Iterative Solver for Indefinite Systems in Interior Eigenvalue Computations

Jonas Thies[1] Lukas Krämer[2] and Andreas Pieper[3]

PMAA, Lugano, July 4, 2014

[1]German Aerospace Center (DLR)
Simulation and Software Technology
jonas.thies@dlr.de

[2]University of Wuppertal
Applied Computer Science
kraemer@uni-wuppertal.de

[3]University of Greifswald
Theoretical Physics
pieper@physik.uni-greifswald.de

DLR

Outline

Graphene simulation and the FEAST method

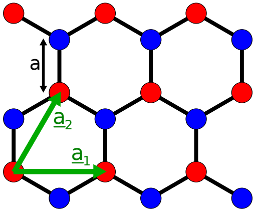
The CGMN algorithm

Parallelization

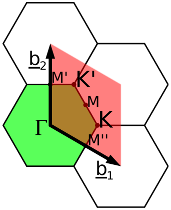
Experiments

Graphene simulation and the FEAST method

Graphene



Physical space: carbon atoms in 2D hexagonal mesh

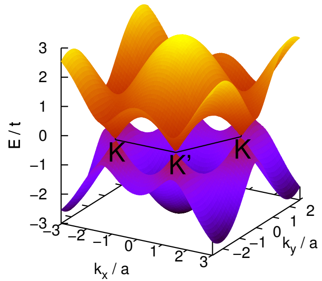


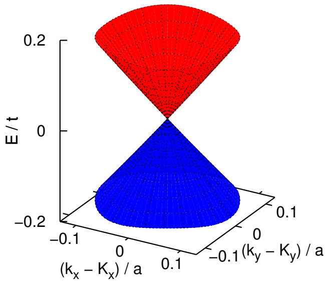
Fourier space ('reciprocal mesh')

Tight-binding Hamiltonian

$$H = -t \sum_{\langle ij \rangle} (c_i^\dagger c_j + c_j^\dagger c_i)$$

Graphene (2)





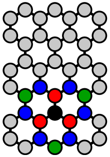
- Analytical solution for infinite Graphene sheet
- Dirac cones: graphene between conducter and semi-conducter

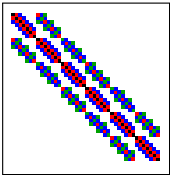
Graphene modelling

- disorder
- long range stencil
- bilayer
- gate-defined quantum dots
- spin-orbit coupling
- ...

Long range Hamiltonian:

$$H = \sum_i V_i c_i^\dagger c_i - t \sum_{\langle ij \rangle} (c_i^\dagger c_j + c_j^\dagger c_i) - t' \sum_{\langle\langle ij \rangle\rangle} (c_i^\dagger c_j + c_j^\dagger c_i) - t'' \sum_{\langle\langle\langle ij \rangle\rangle\rangle} (c_i^\dagger c_j + c_j^\dagger c_i)$$





The FEAST algorithm at a glance

Input: $I_\lambda := \left[\underline{\lambda}, \overline{\lambda}\right]$, an estimate $\widetilde{m}$ of the number of eigenvalues in I_λ.
Output $\hat{m} \leq \widetilde{m}$ eigenpairs with eigenvalue in I_λ.
Perform:

1. Choose $\mathrm{Y} \in \mathbb{C}^{n\times\widetilde{m}}$ of full rank and compute
   $$\mathrm{U} := \frac{1}{2\pi\mathbf{i}}\int_{\mathcal{C}}(z\mathrm{B}-\mathrm{A})^{-1}\mathrm{B}\,\mathrm{d}z\,\mathrm{Y},$$
2. Form $\mathrm{A_U} := \mathrm{U^\star AU}$, $\mathrm{B_U} := \mathrm{U^\star BU}$,
3. Solve size-$\widetilde{m}$ eigenproblem $\mathrm{A_U}\widetilde{\mathrm{W}} = \mathrm{B_U}\widetilde{\mathrm{W}}\tilde{}$,
4. Compute $(\tilde{}, \widetilde{\mathrm{X}} := \mathrm{U}\cdot\widetilde{\mathrm{W}})$,
5. If no convergence: go to Step 1 with $\mathrm{Y} := \widetilde{\mathrm{X}}$.

Linear systems for FEAST/graphene

Tough:

- very large ($N = 10^8 - 10^{14}$)
- complex symmetric and completely indefinite
- random numbers on and around the diagonal
- spectrum essentially continuous
- shifts get very close to the spectrum

But also nice in some ways:

- 2D mesh, very sparse (~ 10 entries/row)
- multiple RHS/shift (block methods, recycling, ...)

We need $\mathcal{O}(100)$ Eigenpairs $\Longrightarrow$ very coputationally heavy...

The CGMN algorithm

An ancient row projection method

- Björck and Elfving, 1979
- CG on the 'minimum norm' problem, $AA^Tx = b$
- preconditioned by SSOR
- efficient row-wise formulation
- extremely robust: A may be singular, non-square etc.
- row scaling aleviates issue of 'squared condition number'

Kernel operation: KACZ sweep

Interpretations:

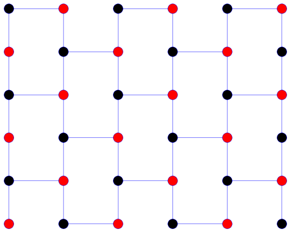
- Kaczmarz algorithm
- SOR(ω) on the normal equations $AA^T x = b$
- successive projections onto the hyperplanes defined by the rows of A

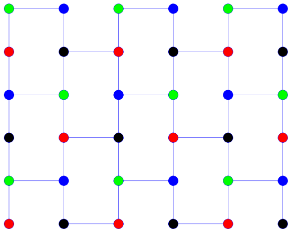
In CRS (rptr,val,col):

```
 1: compute nrms=$||a_{i,:}||_2^2$
 2: for (i=0; i<n; i++) do
    // compute $a_{i,:}x - b_i$
 3:     scal=-b[i]
 4:     for (j=rptr[i]; j<rptr[i+1]; j++) do
 5:         scal+=val[j]*x[col[j]]
 6:     end for
 7:     scal/=nrms[i]
    // update x
 8:     for (j=rptr[i]; j<rptr[i+1]; j++) do
 9:         x[cols[j]]-=omega*scal*val[j]
10:     end for
11: end for
```

Parallelization

Multi-Coloring (MC)





- requires "distance 2" coloring
- software: ColPack
  `http://cscapes.cs.purdue.edu/coloringpage/software.htm`

Component-Averaged Row Projection (CARP)

- Gordon & Gordon, 2005
- Kaczmarz locally
- write to halo
- exchange and average

Equivalent to Kaczmarz on a superspace of $\mathbb{R}^n$

Hybrid method: MC_CARP-CG

- global MC would require...
  - an extremely scalable coloring method
  - very well-balanced colors
  - many global sync-points ($>$ 20 colors in our examples)
- global CARP would require...
  - huge number of MPI procs
  - increasing amount of 'interior halo elements'
  - non-trivial implemention on GPU and Xeon Phi
  - increasing number of iterations

Idea: node-local MC with MPI-based CARP between the nodes

Experiments

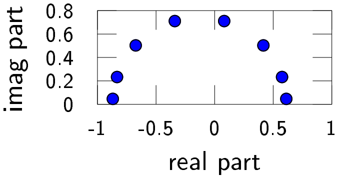
Experimental setup

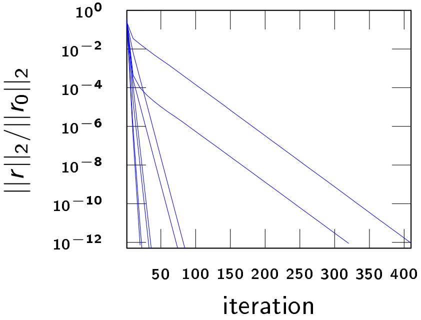
- Machine: Intel Xeon "Ivy Bridge"
- 10 cores/socket, 2 sockets/node
- InfiniBand between nodes

Here's what we do:

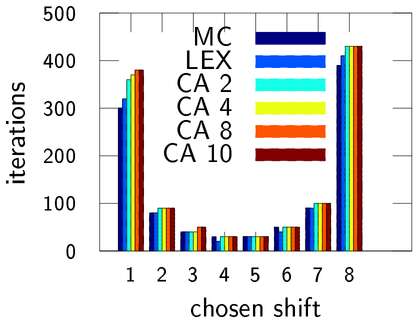
- pick some shifts that may occur in FEAST
- handle 8 rhs at once (for good performance)
- conv tol 10^{-12}
- solve linear systems using CGMN variants

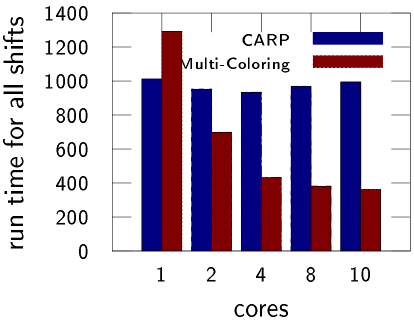
Sequential CGMN for various shifts



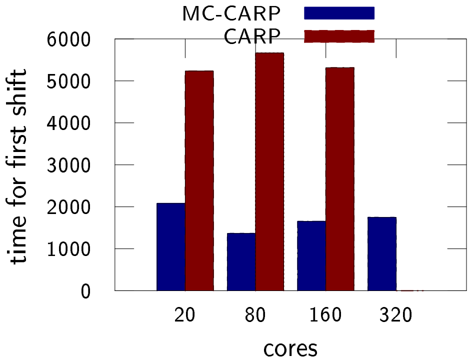


Coloring vs. CARP: single socket (1024^2 dof)





Weak scaling of Hybrid vs. CARP (4096^2 dof/node)



The (almost) final slide

- Graphene gives nice and challenging test cases for Lin. Alg.
- FEAST requires fast linear solvers for indef. systems
- row projection methods work very well here
- hybrid is a natural choice here - and works

Future work:

- integration in FEAST loop
- stencil-based implementation
- GPU and Xeon Phi

Acknoledgement: DFG SPPEXA project ESSEX
(Equipping Sparse Solvers for the EXa-scale)

References

- Pieper et. al.: Effects of disorder and contacts on transport through graphene nanoribbons
  Phys. Rev. B 88, 195409, (2013).
- Polizzi: Density-Matrix-Based Algorithms for Solving Eigenvalue Problems. *Phys. Rev. B.* 79 - 115112 (2009)
- Björck & Elfving: Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations.
  BIT 19, pages 145–163, 1979.
- Gordon & Gordon: Component-averaged row projections: A robust, block-parallel scheme for sparse linear systems.
  SISC 27 (3), 2005, pages 1092–1117.