On Solving Large Eigenvalue Problems from Opto-Electronics

Peter Arbenz[1] Christof Vömel[1] Ratko Veprek[2]

[1]Institute of Computational Science, ETH Zurich,

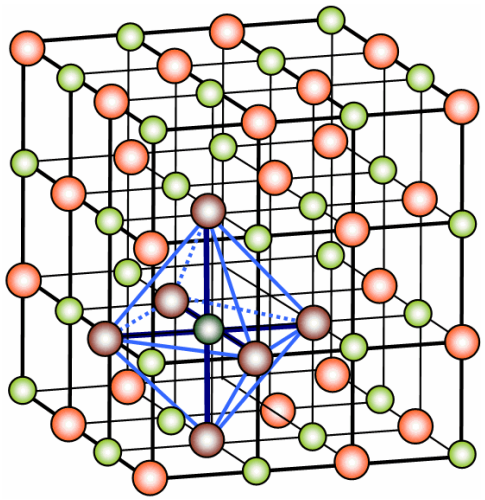
[2]Integrated Systems Laboratory, ETH Zurich

Work in progress

IWASEP, Dubrovnik, Croatia, June 9-12, 2008.

Introduction I

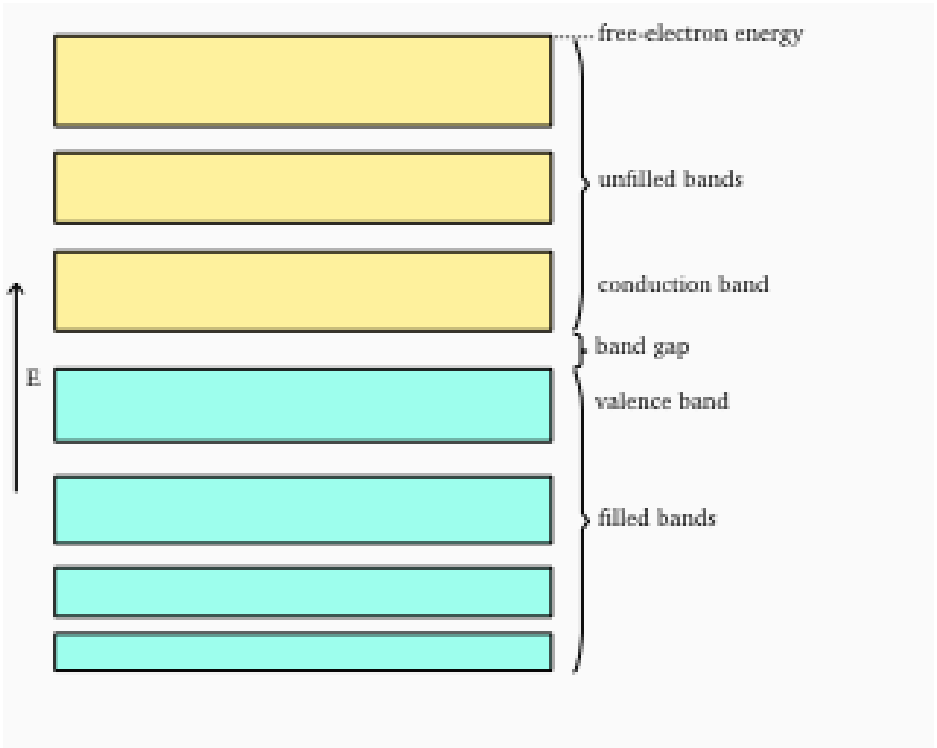
- In solid state physics, the (electronic) band structure of a solid describes ranges of energy that an electron can assume.
- Electrons of single free-standing atom occupy atomic orbitals, that form discrete set of energy levels. When a large number of atoms are brought together to form a solid, the number of orbitals becomes exceedingly large, and the difference in energy between them becomes very small. Bands of energy levels are form rather than discrete energy levels. Intervals that contain no orbitals are called band gaps.



From `wikipedia.org`

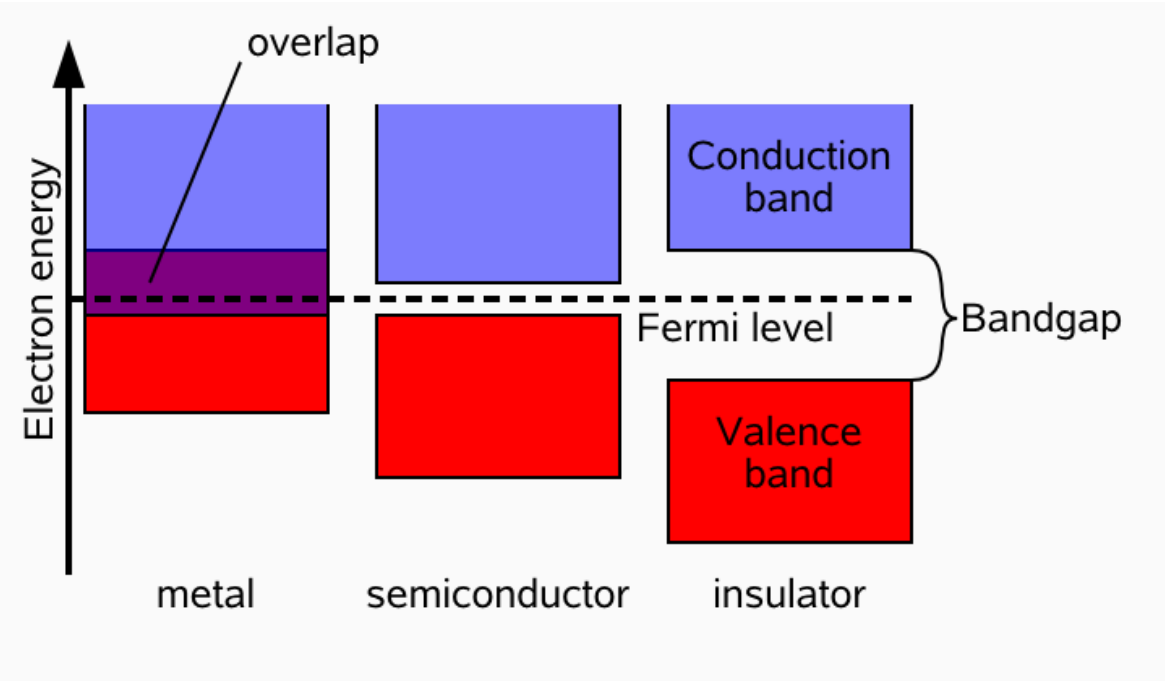
Introduction II

- Any solid has a large number of bands, in theory, ∞ many. However, all but a few lie at energies so high that any electron that reaches those energies escapes from the solid. These bands are usually disregarded.



Introduction III

- The uppermost occupied band is called valence band by analogy to the valence electrons of individual atoms. The lowermost unoccupied band is called the conduction band because only when electrons are excited to the conduction band can current flow in these materials.



Introduction IV

- A numerical evaluation of the band structure takes into account the periodic nature of a crystal lattice.
  The Schrödinger equation for a single particle in a cristal lattice is given by

$$(-\frac{\hbar^2}{2m_0}\nabla^2 + V(\mathbf{r}))\Psi(\mathbf{r}) = E\Psi(\mathbf{r}) \tag{1}$$

  where the potential V exhibits the cristal periodicities:
  $V(\mathbf{r}) = V(\mathbf{r} + \mathbf{R})$ for all lattice vectors $\mathbf{R}$.
  Solutions of (1) are Bloch functions:

$$\Psi_{n\mathbf{k}}(\mathbf{r}) = e^{i\mathbf{k}\cdot\mathbf{r}} u_{n\mathbf{k}}(\mathbf{r}). \tag{2}$$

  where $u_{n\mathbf{k}}(\mathbf{r})$ is lattice-periodic.

Introduction V

Here, **k** is called the wave vector, and is related to the direction of motion of the electron in the crystal, and n is the band index, which simply numbers the energy bands. The wave vector **k** takes on values within the Brillouin zone corresponding to the crystal lattice.
Plugging (2) into (1) yields

$$\frac{\hbar^2}{2m_0}\left(-\nabla^2 + 2i\mathbf{k}\cdot\nabla + |\mathbf{k}|^2 + V(\mathbf{r})\right) u_{n\mathbf{k}}(\mathbf{r}) = E_n(\mathbf{k}) u_{n\mathbf{k}}(\mathbf{r}) \tag{3}$$

For each of the possible **k** there are infinitely many $n = 1, 2, \ldots$.

Introduction VI

- There are a number of possible ways to compute the spectral bands.
  1. Solve (3) for $\mathbf{k} = \mathbf{k}_0 = \mathbf{0}$ and determine the rest of the eigenvalues by perturbation theory. (Effective mass m_*)
  2. Solve (3) in a subspace spanned by a small number of eigenfunctions (zone-centered Bloch functions) $u_{n\mathbf{0}}$ (Kane, 1957).
  3. In the $\mathbf{k}\cdot\mathbf{p}$ method the two methods are somehow combined by writing

$$\Phi(\mathbf{r}) = \sum_{i=1}^{m} g_i(\mathbf{r}) u_{i\mathbf{0}}(\mathbf{r}). \tag{4}$$

     Here, the so-called envelope g_i replaces the plane wave $e^{i\mathbf{k}\cdot\mathbf{r}}$ in (2) (Luttinger-Kohn, 1955; Löwdin, 1951).

We go for the $\mathbf{k}\cdot\mathbf{p}$ method for its superior accuracy.

Introduction VII

- We finally arrive at an eigenvalue problem for the envelopes:

$$\left(\sum_{i,j=1,2,3} \mathbf{H}_{ij}^{(2)}(\mathbf{r})\partial_i\partial_j + \mathbf{H}_i^{(1)}(\mathbf{r})\partial_i + \mathbf{H}^{(0)}(\mathbf{r}) + \mathbf{U}(\mathbf{r}) \right) \mathbf{g}(\mathbf{r}) = E\mathbf{g}(\mathbf{r}) \tag{5}$$

  Here, the perturbation $\mathbf{U}(\mathbf{r})$ takes into account an impurity potential of the material or quantum dot, etc.
  Note, that $\mathbf{g}$ is a m-vector (field).

- For the numerical computation $\mathbf{g}$ is represented by piecewise (linear) finite elements.

The linear algebra problem

We end up with the generalized complex Hermitian eigenvalue problem

$$A\mathbf{x} = \lambda M\mathbf{x}. \tag{6}$$

where

$$A^* = A \in \mathbb{C}^{n\times n}, \qquad M^T = M \in \mathbb{R}^{n\times n},$$

Depending on the bands included, A is either definite or indefinite. The latter holds if valence and conduction band are taken into account.

Just a small number ($k = 1 - 8$) of bands are used. Matrices have $k \times k$ blocks.

Only a *few of the eigenvalues closest to 0* of (6) are desired.

Transformation complex Hermitian $\longrightarrow$ real symmetric I

For $A = A_r + i\,A_i \in \mathbb{C}^{n\times m}$, $A_r, A_i \in \mathbb{R}^{n\times m}$, define the mapping (Day & Heroux, 2001)

$$\varphi : \mathbb{C}^{m\times n} \longrightarrow \mathbb{R}^{2m\times 2n} : A \longmapsto \varphi(A) := \begin{pmatrix} A_r & -A_i \\ A_i & A_r \end{pmatrix} \tag{7}$$

that transforms a complex $m \times n$ matrix into a real $2m \times 2n$ matrix. If operations are allowed, then

$$\varphi(AB) = \varphi(A)\varphi(B). \tag{8}$$

- Many codes for solving eigenvalue problems, in particular for large sparse eigenvalue problems, are available only in real arithmetic.
- Very often the sparsity structures of real and imaginary parts of the underlying matrices differ considerably.

Transformation complex Hermitian $\longrightarrow$ real symmetric II

We can rewrite the complex eigenvalue problem

$$A\mathbf{x} = \mathbf{x}\lambda \tag{9}$$

in the real form

$$\varphi(A\mathbf{x}) = \varphi(A)\varphi(\mathbf{x}) = \varphi(\mathbf{x}\lambda) = \varphi(\mathbf{x})\varphi(\lambda). \tag{10}$$

or,

$$\begin{pmatrix} A_r & -A_i \\ A_i & A_r \end{pmatrix} \begin{pmatrix} \mathbf{x}_r & -\mathbf{x}_i \\ \mathbf{x}_i & \mathbf{x}_r \end{pmatrix} = \begin{pmatrix} \mathbf{x}_r & -\mathbf{x}_i \\ \mathbf{x}_i & \mathbf{x}_r \end{pmatrix} \begin{pmatrix} \lambda_r & -\lambda_i \\ \lambda_i & \lambda_r \end{pmatrix}. \tag{11}$$

So, a complex eigenvalue of A becomes a pair of complex conjugate eigenvalues of $\varphi(A)$.

A Hermitian $\Longrightarrow \lambda_i = 0$.

Transformation complex Hermitian $\rightarrow$ real symmetric III

Special cases:

- $$\varphi(\mathbf{x}^*)\varphi(\mathbf{x}) = \varphi(\|\mathbf{x}\|^2) \iff \begin{pmatrix} \mathbf{x}_r^T & \mathbf{x}_i^T \\ -\mathbf{x}_i^T & \mathbf{x}_r^T \end{pmatrix} \begin{pmatrix} \mathbf{x}_r & -\mathbf{x}_i \\ \mathbf{x}_i & \mathbf{x}_r \end{pmatrix} = \|\mathbf{x}\|^2 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$
- Enforcing $\mathbf{x}^*\mathbf{y} = 0$ thus means that $\begin{pmatrix} \mathbf{y}_r \\ \mathbf{y}_i \end{pmatrix}$ is made orthogonal to both $\begin{pmatrix} \mathbf{x}_i \\ \mathbf{x}_r \end{pmatrix}$ and $\begin{pmatrix} \mathbf{x}_r \\ -\mathbf{x}_i \end{pmatrix}$. The companion vector $\begin{pmatrix} -\mathbf{y}_i \\ \mathbf{y}_r \end{pmatrix}$ will then automatically be orthogonal to these two vectors.

  Computation is done with just one vector; orthogonality is forced against two.

Transformation complex Hermitian $\longrightarrow$ real symmetric IV

Rayleigh-Ritz procedure

- Given a search space $\mathcal{R}(V_k)$, $V_k^T M V_k = I$, determining Ritz pairs amounts to determining eigenpairs of $V_k^T A V_k$:

$$\begin{pmatrix} V_r & -V_i \\ V_i & V_r \end{pmatrix}^T \begin{pmatrix} A_r & A_i^T \\ A_i & A_r \end{pmatrix} \begin{pmatrix} V_r & -V_i \\ V_i & V_r \end{pmatrix} = \begin{pmatrix} \hat{A}_r & \hat{A}_i^T \\ \hat{A}_i & \hat{A}_r \end{pmatrix}$$

- The 2×2 block matrix on the right can be orthogonally transformed in the direct sum of two identical symmetric tridiagonal matrices (LAPACK subroutine `zlarfg`). Instead of Householder reflectors unitary matrices of the form

$$I - \nu \mathbf{u}\mathbf{u}^*, \qquad |\nu - 1| = 1,$$

need to be employed.

Computing only a few eigenpairs

Case 1. A is positive definite. Compute eigenvalues one by one starting with the smallest.

Case 2: A is indefinite. (M is always spd.)

- SI-Lanczos if the problem is not too big.
- Generate two new positive definite eigenvalue problems

$$(A - \sigma_i M)M^{-1}(A - \sigma_i M)\mathbf{x} = \lambda M\mathbf{x}, \qquad i = 1, 2. \qquad (12)$$

  where σ_1 is close to the smallest positive eigenvalue of (A, M) and σ_2 is close to the smallest negative eigenvalue of (A, M) (Vömel, 2007; Shi Shu, 2006).
- Compute eigenvalues of (12) with shift σ_1 one by one starting with the smallest.
- Compute eigenvalues of (12) with shift σ_2 one by one starting with the smallest.
- Preconditioner? AMG preconditioner?

Restarted Shift-and-Invert Lanczos (SI-Lanczos)

- Choose shift σ close to desired eigenvalues.
- Transform $A\mathbf{x} = \lambda M\mathbf{x}$ into

$$(A - \sigma M)^{-1} M\mathbf{x} = \mu\mathbf{x}, \qquad \mu = \frac{1}{\lambda - \sigma}. \tag{13}$$

- Apply Lanczos algorithm, i.e., construct ON-basis of Krylov space

$$\mathcal{K}_k((A - \sigma M)^{-1} M, \mathbf{v}_0) = \text{span}\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k\} = \mathcal{R}(V_k).$$

- Typical (ARPACK) approach:
  - Full reorthogonalization to avoid loss of orthogonality.
  - Limitation of dimension of Krylov spaces to avoid excessive memory consumption.
  - Entails restarting procedure.

Remarks on SI-Lanczos

- Shift-and-Invert emphasizes the eigenvalues close to the shift. Large relative gaps between eigenvalues speed up convergence.
- Restarting makes full reorthogonalization feasible.
- Main problem: System solve with $A - \sigma M$.
  - LU - factorization if problem is small (2D)
  - Iterative solve requires high accuracy if 3-term recurrence shall hold.

Davidson

(Davidson, 1975)

- Let $\mathcal{V}_k = \text{span}\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$, $\mathbf{v}_k^T M \mathbf{v}_j = \delta_{kj}$, be the actual search space (**not** a Krylov space).
- Rayleigh–Ritz–Galerkin procedure: Extract Ritz pair $(\tilde{\lambda}, \tilde{\mathbf{q}})$ in $\mathcal{V}_k$ with $\tilde{\lambda}$ closest to some target value τ.
- Convergence: If $\|\tilde{\mathbf{r}}_k\|_{M^{-1}} \equiv \|(A - \tilde{\lambda} M)\,\tilde{\mathbf{q}}\|_{M^{-1}} < \varepsilon \|\tilde{\mathbf{q}}\|_M$ then we have found an eigenpair
- Otherwise, solve for $\mathbf{t}_k$,

$$K \mathbf{t}_k = -\tilde{\mathbf{r}}_k, \qquad K \approx A - \tau M. \tag{14}$$

  K is called a preconditioner for $A - \tau M$.
- M-orthonormalize $\mathbf{t}_k$ to $\mathcal{V}_k$ to obtain $\mathbf{v}_{k+1}$
- Expand search space: $\mathcal{V}_{k+1} = \text{span}\{\mathbf{v}_1, \ldots, \mathbf{v}_{k+1}\}$.

Remarks on Davidson

- Principle problem with the "preconditioner": must not be too good. If $K = A - \tau M$ we get back the residual.
- $K = A - \tau M$ has bad condition if τ is close to $\sigma(A; B)$.
- The method was found to be successful in many instances, in particular with diagonally dominant problems.
- Eigenvectors corresponding to higher eigenvalues are computed in the orthogonal complement of previously computed eigenvectors.
- To keep the subspace size bounded: If $k = j_{\max}$ reduce size of the search space to $j_{\min}$. Use $j_{\min}$ 'best' Ritz vectors in $\mathcal{V}_{j_{\max}}$ to define $\mathcal{V}_{j_{\min}}$.

Symmetric Jacobi–Davidson (JDSYM)

(Sleijpen/van der Vorst, 1996; Geus, 2003)

- Let $\mathcal{V}_k = \operatorname{span}\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$, $\mathbf{v}_k^T M \mathbf{v}_j = \delta_{kj}$, be the actual search space (**not** a Krylov space).
- Rayleigh–Ritz–Galerkin procedure: Extract Ritz pair $(\tilde{\lambda}, \tilde{\mathbf{q}})$ in $\mathcal{V}_k$ with $\tilde{\lambda}$ closest to some target value τ.
- Convergence: If $\|\mathbf{r}_k\|_{M^{-1}} \equiv \|(A - \tilde{\lambda} M)\,\tilde{\mathbf{q}}\|_{M^{-1}} < \varepsilon \|\tilde{\mathbf{q}}\|_M$ then we have found an eigenpair
- Solve correction equation for $\mathbf{t}_k \perp_M \tilde{\mathbf{q}}$,

$$(I - M\tilde{\mathbf{q}}\tilde{\mathbf{q}}^T)(A - \eta_k M)(I - \tilde{\mathbf{q}}\tilde{\mathbf{q}}^T M)\mathbf{t}_k = -\mathbf{r}_k, \quad \tilde{\mathbf{q}}^T M \mathbf{t}_k = 0. \tag{15}$$

- M-orthonormalize $\mathbf{t}_k$ to $\mathcal{V}_k$ to obtain $\mathbf{v}_{k+1}$
- Expand search space: $\mathcal{V}_{k+1} = \operatorname{span}\{\mathbf{v}_1, \ldots, \mathbf{v}_{k+1}\}$.

Remarks on JDSYM

- In exact arithmetic, 'one-vector JD' is Rayleigh quotient iteration. Eigenvalue approximations converge cubically.
- Stabilization: Shift η_k in (15) is set to target value τ initially ($\sim$ inverse iteration) and to the Rayleigh quotient $\rho(\tilde{\mathbf{q}})$ close to convergence.
- Keep subspace size bounded: If $k = j_{\max}$ reduce size of the search space to $j_{\min}$. Use $j_{\min}$ 'best' Ritz vectors in $\mathcal{V}_{j_{\max}}$ to define $\mathcal{V}_{j_{\min}}$.
- The correction equation is solved only approximatively. We use a Krylov space method: QMRS (admits indefinite preconditioner) (Freund, 1992).
- Eigenvectors corresponding to higher eigenvalues are computed in the orthogonal complement of previously computed eigenvectors.

Locally Optimal Block Preconditioned Conjugate Gradient Method (LOBPCG)

- Preconditioned Conjugate Gradient Method
  Minimize the Rayleigh quotient

  $$\rho(\mathbf{x}_{k+1}) = \rho(\mathbf{x}_k + \delta_k \mathbf{p}_k), \qquad \mathbf{p}_k = -K^{-1}\mathbf{g}_k + \alpha_k \mathbf{p}_{k-1} \perp_A \mathbf{p}_{k-1}.$$

  $\mathbf{g}_k = \nabla \rho(\mathbf{x}_k)$.
- Locally optimal Conjugate Gradient Method (Knyazev, 2001)

  $$\rho(\mathbf{x}_{k+1}) = \min_{\delta_k, \gamma_k} \rho(\mathbf{x}_k - \delta_k K^{-1}\mathbf{g}_k + \gamma_k \mathbf{p}_{k-1})$$

  One degree of freedom more $\Longrightarrow$ faster convergence.
- Block version (LOBPCG): $X_{k+1} \in \mathbb{C}^{n \times p}$ is determined to contain the p 'smallest' Ritz vectors of

  $$\rho(\mathbf{X}_{k+1}) = \rho([\mathbf{X}_k, K^{-1}\mathbf{G}_k, \mathbf{P}_{k-1}]).$$

Remarks on LOBPCG

- All eigenpairs are computed simultaneously.
  However some come earlier than others: Locking of converged vectors.
- Computing $K^{-1}G_k$ completely parallel.
- Preconditioning is more important than so-called local optimality.
- Columns of $[\mathbf{X}_k, K^{-1}\mathbf{G}_k, \mathbf{P}_{k-1}]$ may become linearly dependent.
  Remedy: Restart with random vectors in the orthogonal complement of the already computed eigenvectors.

Preconditioning the correction equation

The correction equation is given by

$$(I - M\tilde{\mathbf{q}}\tilde{\mathbf{q}}^T)(A - \eta_k M)(I - \tilde{\mathbf{q}}\tilde{\mathbf{q}}^T M)\mathbf{t}_k = -\mathbf{r}_k, \qquad \tilde{\mathbf{q}}^T M\mathbf{t}_k = 0.$$

Preconditioning means solving with a system of the form

$$(I - M\tilde{\mathbf{q}}\tilde{\mathbf{q}}^T)K(I - \tilde{\mathbf{q}}\tilde{\mathbf{q}}^T M)\mathbf{c} = \mathbf{b}, \qquad \tilde{\mathbf{q}}^T M\mathbf{c} = 0. \qquad (16)$$

where K is a preconditioner for $A - \rho_k M$.

As we are looking for just a few of the smallest eigenvalues we take $K \approx A - \tau M$ where τ is our target close to the desired eigenvalues.

Preconditioners

- Incomplete Cholesky/LU factorization preconditioners.

$$K = L_K U_K^* \approx A - \tau M.$$

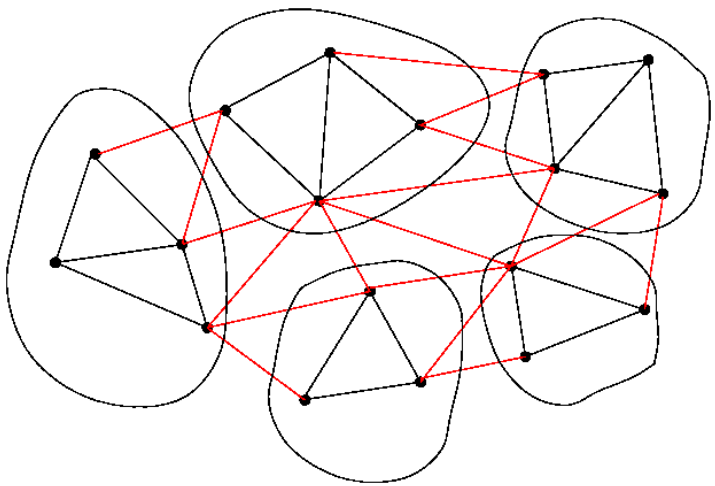
  - ILU(0) transfers the nonzero structure of A to L and U.
  - The parallel version of the IFPACK ILU(0) performs the factorization only locally.
- Algebraic Multigrid (AMG) based on Smoothed Aggregation (SA).

Setup procedure for an abstract multigrid solver

1: Define the number of levels, L
2: **for** level $\ell = 0, \ldots, L-1$ **do**
3: **if** $\ell < L-1$ **then**
4: Define prolongator P_ℓ;
5: Define restriction $R_\ell = P_\ell^T$;
6: $K_{\ell+1} = R_\ell K_\ell P_\ell$;
7: Define smoother S_ℓ;
8: **else**
9: Prepare for solving with K_ℓ;
10: **end if**
11: **end for**

Smoothed aggregation (SA) AMG preconditioner I

1. Build adjacency graph $\mathcal{G}_0$ of $K_0 = K$.
   (Take $m \times m$ block structure into account.)
2. Group graph vertices into contiguous subsets, called *aggregates*. Each aggregate represents a coarser grid vertex.
   - Typical aggregates: $3 \times 3 \times 3$ nodes (of the graph) up to $5 \times 5 \times 5$ nodes (if aggressive coarsening is used)
   - (Par)METIS
   - Note: The matrices $K_1, K_2, \ldots$ need much less memory space than K_0!
     Typical operator complexity for SA: 1.4 (!!!)



Smoothed aggregation (SA) AMG preconditioner II

3. Define a grid transfer operator:
   - Low-energy modes (near-kernel) are 'chopped' according to aggregation

$$B_\ell = \begin{bmatrix} B_1^{(\ell)} \\ \vdots \\ B_{n_{\ell+1}}^{(\ell)} \end{bmatrix}$$

$B_j^{(\ell)}$ = rows of B_ℓ corresponding to grid points assigned to j^{th} aggregate.

   - Let $B_j^{(\ell)} = Q_j^{(\ell)} R_j^{(\ell)}$ be QR factorization of $B_j^{(\ell)}$ then

$$B_\ell = \widetilde{P}_\ell B_{\ell+1}, \qquad \widetilde{P}_\ell^T \widetilde{P}_\ell = I,$$

with

$$\widetilde{P}_\ell = \text{diag}(Q_1^{(\ell)}, \ldots, Q_{n_{\ell+1}}^{(\ell)}) \quad \text{and} \quad B_{\ell+1} = \begin{bmatrix} R_1^{(\ell)} \\ \vdots \\ R_{n_{\ell+1}}^{(\ell)} \end{bmatrix}.$$

Columns of $B_{\ell+1}$ span the near kernel of $K_{\ell+1}$.

   - Notice: matrices K_ℓ are *not* used in constructing tentative prolongators $\widetilde{P}_\ell$, near kernels B_ℓ, and graphs $\mathcal{G}_\ell$.

Smoothed aggregation (SA) AMG preconditioner III

4. For elliptic problems, it is advisable to perform an additional step, to obtain *smoothed aggregation* (SA).

$$P_\ell = (I_\ell - \omega_\ell D_\ell^{-1} K_\ell)\,\widetilde{P}_\ell, \qquad \omega_\ell = \frac{4/3}{\lambda_{\max}(D_\ell^{-1} K_\ell)},$$

smoothed prolongator

In *non-smoothed* aggregation: $P_\ell = \widetilde{P}_\ell$

5. Smoother S_ℓ: polynomial smoother
   - Choose a Chebyshev polynomial that is small on the upper part of the spectrum of K_ℓ (Adams, Brezina, Hu, Tuminaro, 2003).
   - Parallelizes perfectly, quality independent of processor number.

'Matrix-free' multigrid

- Our approach: pcg which "almost" smoothed aggregation AMG preconditioning
- We set $K = K_0 = A - \sigma M$ in the positive definite case and $K = (A - \sigma M) M_{\text{lumped}}^{-1} (A - \sigma M)$ in the indefinite case. (We apply K_0.)

  We lump because M^{-1} is dense. $\mathcal{G}_0$ is the graph corresponding to $(A - \sigma M) M_{\text{lumped}}^{-1} (A - \sigma M)$.
- P_0 is not smoothed, i.e. $P_0 = \widetilde{P}_0$.
- $K_1 = P_0^T K_0 P_0$ is formed explicitely.
- All graphs, including $\mathcal{G}_0$, are constructed.

The Software Environment: Trilinos

- The Trilinos Project is an effort to develop parallel solver algorithms and libraries within an object-oriented software framework for the solution of large-scale, complex multi-physics engineering and scientific applications.
- See http://software.sandia.gov/trilinos/
- Provides means to distribute (multi)vectors and (sparse) matrices (Epetra and EpetraExt packages).
- Provides solvers that work on these distributed data. Here we use iterative solvers and incomplete factorization preconditioners (packages AztecOO/IFPACK), smoothed aggregation multilevel AMG preconditioner (ML), direct solver wrappers (Amesos) and data distribution for parallelization (Zoltan/ParMETIS).

Problem sizes

Problem 1D quantum wire

4x4 n = 1532, nnz = 18352, 2*n = 3064 spd

6x6 n = 2297, nnz = 41292, 2*n = 4596 spd

8x8 n = 3064, nnz = 73408, 2*n = 6128 sym. indef.

Problem 2D quantum wire

4x4 n = 12764, nnz = 447216, 2*n = 25528 spd

6x6 n = 19146, nnz=1006236, 2*n = 38292 spd

8x8 n = 25528, nnz=1788864, 2*n = 51056 sym. indef.

Problem 2D big quantum wire

4x4 n = 49860, nnz=1770384, 2*n = 99720 spd

6x6 n = 74790, nnz=3983364, 2*n = 149580 spd

8x8 n = 99720, nnz=4868556, 2*n = 199440 sym. indef.

Problem sizes

- Number of eigenvalues: 4
- Subspace dimensions
  - IRA: $20 = 5 \times \text{nev}$
  - Davidson: 20
  - JDBSYM: 20 (restart: 8)
  - LOBPCG: $4 = \text{nev}$
- Convergence criterion:

$$\|\mathbf{r}_k\|_{M^{-1}} < \varepsilon \|\mathbf{x}_k\|_M, \qquad \varepsilon = 10^{-6}$$

- Reordering:
  - Mumps: METIS
  - Umfpack: AMD

Numerical results

Solver	Criterion	1D Wire			2D Wire			2D Wire(big)		
		4-band	6-band	8-band	4-band	6-band	8-band	4-band	6-band	8-band
Shift-Invert	Time [sec]	0.4	0.9	0.96	17.25	42.97	75.47	79.40	166.94	291.57
Lanczos	OpVecs	52	52	52	68	68	84	68	68	68
(Umfpack)	Fact.[sec]	0.05	0.1	0.17	7.35	23.75	54.98	55.83	182.46	441.66
Shift-Invert	Time [sec]	0.29	0.37	0.48	5.55	10.44	17.25	25.92	49.74	101.44
Lanczos	OpVecs	52	52	52	68	68	68	68	68	84
(Mumps)	Fact.[sec]	0.03	0.08	0.11	5.95	21.23	54.11	51.99	181.26	421.77
J.-Davidson	Time [sec]	3.17	6.53		43.76	146.65		400.48	752.29	
JDBSYM	Outer It.	33	32		25	34		34	35	
(ML)	Inner It.	16.4	14.3		11.2	8.9		14.1	13.4	
	Prec.[sec]	0.05	0.05		0.55	1.33		2.41	5.16	
J.-Davidson	Time [sec]	0.19	0.35	0.38	10.03	17.93	43.30	66.05	148.79	395.04
JDBSYM	Outer It.	16	18	19	23	22	35	38	41	52
(Ifpack)	Inner It.	1.1	1.1	1.3	10.8	10.2	9.2	10.6	11.4	17.3
	Prec.[sec]	0.06	0.15	0.17	1.88	5.46	11.98	7.56	22.03	43.89
Davidson	Time [sec]	12.54	29.82		56.05	140.46		643.65	1748.6	
(ML)	MatVecs	4228	4404		836	1044		2300	3132	
Davidson	Time [sec]	0.16	0.12		10.35	18.08		129.06	237.74	
(Ifpack)	MatVecs	84	84		732	692		2108	2132	
LOBPCG	Time [sec]	3.28	11.39		29.82	67.69		222.08	555.32	
(ML)	MatVecs	452	816		212	248		384	496	
LOBPCG	Time [sec]	0.2	0.18		7.51	17.01		62.87	109.63	
(Ifpack)	MatVecs	48	48		220	280		392	408	

Remarks on numerical results

- With SI-Lanczos, factorization with UMFPACK and MUMPS take about the same time. MUMPS is a much more effective solver.
- Same preconditioners are used for the Jacobi-Davidson, Davidson, and LOBPCG algorithms, the construction time is only reported once.
- Overall time for LOBPCG is with Incomplete Factorization preconditioner performes best if it converges at all. Not suited for interior eigenvalues.
- The ML preconditioner cannot be applied to the indefinite 8-band problems directly. Folding does not work (yet) and may be expensive anyway.
- All problems are so small that the admit factorization of $A - \sigma M$. SI-Lanczos may win if more eigenvalues are desired.

Parallelization of SI-Lanczos with MUMPS solver

Solver	Criterion	2D Wire			2D Wire(big)		
		4-band	6-band	8-band	4-band	6-band	8-band
1-processor runs							
Shift-Invert	Time [sec]	5.55	10.44	17.25	25.92	49.74	101.44
Lanczos	OpVecs	68	68	68	68	68	84
(Mumps)	Fact.[sec]	5.95	21.23	54.11	51.99	181.26	421.77
2-processor runs							
Shift-Invert	Time [sec]	4.02	6.52	10.54	16.71	30.94	61.58
Lanczos	OpVecs	68	68	68	68	68	68
(Mumps)	Fact.[sec]	4.30	13.18	33.08	31.38	106.89	229.50
4-processor runs							
Shift-Invert	Time [sec]	4.04	7.04	10.57	11.58	25.23	30.32
Lanczos	OpVecs	68	68	68	68	84	68
(Mumps)	Fact.[sec]	3.22	8.25	17.52	16.46	51.62	117.20

Parallelization of JDBSYM with IFPACK preconditioner

Solver	Criterion	2D Wire			2D Wire(big)		
		4-band	6-band	8-band	4-band	6-band	8-band
1-processor runs							
J.-Davidson JDBSYM (Ifpack)	Time [sec]	10.03	17.93	43.30	66.05	148.79	395.04
	Outer It.	23	22	35	38	41	52
	Inner It.	10.8	10.2	9.2	10.6	11.4	17.3
	Prec.[sec]	1.88	5.46	11.98	7.56	22.03	43.89
2-processor runs							
J.-Davidson JDBSYM (Ifpack)	Time [sec]	10.18	21.28	29.53	62.41	125.31	402.28
	Outer It.	32	30	35	40	41	55
	Inner It.	17.9	20.9	14.7	19.2	22.6	21.0
	Prec.[sec]	0.98	2.65	5.97	3.75	10.76	21.76
4-processor runs							
J.-Davidson JDBSYM (Ifpack)	Time [sec]	7.28	14.95	31.30	38.85	113.58	156.24
	Outer It.	32	30	36	42	43	51
	Inner It.	21.9	22.9	17.9	24.3	25.0	20.7
	Prec.[sec]	0.44	1.26	3.41	1.80	5.58	10.56

Parallelization of LOBPCG with IFPACK preconditioner

Solver	Criterion	2D Wire			2D Wire(big)		
		4-band	6-band	8-band	4-band	6-band	8-band
1-processor runs							
LOBPCG	Time [sec]	7.51	17.01		62.87	109.63	
(Ifpack)	MatVecs	220	280		392	408	
	Prec.[sec]	1.88	5.46	11.98	7.56	22.03	43.89
2-processor runs							
LOBPCG	Time [sec]	10.20	15.59		70.12	102.64	
(Ifpack)	MatVecs	436	536		992	776	
	Prec.[sec]	0.98	2.65	5.97	3.75	10.76	21.76
4-processor runs							
LOBPCG	Time [sec]	5.21	9.94		49.76	66.94	
(Ifpack)	MatVecs	520	608		1172	900	
	Prec.[sec]	0.44	1.26	3.41	1.80	5.58	10.56

Conclusions

- SI-Lanczos is an efficient eigensolver. The time for computing the factorisation may be much larger than the actual computation of the desired eigenvalues.
- 'Iterative' eigensolvers outperform SI-Lanczos if only very few eigenpairs are sought.
- LOBPCG is more efficient than JDBSYM for spd problems.
- JD also solves indefinite problems
- The SA multilevel preconditioner is slower than ILU for the problem sizes treated here.
- The SA multilevel preconditioner does not work satisfactory, that is, it exhibits large iteration counts.
- Is spectrum folding a reasonable means to solve indefinite problems?