Dense matrices

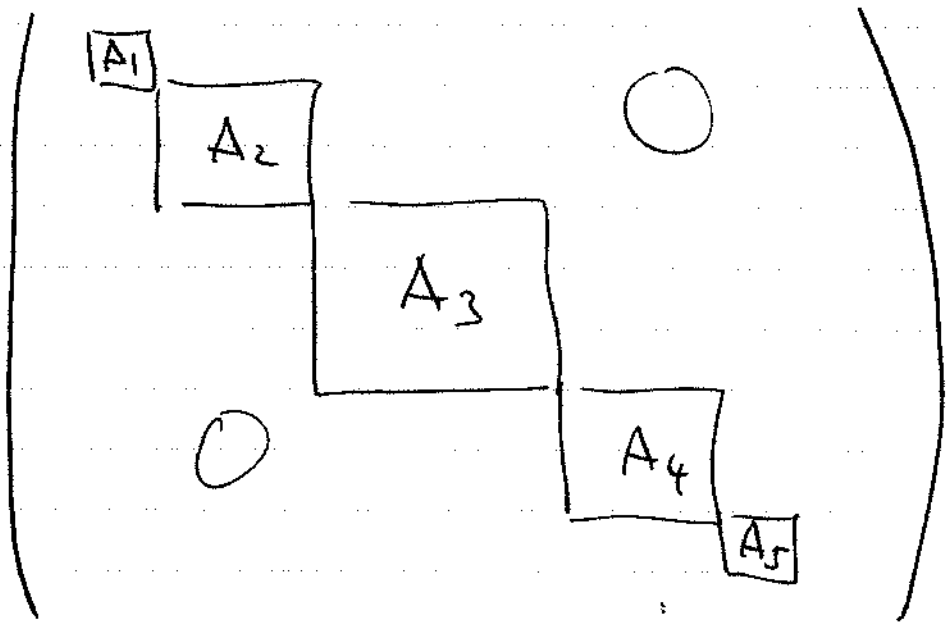
- Householder algorithm to bring A to tridiagonal form T

- then use QR or bi-section or ... to get $\{\lambda_i\}$, $\{\underline{x}_i\}$ eigen-system

- well implemented in LAPACK, NAG, IMSL, ... MKL, ...

→ all eigenvals + _vecs !

- use symmetry in A to first write as



then use LAPACK etc for A_i's!

Sparse matrices

- Matrix A contains many zeros, hence storage and arithmetic with "0"s should be avoided for memory and computing efficiency
  (how many? Enough to make a difference!)
- many storage formats exist (CRS, CCS, ...), see wiki-pedia
- implementation of matrix-vector multiplication is important,

$$(A\,\underline{X})_i = \sum_j A_{ij} X_j$$

$$= \sum_{\substack{ij \text{ s.t.} \\ A_{ij} \neq 0}} A_{ij} X_j$$

- hence structure of matrix can be important.
- $\{\underline{X}_i\}$ takes as much memory as dense storage of A → best to avoid!

Power series method:

$\underline{X}_0$ arbitrary

$\underline{X}_i$, $i = 1, \dots, n$ eigenvectors of A

λ_i, $i = 1, \dots, n$ eigenvalues of A

$$\hookrightarrow X_0 = \sum a_i \underline{X}_i$$

$$A\underline{X}_0 = \sum_i a_i A\underline{X}_i$$

$$= \sum_i a_i \lambda_i \underline{X}_i$$

$$= \lambda_{max} \sum_i a_i \underbrace{\frac{\lambda_i}{\lambda_{max}}}_{\leq 1} \underline{X}_i ,$$

where $\lambda_{max} = \{\lambda_{i_0} \mid \lambda_{i_0} \geq |\lambda_i| \forall i\}$

$$A^n \underline{X}_0 = (\lambda_{max})^n \sum_i a_i \left(\frac{\lambda_i}{\lambda_{max}}\right)^n \underline{X}_i$$

$$\xrightarrow[n \to \infty]{} (\lambda_{max})^n \underline{X}_{i_0}$$

$(i_0 \equiv max)$

↳ $A^n \underline{x}_0 \to (\lambda_{i_0})^n \underline{x}_{i_0}$

to eigenvector corresponding to largest eigenvalue.

- setting all Eval/vecs:

$\{\underline{x}_0, \underline{x}_1, \underline{x}_2, \dots, \underline{x}_{n-1}\}$, $\underline{x}_i \perp \underline{x}_j$

↳ $A^n \underline{x}_0 \to (\lambda_{max})^n \underline{x}_{max}$

$A^n \underline{x}_1 \to (\lambda_{max})^n \underline{x}_{max}$ ↯

↳ keep $\{\underline{x}_0, \dots, \underline{x}_{n-1}\} \perp$, i.e. reorthogonalize

$\underline{x}_1 \perp \underline{x}_0$

$\underline{x}_2 \perp \underline{x}_1, \underline{x}_2$

$\underline{x}_3 \perp \underline{x}_2, \underline{x}_1, \underline{x}_0$

$\underline{x}_j \perp \underline{x}_0, \underline{x}_1, \dots, \underline{x}_{j-1}$

$$\rightarrow \quad A^n \underline{x}_0 \rightarrow \left(\lambda_{max}\right)^n \underline{x}_{max}$$

$$A^n \underline{x}_1 \rightarrow \left(\lambda_{\substack{2nd \\ largest}}\right)^n \underline{x}_{\substack{2nd \\ largest}}$$

$$\vdots \qquad\qquad \vdots$$

$$A^n \underline{x}_{n-1} \rightarrow \left(\lambda_{smallest}\right)^n \underline{x}_{smallest}$$

- basis of all iterative methods, but seldom used

Lanczos method

- Construct sym. tridiagonal matrix

$$T_k = V_k^T A V_k ,$$

$$V_k = \{ \underline{v}_1, \underline{v}_2, \dots, \underline{v}_k \}$$

- $\underline{v}_k$ are constructed via a recursion (iteration):

$$\beta_{i+1} \underline{v}_{i+1} = A \underline{v}_i - \alpha_i \underline{v}_i - \beta_i \underline{v}_{i-1}$$

$$\alpha_i = \underline{v}_i^T A \underline{v}_i$$

$$\beta_{i+1} = \underline{v}_{i+1} A \underline{v}_i$$

$$(\underline{v}_0 = 0 , \quad \underline{v}_1 = \text{arbitrary})$$

- In principle, $\underline{v}_j \perp \underline{v}_i \quad \forall\, i \neq j$

(and hence $\beta_{i+1} = \| A \underline{v}_i - \alpha_i \underline{v}_i - \beta_i \underline{v}_{i-1} \|$)

and

Krylov space $\{\underline{v}_1, \underline{v}_2, \ldots, \underline{v}_k\}$ is complete basis with $k = n$.

$$T_k = \begin{pmatrix} \alpha_1 & \beta_2 & 0 & \cdots & 0 & 0 \\ \beta_2 & \alpha_2 & \beta_3 & & \vdots & \vdots \\ 0 & \beta_3 & \alpha_3 & \beta_4 & \vdots & \vdots \\ \vdots & & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & & & \alpha_{n-1} & \beta_n \\ 0 & \cdots & & & \beta_n & \alpha_n \end{pmatrix}$$

- In practice, $\underline{v}_j \not\perp \underline{v}_i \; \forall \, i \neq j$ due to numerical rounding

Solutions:

- "live with it":

  "ghost" eigenvalues will emerge; since good EVs will replicate, have $k \gg n$ (usually $k = 4n$) until k EVs have

replicated

→ takes longer,
degeneracies very hard
to spot

- "reorthogonalize":
make $\underline{v}_j \perp \underline{v}_i$ by
reorthogonalization
→ takes lots of time
and, in particular,
memory to store
$k = n$ vectors of length
n!

- "partial reorthogonalization":
ARPACK, see below,
uses "implicit restart".

- uses:

  everywhere, when

  a few select EVals,
  EVecs,

  are needed

Steroids for diag methods

- ## pre-conditioning

- condition number

$$\kappa(A) = \frac{\lambda_{max}(A)}{\lambda_{min}(A)} \geq 1$$

so, remember power series algorithm, if $\kappa \gg 1$, then convergence is ~~slow~~ fast!

→ pre-conditioning: get κ ~~close to~~ away from 1!

- trick:

$$Ax = AP^{-1}\underbrace{Px}_{y} = b,$$

then if

$$\kappa(A) \lesssim \kappa(AP^{-1}),$$

one has improved convergence (at the price of an additional matrix multiplication – P^{-1} should be given simply).

– examples:

• Jacobi precond:

$$P = \operatorname{diag}(A) = \begin{pmatrix} a_{11} & & 0 \\ & a_{22} & \\ 0 & & \ddots a_{nn} \end{pmatrix}$$

• three-main-diagonals:

$$P = \begin{pmatrix} a_{11} & a_{12} & & 0 \\ a_{21} & a_{22} & \ddots & \\ & \ddots & & a_{n-1\,n} \\ 0 & & a_{n\,n-1} & a_{nn} \end{pmatrix}$$

work well if matrix is diagonally dominant.

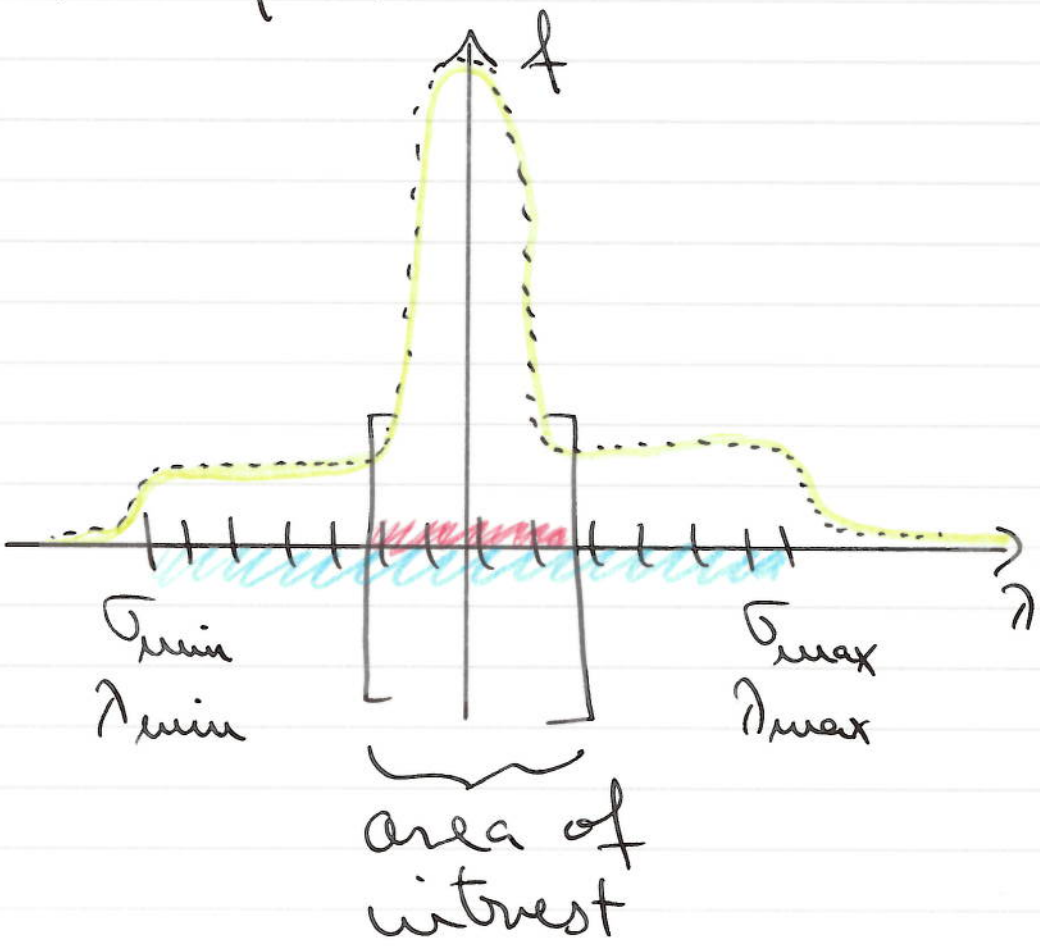
- ## Convergence acceleration

iterative methods work best for λ_{max}. Choose $f(A)$ such that

$$\lambda_{max}(f(A)) >> \lambda_{max}(A)$$

(same idea for σ_{min})

– example:



i.e.

Chebyshev polynomials

Fig 5.1

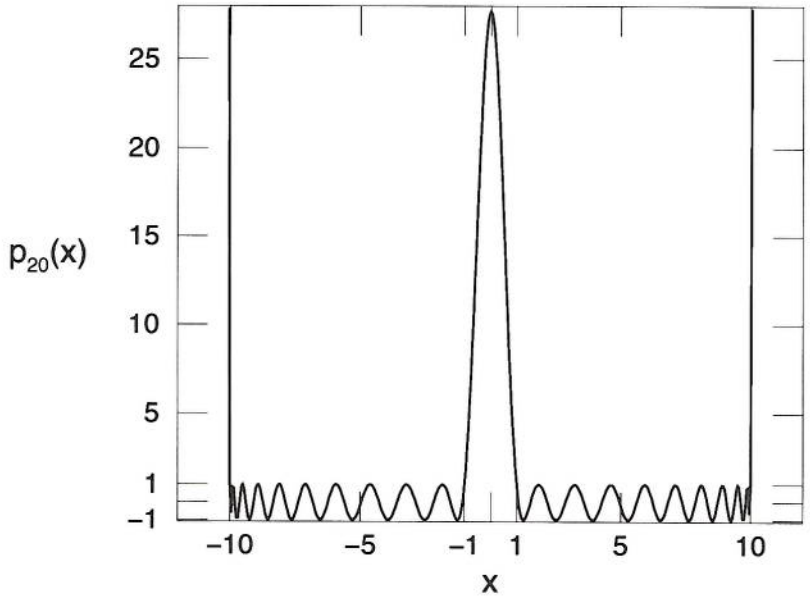


FIG. 5.1. *Chebyshev polynomial* $p_{20}(x)$ *with* $x_1 = 1$ *and* $x_2 = 10$.

Chebyshev-type polynomial, D. Sorensen and C. Sun, private communications.

• shift-and-invert

use as accelerator

$$\frac{1}{A-\lambda_0 \underline{1}\underline{1}}$$

since

$$\frac{1}{A-\lambda_0 \underline{1}\underline{1}} \cdot \underline{x}_0 = \infty \quad ,$$

so in practice, guess for $\underline{x}_0' \approx \underline{x}_0 + \delta \underline{x}$ gives

$$\frac{1}{A-\lambda_0 \underline{1}\underline{1}} \underline{x}_0' \rightarrow \text{very large}$$

↳ adjustable accelerator for area around λ_0

However, inversion A^{-1} is of course at least as comp. expensive as eigen-system solution!

↳ Trick:

instead of

$$\underline{v}_{k+1} = A\,\underline{v}_k$$

use

$$\underline{v}_{k+1} = \frac{1}{A-\lambda_0 \underline{1}}\,\underline{v}_k$$

↳ $$\underline{v}_k = \left(A-\lambda_0\underline{1}\right)\underline{v}_{k+1}$$

known $\hat{=}$ (lin. sys. eqn) unknown

$\Leftrightarrow$ equivalent to lin. sys. solver

→ use modern versions!

-> many such modern methods are <u>iterative</u> as well!

ARPACK (= ARNOLDI Package)

(works for unsymmetric matrices)

$\hat{=}$ Lanczos + Gram-Schmidt reorthogonalization of $k_0 \ll N$

$\underline{v}_k \in \{\underline{v}_0, \underline{v}_1, \ldots\}$

+ implicit restart:

When orthogonalization of k_0 $\{\underline{v}_k\}$ shows increasing loss of orthogonality,

then select new $\underline{v}_0 \perp \{\underline{v}_k\}$ and start process again for remaining $M - k_0$ Ritz vectors

Jacobi-Davidson

Assume

(x, λ) eigenpair s.t. $A\underline{x} = \lambda \underline{x}$

(u, θ) guess s.t. $A\underline{u} - \theta \underline{u} = \underline{r}$

(residuum)

then choose $\underline{t}$ s.t.

$$\underline{x} = \underline{u} + \underline{t}, \quad \underline{u} \perp \underline{t}$$

$$\hookrightarrow A(\underline{u}+\underline{t}) = \lambda(\underline{u}+\underline{t}) \qquad (*)$$

part of $(*) \perp \underline{u}$:

$$(I - \underline{u}\underline{u}^T) A (\underline{u}+\underline{t}) = (I - \underline{u}\underline{u}^T)\lambda(\underline{u}+\underline{t})$$

$\Leftrightarrow$

$$(\Delta) \quad (I - \underline{u}\underline{u}^T)(A - \lambda I)\underline{t} = (I - \underline{u}\underline{u}^T)(-A\underline{u} + \lambda \underline{u})$$

$$\overset{\text{RHS}}{=} -(I - \underline{u}\underline{u}^T)(A\underline{u}) = -(I - \underline{u}\underline{u}^T)(\underline{r} + \theta \underline{u})$$

$$= -\underline{r}$$

Since $\underline{t} \perp \underline{u}$, we can write

$$(1 - \underline{u}\underline{u}^T)\,\underline{t} = \underline{t},$$

then (Δ) is

$$\underbrace{(1-\underline{u}\underline{u}^T)\,(A-\lambda\mathbb{1})\,(1-\underline{u}\underline{u}^T)}\,\underline{t} = -\underline{r}$$

sym. matrix

But λ unknown, hence $\lambda \to \theta$

and

$$\underbrace{(1-\underline{u}\underline{u}^T)\,(A-\theta\mathbb{1})\,(1-\underline{u}\underline{u}^T)}\,\underline{t} = -\underline{r}$$

$$B \cdot \underline{t} = -\underline{r} \quad (\square)$$

$\Leftarrow$ lin. system of eqns

for B with known RHS

$\to$ solve for $\underline{t}$

then improve guess

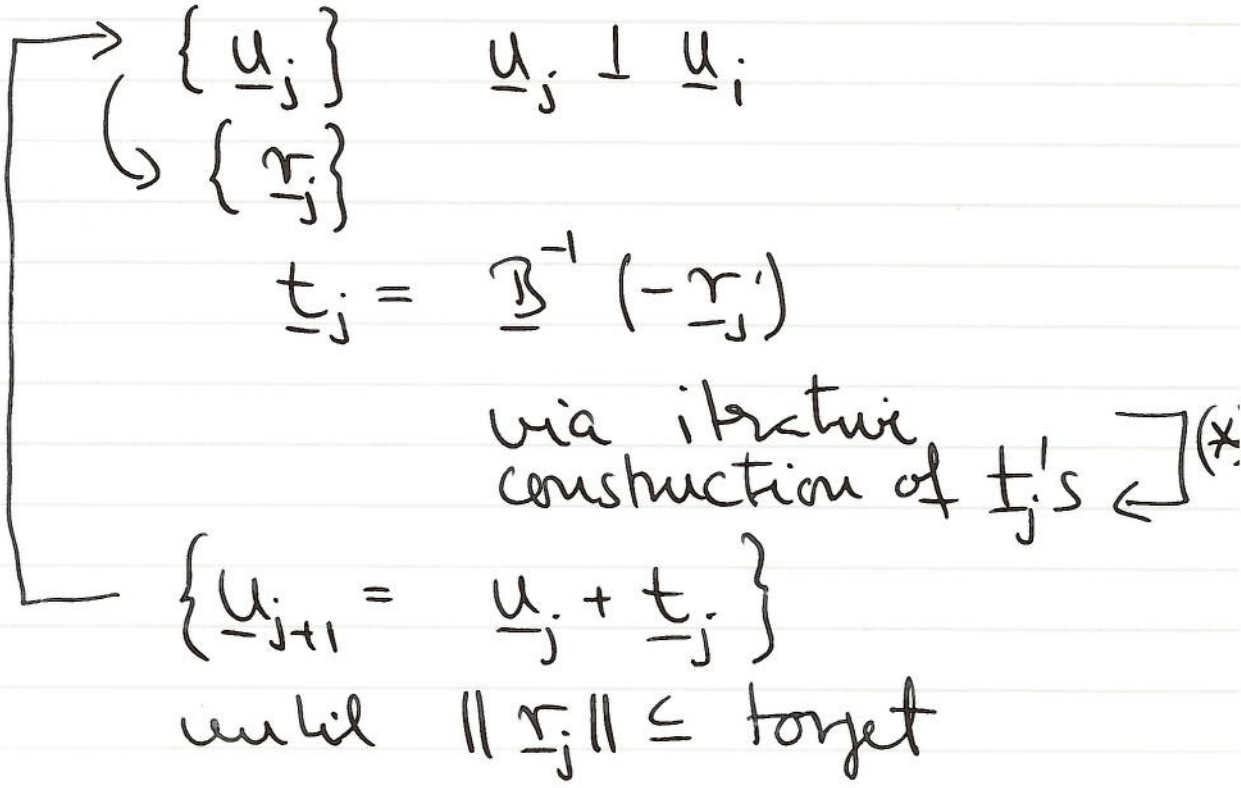
$$\underline{u}_{j+1} = \underline{u}_j + \underline{t}$$

$$\theta_{j+1} = \theta_j + \underline{u}_j^T A \underline{t}$$

Advantage:

The LSE need not be solved with extreme accuracy since we improve outside loop anyhow!

Overview:



(*) we use "ILUPack".

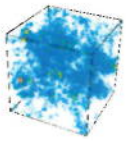
$\hat{=}$ inverse-based incomplete LU decomposition

Large Scale Anderson Diagonalization

2 levels of iterations to solve $H\Psi = E_0\Psi$ 100^3

Outer iteration: Jacobi-Davidson (JD)
Inner iteration: ILUPack$^+$ to solve shift-and-invert problem

- iterative methods use $H^n\Psi_0 \to E^n_{\max}\Psi_{\max}$, Krylov sequence $\Psi_{n+1} = H\Psi_n = H^2\Psi_{n-1} = \ldots$ — 1999 30 days, 2006 3 days
- shift-and-invert approach $H \to 1/(H - E_0)$ to target E_0 region rewrite as $(H - E_0)\Psi_{n+1} = \Psi_n$, aka sys. lin. equations — 6 hrs
- fast *iterative* SLE solver ILUPack and tricks — 40 min
- JD is ok with approximate states Ψ_{n+1} — 20 min
- tinkering with ILUPack and jdbsym interface — 12 min
- use new package JADAMILU — 10 min



$N = L^3 = 350^3 = 42.875.000$ **in 2006!**

Schenk *et al.*, SIAM Reviews 50, 91–112 (2008)

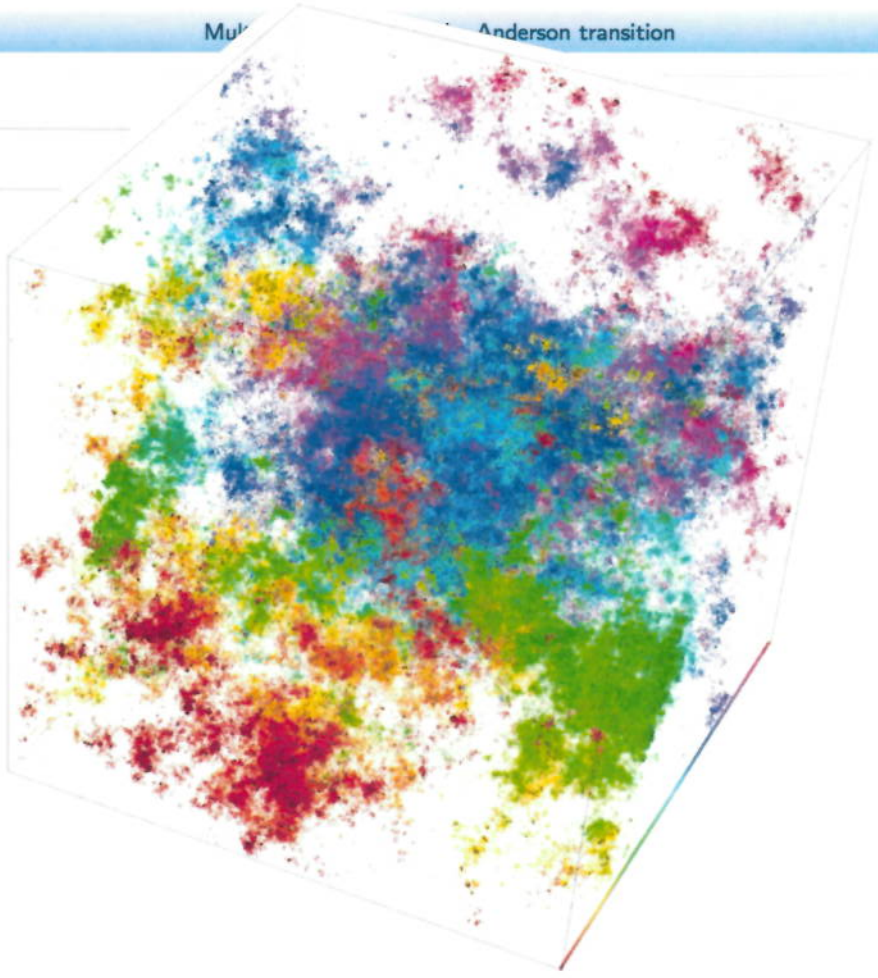


Fig. 2 *Plot of the electronic eigenstate at the metal-insulator transition with $\lambda = 0$, $w = 16.5$, and $N = 350^3$. The box-and-color scheme is as in Figure 1. Note how the state* extends *nearly everywhere while at the same time exhibiting certain* localized *regions of higher $|x_j|^2$ values. The eigenstate has been constructed using* ILUPACK*-based Jacobi–Davidson. See section 9 for details.*

PARDISO – a direct solver

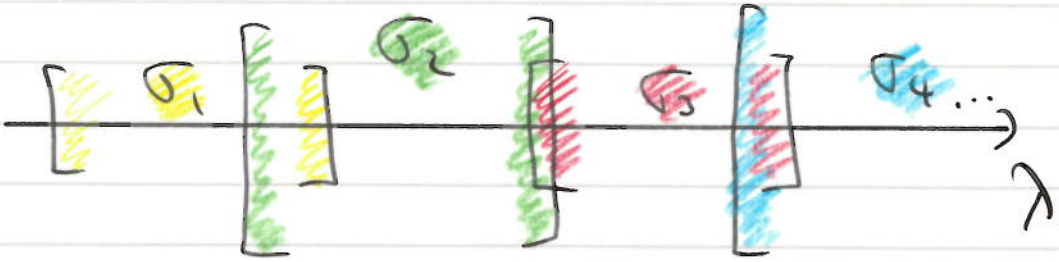
- $$\left.\begin{aligned} B\,\underline{t} &= -\underline{r} \\ \underline{t} &= ? \end{aligned}\right\} \text{LSE}$$

- Shift-and-invert with PARDISO also possible, but generally needs memory (fill-in)

- part of Intel MKL (you may have used it already!)

- ARPACK + S&I + PARDISO is fast if you have the memory!

- More than a few eigenpairs?



Use overlapping intervals in eigenvalue space, preferably via naive / task farm parallelization.

Conclusions

use iterative solvers if

- only few $\{\underline{x}, \lambda\}$ needed

or

- for capability challenge:
  - A too big to fit in memory
  - no other method works

References:

1. Elsner, U., Mehrmann, V., Milde, F., Römer, R. A. & Schreiber, M., The Anderson Model of Localization: A Challenge for Modern Eigenvalue Methods. *SIAM J. Sci. Comput.* **20,** 2089–2102 (1999).

2. Schenk, O., Bollhoefer, M. & Römer, R., On large-scale diagonalization techniques for the Anderson model of localization. *SIAM J. Sci. Comp*. **28**, 963-983 (2006)

Using JADAMILU is free for non commercial applications (For commercial use, please contact the authors). You can acknowledge, using the references

3. M. Bollhöfer and Y. Notay, JADAMILU: a software code for computing selected eigenvalues of large sparse symmetric matrices, Computer Physics Communications, vol. 177, pp. 951-964, 2007.
4. M. Bollhöfer and Y. Notay, JADAMILU code and documentation. Available online at http://homepages.ulb.ac.be/~jadamilu/.

The PARDISO Version 5.0.0 has been released in January 2014. It contains full support of multi-threaded Schur-complement computations and full support for parallel selected inversion. In case that you are using the new version 5.0.0 please cite:

5. M. Luisier, O. Schenk et. al.,Fast Methods for Computing Selected Elements of the Green's Function in Massively Parallel Nanoelectronic Device Simulations, Euro-Par 2013, LNCS 8097, F. Wolf, B. Mohr, and D. an Ney (Eds.), Springer-Verlag Berlin Heidelberg, pp. 533–544, 2013,

6. O. Schenk, M. Bollhoefer, and R. Roemer, On large-scale diagonalization techniques for the Anderson model of localization. Featured SIGEST paper in the SIAM Review selected "on the basis of its exceptional interest to the entire SIAM community". SIAM Review 50 (2008), pp. 91-112. (updated version of [2])

7. O. Schenk, A. Waechter, and M. Hagemann, Matching-based Preprocessing Algorithms to the Solution of Saddle-Point Problems in Large-Scale Nonconvex Interior-Point Optimization. Journal of Computational Optimization and Applications, pp. 321-341, Volume 36, Numbers 2-3 / April, 2007.