Fields institute mini-course : Lecture 5 April 18, 2018

Today

Eigensystem:

* 1D eig. vs. eigs
* 2D (3D)
  - Specific observations/comments about single particle Schroedinger operator in 2D (3D)
  - Subspace methods.

Electrostatics { singular sources, & doping solution in 2D & 3D.

References:

* Parlett "The Symmetric Eigenvalue Problem"
* Golub and Van Loan "Matrix Computations" Chap. 8. (8.2.4)

In sample codes (https://github.com/canderson6151/SP1D to compute the eigensystem I use eig and not eigs.

```
[eigVecTemp,eigValues] = eig(full(A));
```

Since A is represented as a sparse matrix, alternately one can use

```
[eigVecTemp,eigValues] = eigs(A);
```

I went to the trouble of creating sparse representations - so why not use eigs? I don't know how well eigs works since I've not used eigs extensively, so I use eig, a function that I've used many times. When the grid size gets large, using eig may lead to inefficiencies.

Things to do

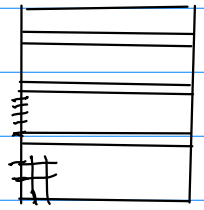
(1) investigate the use of eigs vs. eig

(2) investigate both when requesting only a small # of eigenvalues and eigenvectors.

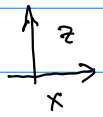
For the eigensystem problems for 2D & 3D Schroedinger-Poisson - you will definitely want to use eigs (if you are doing it in matlab).

Schroedinger - Poisson eigensystem 2) (3D)

* Assume the effective mass is constant as a function of z. Domain periodic in x.



$$\left[\frac{\hbar^2}{2m_e} \Delta + V(\vec{x}) \right] \psi = \lambda \psi \qquad \Delta = \frac{\partial^2}{\partial z^2} + \frac{\partial^2}{\partial x^2}$$

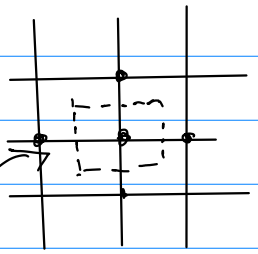


set up discrete approximation, using finite / volume discretization for example.

Grid ?

z: semi-uniform

x: uniform



A discrete symmetric linear operator on the grid values

integrate equation over control volume and use divergence theorem.

$S_n : \mathbb{R}^n \to \mathbb{R}^n$

$A : \mathbb{R}^n \to \mathbb{R}^n$ (matrix representation of the operator)

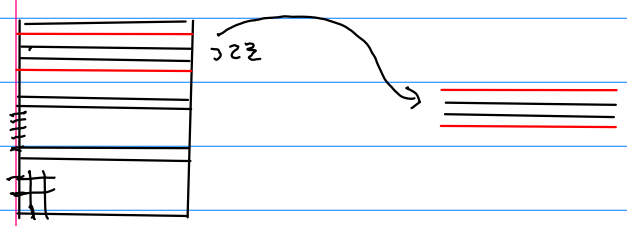
$n \sim$ # of grid points (not exactly equal because the boundary points aren't used)

n can be large, $100 \times 100 = 10,000$ $\quad 1000 \times 100 = 100,000$

(3D) $100 \times 100 \times 100 = 1M.$

Very large eigen problems.

One may be able to reduce the size of the problem by only computing eigenfunctions in layers near the quantum wells



pull out quantum wells and immediately adjacent wells

Don't use a grid where you know the wave functions will be 0.

How do you know which wells are isolated? Use a 1D solution with, say a locally averaged potential, to get an idea of the vertical extent of the wavefunction.

Exercise for 1D code: Compare solutions of eigenproblems solved over the whole domain with solutions created locally.

Even if you are able to localize the calculation, you still have a large (high dimensional) eigenproblem. Generally, to reduce the computation time you choose methods that only compute the eigenvalues and eigenvectors you need.

$$A\vec{v} = \lambda\vec{v}$$

only the lowest eigenvalue (ground state) — "Standard Lanczos"

multiple lowest states

selected eigenvalues (usually the lowest) and you want to "get involved" with the procedure.

packages that implement

- Block Lanczos
- Krylov Subspace methods (ARPACK)
- Subspace iteration.

- subspace iteration.

I want to spend time on just discussing the general task of solving eigenproblems, and then discuss subspace iteration. A way to get a general idea about why the methods I mentioned work.

Eigensystem problems associated with multi (>1) dimensional linear operators. Assume self-adjoint operators.

$$\hat{S} V = \lambda V \qquad V(\vec{x}) : \mathbb{R}^n \to \mathbb{R}^n$$

$\hat{S}$: multi-dimensional differential operator

Solution techniques?

One way:

Choose a set of functions $\{ q_j(\vec{x}) \}_{j=1}^{P}$ and assume

$$V(\vec{x}) = \sum_{j=1}^{P} c_j q_j(\vec{x})$$

To get a set of equations, insert representation into $S\vec{v} = \lambda\vec{v}$ and form inner products with q_j's.

$$\langle q_i, S \sum_{j=1}^{P} c_j q_j \rangle = \langle q_i, \lambda \sum_{j=1}^{P} c_j q_j \rangle \quad \text{for } i = 1 \ldots P$$

$$\begin{pmatrix} & \tilde{S} & \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_P \end{pmatrix} = \lambda \begin{pmatrix} & \tilde{O} & \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_P \end{pmatrix}$$

$$\tilde{S}_{i,j} = \langle q_i, S q_j \rangle \qquad \tilde{O}_{i,j} = \langle q_i, q_j \rangle$$

If the $\{q_j\text{'s}\}$ are not orthonormal $\Rightarrow \tilde{O}_{i,j}$ will not be the identity so one has a generalized eigenproblem.

If the set of vectors $\{q_j\}$ are good approximation to the desired set of eigenvalues and eigenvectors, this isn't a bad idea. If P isn't to large there are routines in LAPACK or matlab calls that can be used to solve generalized eigenproblems.

However,

(a) You need to choose/find the q_js. Also, you don't necessarily know which eigenvalues and eigenvectors you are approximating.

(b) You need a systematic way to choose additional q_js to improve accuracy.

(c) The matrix $\tilde{S}$ is typically a dense matrix (mostly all entries are non-zero) so that the computational cost of solving the generalized eigenproblem is $O(P^3)$. When P gets large, this can be prohibitively expensive, or require that sophisticated solution procedures be developed.

(d) It's a generalized eigenvalue problem, and the perturbation theory and error estimation are more complicated than for standard eigenvalue problems

Even with such concerns, it is a not so uncommon technique used to determine eigenfunctions and eigenvalues.

What else can be done? Instead of using analytic functions, use functions represented on a grid. This choice has more consequences besides expanding the number of degrees of freedom used to represent the eigenvalues and eigenvectors.

Suppose you can approximate S by a symmetric operator on a grid?

$\vec{v}$

$v(x) \sim$  $\quad Sv \approx S_n \vec{v}$

So, one approximates the eigensystem problem by
$S\vec{v} = \lambda\vec{v} \;\approx\; S_n\vec{v} = \lambda\vec{v}$.

How to solve $S_n\vec{v} = \lambda\vec{v}$?
In matrix form:
$S_n\vec{v} \to A\vec{v} = \lambda\vec{v}$ $\qquad A : \left(\#\text{grid points}\right) \times \left(\#\text{grid points}\right)$ $\qquad \left\{ \begin{matrix} 100 \times 100 \\ \Rightarrow 10{,}000 \times \\ 10{,}000 \end{matrix} \right.$ (2D)

$\Rightarrow$ A large (but sparse)

How to reduce the size of the matrix we need to work with?
Use the same idea as in the continuous operator case.

Choose set of $\{\vec{q}_j\}_{j=1}^{p}$ of grid functions, assume $\vec{v} = \sum_{j=1}^{p} c_j \vec{q}_j$
plug in and obtain

$$\hat{A}\vec{c} = \lambda \hat{O}\vec{c} \qquad \hat{A}_{i,j} = \langle \vec{q}_i, \vec{q}_j \rangle_h \qquad \hat{O}_{i,j} = \langle \vec{q}_i, \vec{q}_j \rangle_h$$

A generalized eigenproblem.

!!! But, it's easy to orthonormalize a set of grid functions
(modified gram-schmidt, QR decomposition, etc.)

Therefore given $\{\vec{q}_j\}_{j=1}^{p}$ $\rightsquigarrow$ make them orthonormal $\{\vec{q}_j\}_{j=1}^{p}$
and we therefore can always obtain a standard eigenproblem $\hat{A}\vec{c} = \lambda\vec{c}$

There are good, and very accurate methods for finding eigenvectors of small to medium size dense matrices.

What about choosing the $\vec{q}_j$'s? This is where subspace iteration comes (or Krylov subspace, or vectors in Lanczos's method ...)

Subspace iteration.

General subspace iteration with a fixed size subspace. A: $n \times n$ symmetric.

- Choose r = dimension of a subspace
- $n \times r$ Q_0 s.t. $Q_0^T Q_0 = I$. $\quad Q_0 = [\vec{q}_0 \cdots \vec{q}_r] \quad (S_k = \text{span}\{\vec{q}_j\}_{j=1}^{r})$

For $k = 1, 2, 3 \ldots$

"Enrichment" $\quad Z_k = F(A) Q_{k-1}$ ..... $F(A) = A$: Name: orthogonal or simultaneous iteration

"Orthonormalization" $\quad \tilde{Q}_k R_k = Z_k$ $\quad F(A) = A^{-1}$: Inverse subspace iteration

"Diagonalization" $\quad \tilde{Q}_k^T A \tilde{Q}_k \tilde{c} = \tilde{\theta} \tilde{c}$ $\quad F(A) = p(A)$: p, q polynomials

$\vec{u}_i^k = \tilde{Q}_k \tilde{c}_i \quad \lambda_i^k = \tilde{\theta}_i$ $\quad F(A) = q^{-1}(A) p(A)$

approx. eigenvectors $\quad$ approx. eigenvalues

"Update" $\quad Q_k = U_k = [\vec{u}_1^k \; \vec{u}_2^k \cdots \vec{u}_r^k]$

check for convergence.

The design goal for F is to pick a function that enriches the subspace with components of the eigenvectors you want.

What eigenvectors and eigenvalues do the approximations converge to?

$F(A) = A$? $\vec{u}_i$ corresponding to λ_i s for which $|\lambda_i|$ is largest.

$F(A) = A^{-1}$? $\vec{u}_i$ corresponding to λ_i s for which $\frac{1}{|\lambda_i|}$ are largest

$F(A) = p(A)$? $\vec{u}_i$ corresponding to λ_i s for which $|p(\lambda_i)|$ are largest.

$F(A) = \vec{g}^{-1}(A)p(A)$: $\vec{u}_i$ corresponding to λ_i s for which $\frac{|p(\lambda_i)|}{|g(\lambda_i)|}$ are largest.

To find eigenvalues of smallest magnitude, one can use $F(A) = A^{-1}$.

Q: Isn't it expensive? Requires A^{-1}. If you have an efficient method for solving $A\vec{z} = \vec{q}$ ($\vec{z} = A^{-1}\vec{q}$) then this may be an acceptable procedure.

What's going on in the "enrichment" step? If $\{\vec{v}_i\}_{i=1}^{N}$ are the eigenvectors of A then given $\vec{w} = \sum_j \beta_j \vec{v}_j$

$$A\vec{w} = \sum_j \underbrace{\lambda_j \beta_j}_{\beta_j'} \vec{v}_j \qquad \beta_j' = \lambda_j \beta_j$$

(components of large eigenvalues get boosted relative to smaller eigenvalues)

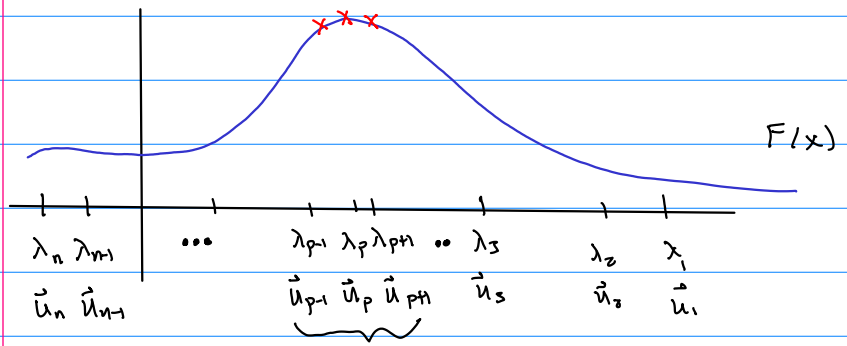
$$A^{-1}\vec{w} = \sum_j \frac{1}{\lambda_j}\beta_j \vec{v}_j \qquad \beta_j' = \frac{1}{\lambda_j}\beta_j$$

(components of smallest eigenvalues get boosted relative to larger eigenvalues)

$$\vec{g}^{-1}(A)p(A)\vec{w} = \sum_j \frac{p(\lambda_j)}{g(\lambda_j)}\beta_j \vec{v}_j$$

(components where $p(\lambda_j)/g(\lambda_j)$ is largest get boosted.)

Suppose on applies the procedure with $r = 3$ and where $F(x)$ has the following form.



$$Q_K \to \left[\vec{u}_{p-1}, \vec{u}_p, \vec{u}_{p+1}\right] \text{ as } K \to \infty .$$

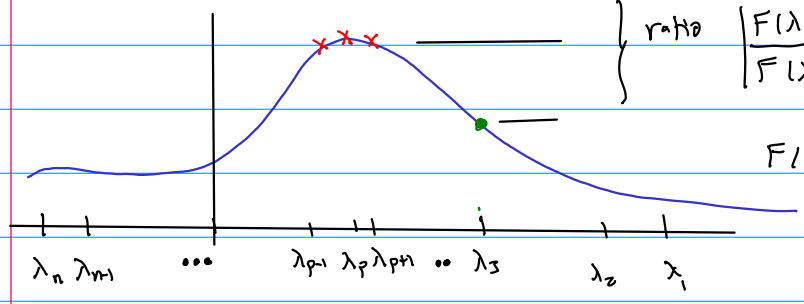
The subspace the method will converge to will be the eigenvectors $\vec{u}_{p-1}, \vec{u}_p, \vec{u}_{p+1}$ since $|F(\lambda_{p-1})|, |F(\lambda_p)|, |F(\lambda_{p+1})|$ are the largest.

∴ the eigenvalues of $Q_K^T A Q_K$ will converge to $\lambda_{p-1}, \lambda_p, \lambda_{p+1}$.

Results in §8.2.1 for orthogonal iteration applied to $F(A)$ imply that when using a subspace of size r the rate of convergence of the subspace Q_K is proportional to

$$\left| \frac{F(\lambda^*)}{F(\tilde{\lambda})} \right|^K \qquad \tilde{\lambda} = \underset{\lambda_j \notin D}{\arg\min} |F(\lambda_j)| \qquad \lambda^* = \underset{\lambda_j \notin D}{\arg\max} |F(\lambda_j)|$$

where D = r eigenvalues corresponding to the largest values of $F(\lambda_j)$.



ratio $\left|\frac{F(\lambda_3)}{F(\lambda_{p-1})}\right|$ determines the rate of convergence.

large gap ⇒ fast convergence

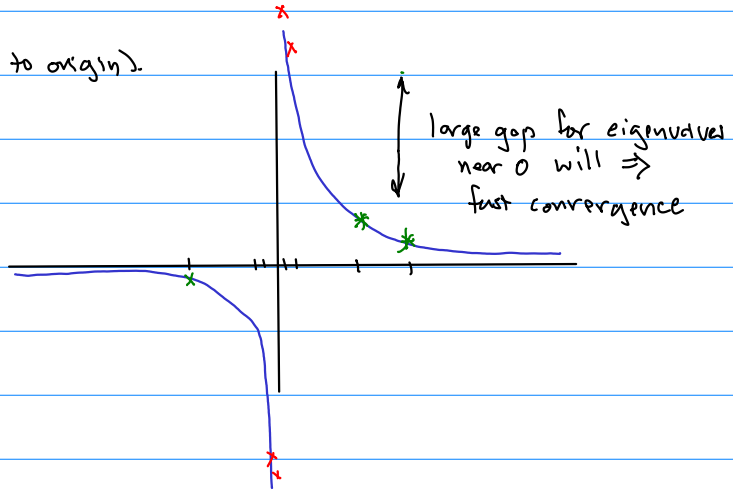
Want to find eigenvalues & eigenvector in a given range?

Idea: Choose $p(x)/q(x)$ so that $\left|p(x)/q(x)\right|$ is largest when x = desired λ_j's.

Standard examples.

- Eigenvalues of smallest modulus (closest to origin).

  Use $F(A) = A^{-1}$ $\left(p(x)q(x) = \frac{1}{x} \right)$



  convergence rate:

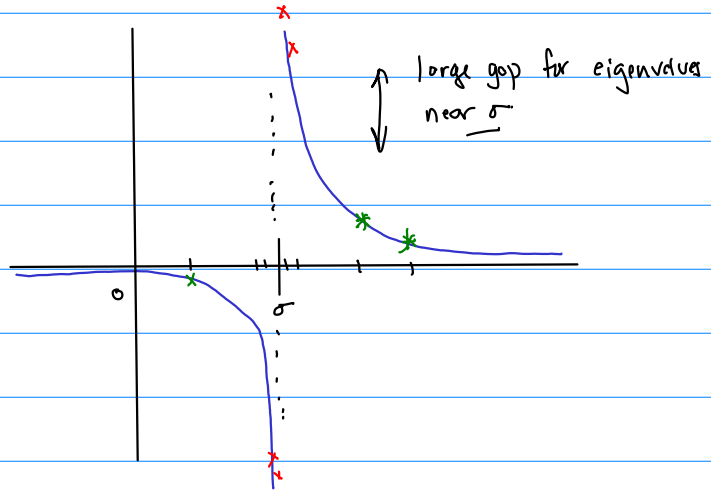
$$\left| \frac{1}{\lambda_{n-r}} \right| / \left| \frac{1}{\lambda_{n-(r-1)}} \right| = \left| \frac{\lambda_{n-(r-1)}}{\lambda_{n-r}} \right|^{k}$$

  "Inverse Subspace Iteration."

- Eigenvalues close to a value σ.

  $F(A) = (A - \sigma)^{-1}$

  → one can get very rapid convergence for eigenvalues near σ.



"Inverse subspace iteration with shift".

This is fine, but one has to be able to apply A^{-1} or $(A - \sigma)^{-1}$.

* If A is large then this ⇒ need to be able to solve rapidly large linear systems.
* Generally you pick σ near the eigenvalues of interest.

  Unfortunately ⇒ $A - \sigma$ is nearly (or is) singular ⇒ complications in applying $(A - \sigma)^{-1}$.

Some ways around the complications:

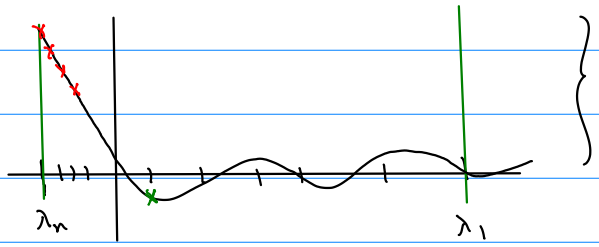
(1) If it is acceptable to use Gaussian Elimination with partial pivoting, then solving $(A - \sigma)u = f$, even when $A - \sigma$ is nearly singular is "ok".

I'll demonstrate this. Justification given in Parlett, or, work out what happens when GE with partial pivoting is applied to a small singular system.

(2) Don't use A^{-1} or $(A-\sigma)^{-1}$; avoid inverses by using a $p(x)$ s.t. $|p(x)|$ is large when $x = \lambda_j$'s of interest.

Example:

Algebraically smallest eigenvalues?

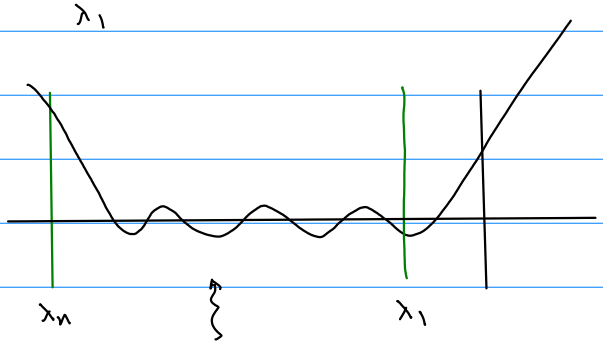


Choose $p(x)$ so $|p(\lambda_j)|$ largest for $j = n, n-1, \ldots n-(r-1)$

Ideas?

Chebyshev polynomial.

$C_k(x)$



Choose degree and scale so polynomial has this form.

"Rayleigh-Chebyshev" method

- Origin of the idea due to Lanczos, extended to subspaces & added adaptivity.

Generally, if you are going to construct good $p(x)$'s you need to have estimates of the spectral range $[\lambda_n, \lambda_1]$. Fortunately Lanczos's method is particularly good at this.

For details on this see the reference:

C.R. Anderson, "A Rayleigh-Chebyshev procedure for finding the smallest eigenvalues and associated eigenvectors of large sparse Hermitian matrices", J. Comput. Phys. 229, 7477-7487, (2010).