Fields institute mini-course : Lecture 4 April 17, 2018

Continuing with the solution of the 1D Schroedinger-Poisson equations.

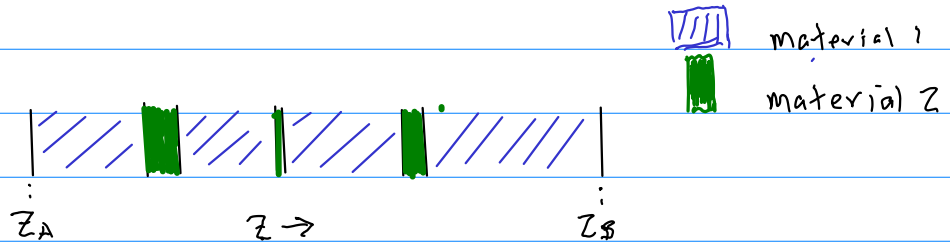
Today:

- Quick summary
- 1D Schroedinger-Equation — Eigensystem computation
  - Self-consistent iteration
  - Matlab (or Octave) sample code.
- N-particle Schroedinger ⇝ Schroedinger-Poisson.

Last time: * many models of systems involving QM behavior involve solving Poisson's equation and some form of Schroedingers equation

* The Schroedinger-Poisson equations are a system of equations where one can relatively easily develop experience with solution procedures for both of the types of equations. (For some problems it is a useful model to use.) Also, by starting out with the 1D Schroedinger-Poisson problem, one is working on a problem whose solutions can be obtained quickly and get an understanding about practices and procedures to follow for 2D and 3D problems

Equations



$$\frac{d}{dz}\chi(z)\frac{d}{dz}\phi_D = -\rho_D(z) \qquad \phi_D(z_A) = g_A \qquad \phi_D(z_B) = g_B \qquad \rho_D = \text{fixed charge}$$

$$\frac{d}{dz}\chi(z)\frac{d}{dz}\tilde{\phi} = -\rho_e(z) \qquad \tilde{\phi}(z_A) = 0 \qquad \tilde{\phi}(z_B) = 0$$

$$\phi(z) = \tilde{\phi}(z) + \phi_D(z) + \phi_B(z) \qquad [\phi_B(z) < \text{"band shift"}$$

$$\rho_e(z) = 2\sum_{\lambda_j < E_f}{}' \psi_j^*(z)\psi_j(z)\,W(\lambda_j, E_f) \qquad \left\{ (E_f - \lambda_j)\frac{m_r(z)}{2\pi\hbar^2}\right.$$

$$-\frac{\hbar^2}{2}\left[\frac{d}{dz}\left(\frac{1}{m_e(z)}\frac{d}{dz}\right) + \left[\tilde{\phi}(z) + \phi_D(z) + \phi_B(z)\right]\right]\psi_j = \lambda_j\psi_j \qquad \boxtimes$$

Solution of equations for ϕ_D and $\tilde{\phi}$?

Equations $\rightsquigarrow$ finite volume discretization

$$L\phi_D = -D\rho_D - f_{bdry} \qquad L\tilde{\phi} = D\rho_e$$

D: diagonal matrix; doping (D)

approximation obtained by solving a symmetric tri-diagonal system.

Next? Computing approximations to the eigensystem for $\boxtimes$

Given a problem $\frac{d}{dz} a(z) \frac{d}{dz} u = f$ with $u(z_A) = u(z_B) = 0$ we know how to create matrices so that an approximate solution is obtained by solving a matrix problem

$$L\vec{u} = D\vec{f}$$

↙ no boundary forcing when $u(z_A) = u(z_B) = 0$.

Using the same discretization technique we can construct matrices so that eigenfunctions of ⊠ are obtained by solving the matrix eigenvalue problem

$$K\vec{u} + DP\vec{u} = \lambda D\vec{u}$$

from $\frac{d}{dz} x(z) \frac{d}{dz}$

$$P = \begin{bmatrix} \phi(z_1) & & \\ & \phi(z_2) & \\ & & \ddots \\ & & & \phi(z_N) \end{bmatrix} \qquad D = \begin{pmatrix} h_1 & & \\ & \frac{h_1+h_2}{2} & \\ & & h_2 \\ & & & \ddots \end{pmatrix}$$

(K for kinetic) (P for potential) diagonal matrix of mesh sizes.

This is a generalized eigenproblem because it's $\lambda D\vec{u}$ and not $\lambda\vec{u}$. There are standard ways to turn such problems into standard eigenvalue problems - we'll use one that uses $D^{1/2}$.

Let $\vec{v} = D^{1/2}\vec{u}$ so $\vec{u} = D^{-1/2}\vec{v}$

$$\Rightarrow \quad K D^{-\frac{1}{2}}\vec{v} + DPD^{-1/2}\vec{v} = \lambda D^{1/2}\vec{v}$$

$$\equiv \quad D^{-1/2} K D^{-1/2}\vec{v} + D^{-1/2} D P D^{-1/2}\vec{v} = \lambda\vec{v}$$

↓ ↓ ← ⊠ diagonal matrices commute.

$$\tilde{K}\vec{v} + P\vec{v} = \lambda\vec{v}$$

so, solve $(\tilde{K} + P)\vec{v}_j = \lambda_j \vec{v}_j$, then $\vec{u}_j = D^{-1/2}\vec{v}_j$

$\tilde{K} = D^{-1/2} K D^{-1/2}$ $\Rightarrow$ a symmetric tri-diagonal matrix

I mentioned that it's important (or at least very useful) when solving linear problems with self-adjoint operators that the matrices that arise in the discrete approximations be symmetric.

When approximating eigenfunctions of self-adjoint operators it is very very important that the matrices are symmetric.

Note: The discrete eigenvectors $\vec{v}_j$ will be orthonormal with respect to the standard inner product $\langle \vec{v}, \vec{w} \rangle = \sum_i w_i v_i$ but the vectors $\vec{u}_j$ will in general not be, $\langle \vec{u}_p, \vec{u}_q \rangle \neq \delta_{p,q}$.

However, $\langle \vec{v}_p, \vec{v}_q \rangle = \delta_{p,q} \Rightarrow \langle \sqrt{D}\,\vec{u}_p, \sqrt{D}\,\vec{u}_q \rangle = \langle \vec{u}_p, D\vec{u}_q \rangle = \delta_{p,q}$.

But $\langle \vec{u}_p, D\vec{u}_q \rangle$ = Trapezoidal rule approximation to $\int_{z_A}^{z_B} u_p(z) u_q(z)\, dz$.

which is actually what we want, since we are letting the mesh spacing $\to 0$ we need a mesh weighted inner product (and norm).

GK. One can therefore create approximations to the solutions of each part of the 1D Schroedinger-Poisson equations.

$$\Delta_x \phi_D = -\rho_D \quad \phi_D(z_A) = g_A \quad \phi_D(z_B) = g_B \qquad \left(\Delta_x = \frac{d}{dz}\chi(z)\frac{d}{dz}\right)$$

(I) $$\Delta_x \tilde{\phi} = -\rho_e \quad \tilde{\phi}(z_A) = 0 \quad \tilde{\phi}(z_B) = 0$$ / Electrostatics

$$\rho_e = \sum_{\lambda_j < E_f} \psi_j^*(z)\psi_j \, W(E_f, \lambda_j)$$

(II) $$\left(\Delta_m + [\tilde{\phi} + \phi_D + \phi_B]\right)\psi_j = \lambda_j \psi_j$$ / Eigensystem.

But they are coupled ... and $\tilde{\phi}$ and ρ_e have to be "self-consistent"
To see this more clearly, I'm going to express operators at a higher level.
Let $\phi_{base} = \phi_B + \phi_D$. It need only be evaluated once.
One can then formally write the self-consistent solution as

$\tilde{\phi} = -\Delta_x^{-1} \rho_e(\tilde{\phi}, \phi_{base})$ where $\rho_e(\tilde{\phi}, \phi_{base})$ is the construction of ρ_e by solving the eigenproblem using $\phi = \tilde{\phi} + \phi_{base}$.

So finding a self-consistent solution is one of solving a 'fixed point' problem.

Simple iteration (or "fixed point iteration") takes the form
$\tilde{\phi}^0$ given

$$\tilde{\phi}^{k+1} = -\Delta_x^{-1} \rho_e(\tilde{\phi}^k, \phi_{base}) \qquad \tilde{\phi}^1, \tilde{\phi}^2 \cdots \text{ until } \|\tilde{\phi}^{k+1} - \tilde{\phi}^k\| \text{ small.}$$

Set $\phi = \bar{\phi}^k + \phi_D + \phi_B$ $\quad k$ iterate of converged $\tilde{\phi}$.

Sometimes this doesn't converge - so you use a relaxation factor γ.

Self-Consistant iteration with a relaxation factor γ.

ϕ^0 given

$$\phi^* = -\Delta_x^{-1} \rho_e(\phi^k, \phi_{base})$$

$$\phi^{k+1} = (1-\gamma)\phi^k + \gamma\phi^*$$

{ typically $\gamma < 1$, so one is adding less of the most recently evaluated $\tilde{\phi}$

Problem: How do you pick γ? Pick a value, see what happens

This is not particularly efficient, and the values of γ that lead to rapid convergence (if convergence at all) are problem dependent.

Another approach "evolve" to the solution.

$$\tilde{\phi} = -\Delta_x^{-1} \rho_e(\tilde{\phi}, \phi_{base})$$

$$\Rightarrow \quad 0 = -\Delta_x^{-1} \rho_e(\tilde{\phi}, \phi_{base}) - \tilde{\phi}$$

Solve (**) $\dfrac{d\phi}{dt} = -\Delta_k \rho_e(\tilde{\phi}, \phi_{base}) - \tilde{\phi}$ to steady state.

Where did this idea come from?

$$\phi^* = -\Delta_x^{-1} \rho_e(\phi^k, \phi_{base}) \Rightarrow \frac{\phi^{k+1} - \phi^k}{\gamma} = -\Delta_x^{-1} \rho_e(\phi^k) - \phi^k$$
$$\phi^{k+1} = (1-\gamma)\phi^k + \gamma\phi^*$$

Forward Euler for (**)

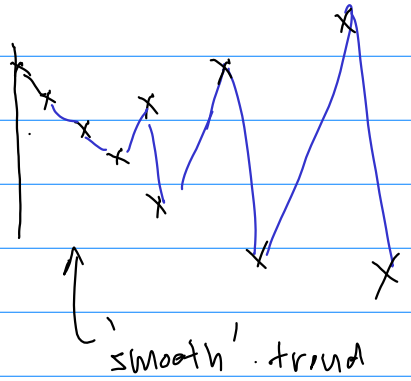
Small γ to obtain convergence $\Rightarrow$ a mildly 'stiff' or 'stiff' ODE.

→ There are other ODE methods to use that might work better.

One family "stabilized Runge-Kutta methods", but there are many to choose from.

Also, one can do adaptive time stepping. A simple way of doing it, which in my experience works rather well is based upon observing what happens when numerical ODE methods go "unstable"

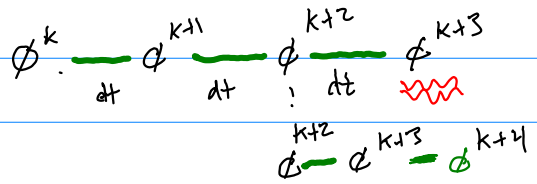
$\frac{d\vec{y}}{dt} = f(\vec{y})$



← large oscillations when numerically unstable.

so, observe $\|\tilde{\phi}^k\|$, $\|\tilde{\phi}^{k+1}\|$, $\|\tilde{\phi}^{k+2}\|$ ... etc:

and "roll back and refine the time step" when a quantity such as $(\|\phi^{k+1}\| - \|\phi^k\|) - (\|\phi^k\| - \|\phi^{k-1}\|)$ is large



} roll back, start with a small time step $\frac{dt}{2}$ say.

Point: since we want to get to steady state, we're not overly concerned with time accuracy along the way – we try to pick dt as large as possible and still have numerical stability.

https://github.com/canderson6151/SP1D.git

The sample code is written in matlab syntax, and runs in matlab or Octave.

It's an implementation of the discretization procedures and self-consistant solution procedure I've previously described.

- It's a type of code that I would typically develop/use while teaching.

Exercise #1: Using the scripts to create the sparse matrices used to create approximate solutions, think up a test problem and verify the accuracy and rate of convergence of the approximation.

Exercise #2: Validate the eigensystem computation. Verify accuracy and rate of convergence.

Exercise #3: Investigate the use of other ODE methods to obtain the self-consistent solution.

But, we don't have time for the exercises, so you are just being presented with the complete code. It's therefore an "alpha" version, since the component validations haven't been completed.

Structure:

* set material properties, values of physical constants. (units? nm, eV, sec)
* specify layer structure and material properties
* create sparse matrix representations of operators
* Do some number of sets of forward Euler. ↻

A.Szabo and N.S. Ostlund.
Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory
Dover Books

When I started working on methods to simulate a quantum system, I was just "shown" the Schroedinger-Poisson equations. I too wondered how they were related to the other types of approximations I'd heard about ... density functional theory, Hartree-Fock, ...

Without going into very great detail, a sketch of a 'derivation' is as follows.

$i\hbar \frac{\partial \tilde{\psi}}{\partial t} = H\tilde{\psi}$ $\quad$ H N-particle Schroedinger operator for electrons in a potential $V(x)$

$\downarrow$

(spatial coord. ↙, spin coord. ↙)

See solutions $\tilde{\psi} = e^{iwt}\psi(\vec{r}_1, \vec{r}_2 \cdots, \sigma_1 \cdots \sigma_N)$

$\Rightarrow$ ψ satisfies an eigenvalue problem $H\psi = \lambda\psi$.

Solutions of $H\psi = \lambda\psi$?

Use a representation of ψ that is a product or sums of products of functions of $(\vec{r}, \sigma)$ $\quad \vec{r} \in \mathbb{R}^3$. $\sigma \in \mathbb{Z}^2$. (spin coordinate)

Product representation of ψ with undetermined coefficients $\rightsquigarrow$ Restrict to one special product (single Slater determinant) + use fact $\min \lambda = \min \frac{\langle \psi | H | \psi \rangle}{\langle \psi, \psi \rangle}$

$\Downarrow$

Hartree-Fock equations

If you ignore one type of term in Hartree-Fock $\Rightarrow$ Schroedinger-Poisson.

If one identifies individual electrons with the functions in the product (the orbitals), then it really isn't a non-interacting approximation it's just that one is ignoring a particular type of interaction. There are still electrostatic interaction (Coloumb terms) because of the $\frac{1}{r}$ terms in the original Hamiltonian.

Full Configuration Iteraction

$$\psi = \sum_j c_j \left(\phi_1^j \phi_2^j \cdots \phi_N^j\right)$$

think of this a a generalized multi-dimensional Fourier approximation.

Equations for the coefficients ⇒ A very large matrix eigenproblem

$$\tilde{H}\vec{c} = \lambda \vec{c}$$

size grows very fast as N (# particles) and m (# orbitals) increases.

Today: 3D eigenproblem on $100 \times 100 \times 100$ grid ⇒ 10^6 basis functions. so one can use FCI.

Also, if you have problems where the potential isn't a nuclear potential then you may be able to get good results with a small # of orbitals.

(Nuclear potentials aren't very confining ⇒ ψ's fill up more space ⇒ more functions to represent them).