Follow-ups to Session 5 . . .

Session 6 Preview
Solution to cryptic error about “cb” and color axis from gnuplot: in the latest version, `set logscale` sets all the axes to logscale, including a (not used!) color scale. Fix: `set logscale xy`

Comments on table of $R_{\text{max}}$ and $N$ values for matrix diagonalization.

- Two variables, so make sure to explore them *independently*
- Two sources of error ($N$ too small for fixed $R_{\text{max}}$ and $R_{\text{max}}$ too small — what is the source of error in each case?)

Error plot versus $N$

- Why does the nice power law stop at large $N$?

Debugging loops: what can go wrong?

How far to go in the activities?
Diagonalization in coordinate representation

- Solve $l = 0$ Schrödinger equation:

\[
-\frac{\hbar^2}{2m} \frac{d^2 u^{(n)}(r)}{dr^2} + V(r)u^{(n)}(r) = E_n u^{(n)}(r) \quad \text{with} \quad u^{(n)}(r = 0) = 0
\]

- Normalization: $\int_0^\infty |u^{(n)}(r)|^2 \, dr = 1$; units: $\hbar^2/2m = 1$

- Solve as matrix problem: $H\Psi = E\Psi$ in discrete $r$ basis

- If we use the approximation with ($h = R_{\text{max}}/N$):

\[
\frac{d^2 u}{dr^2} \approx \frac{u(r + h) - 2u(r) + u(r - h)}{h^2} + O(h^2)
\]

what are the kinetic energy and potential matrices? $u_0$? $u_N$?

\[
\begin{pmatrix}
\frac{2}{\hbar^2} + V_1 & -\frac{1}{\hbar^2} & 0 & \cdots & 0 \\
-\frac{1}{\hbar^2} & \frac{2}{\hbar^2} + V_2 & -\frac{1}{\hbar^2} & & \\
0 & -\frac{1}{\hbar^2} & \ddots & \ddots & \\
& & \ddots & \ddots & -\frac{1}{\hbar^2} \\
0 & \cdots & \cdots & -\frac{1}{\hbar^2} & \frac{2}{\hbar^2} + V_{N-1}
\end{pmatrix}
\begin{pmatrix}
u_1 \\
u_2 \\
u_3 \\
\vdots \\
u_{N-1}
\end{pmatrix}
= E
\begin{pmatrix}
u_1 \\
u_2 \\
u_3 \\
\vdots \\
u_{N-1}
\end{pmatrix}
\]
Make sure you can give the simple argument of why matrix multiplication (in general) scales with the matrix size $N$ as $N^3$.

- What is the scaling for matrix-vector multiplication?

**How to you verify that the *eigen*._diagonal code is working?**

*Always* check a known case. Here: harmonic oscillator

- Why is $E_{n,l} = \hbar \omega (2n + l + 3/2)$, $n = 0, 1, 2, \cdots$ the correct spectrum? (Draw a picture of $V(r)$ and wfs for the 3d isotropic oscillator and compare to 1d.)

- [Alternative convention: $E_{n,l} = \hbar \omega (2(n-1) + l + 3/2)$, $n = 1, 2, 3, \cdots$ ]

**Boundary conditions for Schrödinger equation**

- Wavefunction $u(r)$ at endpoints: $u(0) = u(R_{\text{max}}) = 0$
- Can you see how these are built into the matrix problem?
- The real second condition is $u(\infty) = 0$. What must we do to keep this approximation under control (e.g., so the error we make is less than the worst we can tolerate)?
- Do all wave functions stick out the same? Which are more effected by $u(R_{\text{max}}) = 0$?
- For experts: Should $u(R_{\text{max}}) = 0$ raise or lower the energy?
Normalization of wave functions and eigenvectors

- If you have a vector \( |u\rangle = \{u_0, u_1, u_2, \cdots, u_N\} \), then an eigensolver will usually set \( \langle u | u \rangle = 1 = u_0^2 + u_1^2 + \cdots u_N^2 \).
- Is this the same as \( \int_0^{R_{\text{max}}} u(r)^2 dr = 1 \)?
  - No: the result will change with the mesh spacing. (Try it!)
  - But multiply \( u(r) \) by \( \sqrt{h} = \sqrt{R_{\text{max}}/N} \) and it’s ok! Why?

Expansion in a basis (here: harmonic oscillator wfs)

- Write \( u(r) = \sum_{i=0}^{D-1} c_i \phi_i(r) \) where \( \phi_i(r) \) is an HO wave function.
- cf. Fourier expansion in sines and cosines
- More abstractly: \( \langle r | u \rangle = \sum_{i=0}^{D-1} c_i \langle r | \phi_i \rangle \implies |u\rangle = \sum_{i=0}^{D-1} c_i |\phi_i\rangle \)
- Our problem: given \( D \), how to determine the best values of the \( c_i \) coefficients?
- Answer: solve it as a matrix problem! (see Session 6 notes for many details)
Your computer (laptop, desktop) divides its time doing different things (adding, multiplying, storing and retrieving data, ...).

But there are multiple processors, called “cores”, that can run different things simultaneously.

Or, parts of one program can run on more than one core at once.

If dual core (for example), than in principle we can make our code run twice as fast.

How to do this in practice? $\rightarrow$ OpenMP [is one way]
Rewriting `eigen/tridiagonal.cpp` with classes

- Compare new printout with the one from Session 5
  - We have moved details into a class and created an *interface*.
- Motivations for using classes:
  - The GSL function calls are a mess; what if we want to use the same code elsewhere or we want to use a different library?
  - Why should we need to know in the calling program how the Hamiltonian is stored? (e.g., as a vector, an array? indices from 0 to \( N - 1 \) or 1 to \( N \)?)
  - Encapsulation, data hiding, reusability.
- Always an issue: how to choose classes? Here: Hamiltonian.
  - Could be more general and use a “matrix” class.
- Declare: `Hamiltonian my_hamiltonian(N)` invokes “constructor”, which has the allocation statements. The deallocation (freeing) statements in “destructor”.
- Make functions (“methods”) for all activities (e.g., “set” and “get”).

More later! (Learn by osmosis . . . )
Solving differential equations step-by-step

- Builds on numerical derivatives

\[
\frac{dx}{dt} = f(x, t) \quad \text{e.g.,} \quad \frac{dx}{dt} = -ax \quad \text{or} \quad \frac{dx}{dt} = -bt \quad \text{or} \quad \ldots
\]

- Instruction on how to change \( x \) from one \( t \) value to a nearby one

- Goal: Given \( x(t = t_0) \), find \( x(t = t_f) \)

- Plan: diff eq tells us how to take one step at a time
  \[ \implies \text{find } x(t_0 + h), \text{ then } x(t_0 + 2h), \text{ then } x(t_0 + 3h) \text{ until } x(t_f) \]

\[
x(t_0 + h) = x(t_0) + h \left. \frac{dx}{dt} \right|_{t=t_0} + \mathcal{O}(h^2) \quad \text{from Taylor expansion} \ (1)
\]

\[
= x(t_0) + hf(x(t_0), t_0) + \mathcal{O}(h^2) \quad (2)
\]

- How to do better?
  - Explore errors, other strategies, how to pick \( h \), etc.
  - Some tricky parts: look for implementation errors if you get unexpected results