The idiots guide to finding low-lying eigenstates of the time-independent Schrödinger equation on a non-uniform lattice, using the non-Hermitian Lanczos algorithm

Jonas Gahr Sturtzel Lunde (`jonassl@berkeley.edu`)

April 4, 2019

1 Intro

1.1 Diagonalization

Cosider a $(N \times N)$ Hermitian and real (and therefore symetric) matrix H, which we want to solve the eigenvalues and eigenvectors of. Since the matrix is symetric, it is orthogonally diagonazible, as

$$D = W^* H W \quad \Rightarrow \quad H = W D W^* \tag{1}$$

where D is a $(N \times N)$ diagonal matrix, consisting of the eigenvalues of H, while W is a orthogonal, unitary $(N \times N)$ matrix, cosisting of the eigenvectors of H:

$$D = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix} \qquad W = \begin{pmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \cdots & \mathbf{h}_n \end{pmatrix}$$

1.2 Tri-diagonalization

We can also, if we wish, tridiagonalize H instead, as

$$T = V^* H V \quad \Rightarrow \quad H = V T V^* \tag{2}$$

where

$$T = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots \\ & & & \beta_{N-1} & \alpha_N \end{pmatrix} \qquad V = \begin{pmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \end{pmatrix}$$

The entries of either matrix no longer has any direct meaning, but D is tridiagonal, and V is still orthogonal and unitary.

However, T and V combined, still holds access to the eigenvalues and eigenvectors of H.

If we denote $\{\mathbf{t}_i\}$ as the eigenvectors, and $\{\theta_i\}$ as the eigenvalues of T, we have that

- If θ is an eigenvalue of T, it is an eigenvalue of H.
- If $\mathbf{t}$ is an eigenvector of T, then $\mathbf{h} = V\mathbf{t}$ is an eigenvector of H.

$$H\mathbf{h} = HV\mathbf{t} = \underbrace{(VTV^*)}_{=H} V\mathbf{t} = VT\mathbf{t} = V\theta\mathbf{t} = \theta\mathbf{h}$$

showing both that θ is an eigenvalue, and that $V\mathbf{t}$ is an eigenvector of H.

1.3

Now, consider that we instead wish to write T as a $(n \times n)$ tridiagonal matrix, where $n < N$. This will make V a $(N \times n)$ matrix. Note that V can still be orthogonal (and we will construct it as so), but it will have lost its unitary properties, such that still $V^*V = I$, but $VV^* \neq I$.

It can be shown that it is still possible to construct T as a tridiagonal matrix as

$$T = V^*HV \tag{3}$$

but note also that we no longer have $H \neq VTV^*$ due to V not being unitary anymore.

1.4 Deriving T

$$\begin{aligned} T &= V^*HV \quad |\text{Multiplying by } V^* \text{ on each side.} \\ TV^* &= V^*HVV^* \\ TV^* &= V^*HI \end{aligned}$$

2 Stuff

2.1 Discrete Laplacian Operator

Consider the laplacian operator ∇^2 acting on some descrete system represented as a vector $\mathbf{f} = [f_0,\ f_1,\ ...f_N]$. Using, for instance, a 3-point central finite difference scheme, we write the laplacian acting on as

$$\nabla^2 f_i = \frac{f_{i-1} - 2f_i + f_{i+1}}{h^2}$$

Let's try to make a matrix version of this operator D, such that $\frac{\partial^2 f}{\partial x^2} = D\mathbf{f}$. The i'th output element from a matrix-vector product is the inner product between the i'th row of the matrix, and the entire input vector. Looking at some specific output element i, we expect the picture to look something like

$$\frac{1}{h^2} D\mathbf{f} = \begin{pmatrix} \ddots & \ddots & \ddots & \ddots & \ddots \\ \ddots & 1 & -2 & 1 & \ddots \\ \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix} \begin{pmatrix} \vdots \\ f_{i-1} \\ f_i \\ f_{i+1} \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots \\ f_{i-1} - 2f_i + f_{i+1} \\ \vdots \end{pmatrix} \tag{4}$$

The point is that the derivative of the ith element is decided by the ith row of the matrix.

Now, expanding this to the entire matrix, we get a tridiagonal matrix on the form

$$\frac{1}{h^2} D = \begin{pmatrix} -2 & 1 & 0 & 0 & \cdots \\ 1 & -2 & 1 & 0 & \cdots \\ 0 & 1 & -2 & 1 & \cdots \\ \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix} \tag{5}$$

Generally, if an operator can be written as

$$\mathcal{O} f_i = \sum_j C_{ij} f_j$$

it has a matrix

$$O = C_{ij} = \begin{pmatrix} C_{00} & C_{01} & C_{02} & \cdots \\ C_{10} & C_{11} & C_{12} & \cdots \\ C_{20} & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & \ddots \end{pmatrix}$$

3 1D Radial Deuteron Potential

3.1 Open boundary conditions

Open boundary conditions works by simply considering there to be no force applied past the endpoints. This is equivalent to putting a point outside each boundary that mirrors the potential of the last poin inside our system.

4 3D Deuteron Potential

4.1 Periodic boundary conditions

Let us define $X_{x,y,z}$ or just $[x, y, z]$, for $x, y, z \in [0,\ N-1]$ as our gridpoints on our 3D $N \times N \times N$ grid.

We will employ periodic boundary conditions, meaning that the ends "wrap around" each egde, such that the neighboring point of each edgepoint is the opposite edgepoint. For a grid of N points in each dimension, from 0 to $N-1$, this can be written as

$$[-1,\ y,\ z] = [N-1,\ y,\ z] \qquad [N,\ y,\ z] = [0,\ y,\ z]$$

where the N and -1 points are artificial points, only used when considering the neighbors of the edgepoints, and not considered part of the actual system.

4.2 Unraveled indexes

Consider a 3D grid of N points in each of the 3 dimensions. The N^3 points needs to be stored in a 1D array, X_i, or simply $[i]$, with one unraveled index i. We will store them in (x,y,z) order, such that $i \in (0,\ N-1)$ corresponds to the first row of x-values, at $y = 0$ and $z = 0$.

We define the conversion between $[x, y, z]$ and $[i]$ as:

$$\begin{aligned}
i &= x + Ny + N^2 z \\
x &= i\%N) \\
y &= (i//N)\%N \\
z &= i//N^2
\end{aligned}$$

where % is the modulo operator and // is an interger division.

4.3 The 7-point stencil

5 Python

5.1 Sparse Matrices

5.2 Memory Consideration

Consider a system of size $M = N^3$, where each dimension has a grid of N points. If we use a l-point Laplacian stencil, the hamiltonian will consist of $l \cdot N^3$ elements. For a 7-point stencil, and a typical $N = 100$, we get $7 \cdot 100^3 = 7 \cdot 10^6$.

The hamiltonian is stored as a sparse matrix.

The main limiting factor is going to be the storage of the V-matrix, holding the basis of the generated matrix. The V matrix has a size of $M \cdot n = N^3 \cdot n \cdot 8\,\mathrm{B}$, which is typically of the size $100^3 \cdot 400 = 3.2\,\mathrm{GB}$, when using 64bit floats.

6 Irregular Lattice - Laplacian

6.1 Irregular Lattice

Now, suppose you have a lattice of some dimensionality (let's say 3D). The potential we wish to study often changes rapidly in some areas, and is very uniform in others. Using a uniform lattice, we are in other words wasting lots of computational resources on un-interesting areas of the potential. We could instead let our lattice be non-uniform, with higher lattice-density in important areas.

We wish our lattice to be irregularly spaced, but constrain it to fit on some *fine lattice*, of spacing s. In other words, the spacing between each lattice point must be a multiple of s. In the simplest case, we could imagine some box around the origin where the lattice spacing is s:

$$\Delta r = 1 \cdot s, \quad -c \leq r \leq c$$

while outside the box, the lattice spacing is doubled:

$$\Delta r = 2 \cdot s, \quad |r| > c$$

6.2 Laplacian Error

The main impact on the lattice spacing is the avaliability of points to sample the Laplacian from. For irregular lattice-spaciing, some algorithm needs to be developed to pick what points are included, and how they are weighted. For now, we will focus on an algorithm for picking the optimal spacing, with regard to the laplacian error.

Let's say you have some function on the form

$$\Psi(r) = Ae^{\kappa r} \left(+Be^{-\kappa r}\right) \tag{6}$$

for some real or complex κ. This will include both all exponential/decay functions, and all harmonic functions (and therefore most wavefunctions we are interested in). In a standard Q.M. well potential, we will get $\kappa = i\sqrt{2m|E_0|}/\hbar$ for a "trapped" solution (particle has less energy than well-depth, $E_0 < 0$), and $\kappa = \sqrt{2mE}/\hbar$ for "free" solutions (particle has more energy than well-depth $E_0 > 0$).[1]

We will need to evaluate the Laplacian of this function at each latticepoint, using some finite difference scheme. Let's consider the finite difference Laplacian approximation of the 3D 7-point stencil with a uniform lattice spacing a:

$$\nabla^2\Psi(r) = \frac{\Psi(r-a) + \Psi(r+a) - 2\Psi(r)}{a^2} + \epsilon \tag{7}$$

where ϵ is the error we have introduced with our approximation.

Now, insert the wavefunction from 6:

$$\nabla^2\Psi(r) = \frac{A}{a^2}\left[e^{\kappa r}\left(e^{-\kappa a} + e^{i\kappa a}\right) - 2e^{\kappa r}\right] + \epsilon \tag{8}$$

Now, on the left hand side remains the analytical expression for the Laplacian. Inserting for the wavefunction, and doing a double derivation in regards to r, gives $\Psi''(r) = \kappa^2 Ae^{\kappa a} \ = \kappa^2\Psi(r)$. Inserting for this, and pulling $\Psi(r) = e^{\kappa r}$ out on the right side, gives

$$\kappa^2\Psi(r) = \frac{1}{a^2}\Psi(r)\left[e^{-\kappa a} + e^{i\kappa a} - 2\right] + \epsilon \tag{9}$$

Let's Taylor-expand $e^{\pm\kappa a}$ in powers of κa, giving

$$\begin{aligned} e^{\kappa a} &= 1 + \kappa a + \frac{(\kappa a)^2}{2!} + \frac{(\kappa a)^3}{3!} + \frac{(\kappa a)^4}{4!} + \mathcal{O}(a^5) \\ e^{-\kappa a} &= 1 - \kappa a + \frac{(\kappa a)^2}{2!} - \frac{(\kappa a)^3}{3!} + \frac{(\kappa a)^4}{4!} + \mathcal{O}(a^5) \end{aligned}$$

[1]Note that this is assuming the well goes to 0 at infinity. If it goes to V_0, E_0 will instead be $E_0 = E_0' - V_0$.

Inserting this back, we see that the first order terms cancells with the 2, and the thrid order term cancels with each other. We are left with

$$\kappa^2\Psi(r) = \frac{1}{a^2}\Psi(r)\left[2\frac{\kappa^2a^2}{2} + 2\frac{\kappa^4a^4}{4!} + \mathcal{O}(a^5)\right] + \epsilon \tag{10}$$

We are left with the final error-estimate:[2]

$$\frac{\epsilon}{\Psi''(r)} = \frac{\kappa^2a^2}{12} \tag{11}$$

[2](remember that $\Psi''(r) = \kappa^2\Psi(r)$)

7 A Slightly Simplified Irregular Lattice

7.1 Point Spacing

Consider an irregularly spaced lattice, which fits on some fine grid of spacing s. We will have to decide, by some scheme, what kind of spacing $\Delta x = a \cdot s,\ a \in [1, 2, 3, ...]$ each point on the lattice should have. One simplification of the general irregular grid, is dividing the grid into a set of b boxes, which each has a uniform grid of some lattice spacing Δx_b. This will significantly simplify calculations of the Lapacians when inside each box, as we can employ a standard, symetric stencil. The edges will be to be handled as special cases.

7.2 Further Simplifying - Handling Edge Stencils

Yet anothing simplification of the box-approach can be done, which will allow us to also employ symetric stencils at the edges. Consider an edge between two boxes, where the point-spacing changes. We could simply employ whatever stencil we were using at the lower density box onto both boxes, when handling the box edges. This will mean ignoring some close points on the higher density box, but we keep the symetry.

Now, for this to work, we must require that all boxes has a spacing of $2^n \cdot s,\ n \in [0, 1, 2, ...]$. If this were not true, and we considered the edge between a box with spacing of, i.e., $2s$ and $3s$, there would be no way of ignoring points in the higher density box to match the lower density one.[3] While if we are looking at the edge between one $2s$ box, and one $8s$ box, we could simply ignore 3 out of every 4 points in the higher density box, and we would be left with a symetric stencil.

7.3 Setup

One clear advantage that might seem lost when the grid was made irregular was the simplicity of neighbor-searching, due to the neighboring points in each dimension being predictibly layed out in memory. In a totally chaotic grid, one would have little choice but to store points in rather abritrary order, meaning nearest-neighbor searching would involve some form of calculating distances[4]. Another advantage of the box-implementation is that now any box will have a predictable memory layout, and neighbor-searching inside a box is simple yet again. If the second simplification mentioned above is also implemented, nearest-neighbor edge searching can also be set up in similar fashion.

8 Picking a stencil for non-rotationally symtric points

When the points have 90 rotational symetry, we can construct all the common stencil groups we're used to on a uniform grid. There will however be cases where no such construction is possible. It will still hold 180 degrees rotationally symetry, so we will restict our stencils to this instead. This does however involve coming up with new, less common, sets of stencils. One important consideration that should be kept in mind in these cases is to preserve angular density of points, such that no direction holds more importance in the laplacian. One important note is that, even though the

[3]We could find a lowest common denominator and ignore some points from both boxes, but that would be wasteful.
[4]Although a clever datastructure could assist in this.