ML2R Coding Nuggets
Numerically Solving the Schrödinger Equation (Part 2)

Christian Bauckhage*
Machine Learning Rhine-Ruhr
University of Bonn
Bonn, Germany

ABSTRACT

We revisit the problem of numerically solving the Schrödinger equation for a one-dimensional quantum harmonic oscillator. We reconsider our previous finite difference scheme and discuss how higher order finite differences can lead to more accurate solutions. In particular, we will consider a five point stencil to approximate second order derivatives and implement the approach using *SciPy* functions for sparse matrices.

1 INTRODUCTION

Previously [1], we considered a simple numerical scheme for solving the time-independent Schrödinger equation for a one-dimensional quantum harmonic oscillator (in atomic units where $m = 1, \hbar = 1$)

$$\left(-\frac{1}{2}\frac{d^2}{dx^2} + \frac{1}{2}x^2\right)\Psi(x) = \hat{H}\,\Psi(x) = E\,\Psi(x) \tag{1}$$

The key idea was to confine and discretize the position variable x. We thus introduced an interval $[-l/2, +l/2]$ of length l and a grid of N equally spaced points $-l/2 \leq x_i \leq +l/2$. This allowed us to approximate the wave function $\Psi(x)$ in terms of an N-dimensional vector $\boldsymbol{\psi}$ and the Hamiltonian $\hat{H}$ in terms of an $N \times N$ matrix $\boldsymbol{H}$ so that the Schrödinger equation became

$$\boldsymbol{H}\,\boldsymbol{\psi} = E\,\boldsymbol{\psi} \tag{2}$$

We then saw that it is easy to set up and solve this eigenvalue / eigenvector problem using standard *NumPy* functions.

However, when we looked at the accuracy of the resulting solutions, we found it to be rather good but arguably not as good as one would wish for. In this note, we therefore revisit the approach and discuss the idea of incorporating higher order finite differences. We also address the following issue: If the number N of grid points is large, dense representation of the $N \times N$ matrix $\boldsymbol{H}$ will have a very large memory footprint. The implementation in our previous note may thus not be practically feasible if very dense grids are to be considered. However, since $\boldsymbol{H}$ is *sparse*, we can work with sparse matrix representations and we will discuss the use of corresponding functionalities in *SciPy*'s `sparse` module.

As always, we will first review the necessary theory (in section 2) and then present practical implementations (in section 3).

Throughout, we assume that readers are familiar with the content of our previous note [1]. Those who want to experiment with our code snippets in section 3 should have experience with *NumPy* and *SciPy* [5] and only need to

```
import numpy as np
import scipy.sparse as sprs
import scipy.sparse.linalg as sprsla
```

*0000-0001-6615-2128

2 THEORY

In [1], we saw that modeling the domain of the continuous position variable x of a quantum harmonic oscillator in terms of a discrete grid of $N > 1$ equally spaced points $-l/2 \leq x_i \leq +l/2$ allows for a finite difference approximation of the second derivative of the wave function at the grid points. In particular, we considered

$$\frac{d^2}{dx^2}\Psi(x_i) \approx \frac{\Psi(x_{i-1}) - 2\,\Psi(x_i) + \Psi(x_{i+1})}{\delta x^2} \tag{3}$$

where $\delta x = x_i - x_{i-1} = 1/N{-}1$ is a number between 0 and 1. In what follows, we will show that

$$\frac{d^2}{dx^2}\Psi(x_i) \approx \frac{-\Psi(x_{i-2}) + 16\,\Psi(x_{i-1}) - 30\,\Psi(x_i) + 16\,\Psi(x_{i-1}) - \Psi(x_{i-2})}{12\,\delta x^2} \tag{4}$$

gives a better approximation of the second derivative.

In order to derive this result, we will first have a closer look at the rational behind the approximations in (3) and (4). To emphasize that the underlying ideas are indeed general, we will work with a general function $f : \mathbb{R} \to \mathbb{R}$.

2.1 Finite Differences and First Derivatives

To begin with, we look at finite difference approximations of the first derivative $f'(x)$ of a $f(x)$. To this end, we consider the following truncated Taylor series expansion

$$f(x + \delta x) \approx f(x) + \delta x\, f'(x) + \frac{1}{2}\,\delta x^2\, f''(x)$$

If we rearrange this expression, we obtain

$$\frac{f(x + \delta x) - f(x)}{\delta x} - f'(x) \approx \delta x\,\frac{f''(x)}{2}$$

which tells us that the so called forward difference

$$\frac{f(x + \delta x) - f(x)}{\delta x}$$

approximates $f'(x)$ but comes with an approximation error that is proportional to the quantity δx.

To obtain a better approximation of the first derivative, we next consider the following two expressions

$$f(x + \delta x) \approx f(x) + \delta x\, f'(x) + \frac{1}{2}\,\delta x^2\, f''(x) + \frac{1}{6}\,\delta x^3\, f'''(x)$$

$$f(x - \delta x) \approx f(x) - \delta x\, f'(x) + \frac{1}{2}\,\delta x^2\, f''(x) - \frac{1}{6}\,\delta x^3\, f'''(x)$$

If we subtract the second one from the first one and rearrange the resulting expression, we obtain

$$\frac{f(x + \delta x) - f(x - \delta x)}{2\,\delta x} - f'(x) \approx \delta x^2\,\frac{f'''(x)}{6}$$

which tells us that the so called central difference

$$\frac{f(x + \delta x) - f(x - \delta x)}{2\,\delta x}$$

is an approximation of $f'(x)$ whose error is proportional to δx^2. And, since $0 < \delta x < 1$, we have $\delta x^2 < \delta x$ which is to say that the central difference gives a better approximation of $f'(x)$ than the forward difference.

To obtain even better (centered) finite difference approximations of $f'(x)$ we have to consider more terms in the Taylor expansions as well as larger neighborhoods around x. For instance, if we work with

$$f(x+1\cdot\delta x) \approx \sum_{n=0}^{5} (+1)^n\, \delta x^n\, \frac{f^{(n)}(x)}{n!} \tag{5}$$

$$f(x-1\cdot\delta x) \approx \sum_{n=0}^{5} (-1)^n\, \delta x^n\, \frac{f^{(n)}(x)}{n!} \tag{6}$$

$$f(x+2\cdot\delta x) \approx \sum_{n=0}^{5} (+2)^n\, \delta x^n\, \frac{f^{(n)}(x)}{n!} \tag{7}$$

$$f(x-2\cdot\delta x) \approx \sum_{n=0}^{5} (-2)^n\, \delta x^n\, \frac{f^{(n)}(x)}{n!} \tag{8}$$

we find that $8\cdot\big((5)-(6)\big)-\big((7)-(8)\big)$ cancels out any terms involving δx^2 and δx^3 and yields

$$\frac{-f(x+2\,\delta x)+8\,f(x+1\,\delta x)-8\,f(x-1\,\delta x)+f(x+2\,\delta x)}{12\,\delta x}$$

as an $O\big(\delta x^4\big)$ approximation to $f'(x)$.

2.2 Finite Differences and Second Derivatives

Approximations of higher order derivatives can be obtained in a similar fashion. For instance, for the second derivative $f''(x)$, which is of major interest in the context of Schrödinger equations, we may consider

$$f(x+1\cdot\delta x) \approx \sum_{n=0}^{4} (+1)^n\, \delta x^n\, \frac{f^{(n)}(x)}{n!}$$

$$f(x-1\cdot\delta x) \approx \sum_{n=0}^{4} (-1)^n\, \delta x^n\, \frac{f^{(n)}(x)}{n!}$$

to get

$$\frac{f(x+\delta x)-2\,f(x)+f(x-\delta x)}{\delta x^2} - f''(x) \approx \delta x^2\,\frac{f^{(4)}(x)}{24}$$

This tells us that the finite difference approximation we considered in (3) actually is an $O\big(\delta x^2\big)$ approximation of the second derivative of the wave function of the quantum harmonic oscillator.[1]

[1]Here is an interesting side note: In [1], we numerically solved the Schrödinger equation in (1) on discrete grids of $N_1 = 1001$ and $N_2 = 2001$ points in the interval $[-l/2, +l/2]$. We observed that the eigenenergies we obtained from working with the larger and thus denser grid were closer to the theoretically prescribed values. Given our present discussion so far, this now makes mathematical sense. As $\delta_2 x = 1/2000 < \delta_1 x = 1/1000$, we have $\delta_2 x^2 < \delta_1 x^2$ so that the finite difference approximation based on 2001 grid points provides a more accurate approximation of the second derivative of the wave function the the one based on 1001 grid points.

Again, we can do better. For instance, if we consider

$$f(x\pm 1\cdot\delta x) \approx \sum_{n=0}^{5} (\pm 1)^n\, \delta x^n\, \frac{f^{(n)}(x)}{n!} \tag{9}$$

$$f(x\pm 2\cdot\delta x) \approx \sum_{n=0}^{5} (\pm 2)^n\, \delta x^n\, \frac{f^{(n)}(x)}{n!} \tag{10}$$

we find that

$$16\,f(x+1\,\delta x)+16\,f(x-1\,\delta x) \approx 32\,f(x)+16\,\delta x^2 f''(x)$$

and

$$f(x+2\,\delta x)+f(x-2\,\delta x) \approx 2\,f(x)+4\,\delta x^2 f''(x)$$

Subtracting the second expression from the first and rearranging the resulting term then establishes that

$$\frac{-f(x+2\,\delta x)+16\,f(x+1\,\delta x)-30\,f(x)+16\,f(x-1\,\delta x)-f(x-2\,\delta x)}{12\,\delta x^2}$$

is an $O\big(\delta x^4\big)$ approximation to $f''(x)$. In other words, the finite difference approximation we presented in (4) actually is an $O\big(\delta x^4\big)$ approximation of the second derivative of the wave function of the quantum harmonic oscillator and hence indeed more precise than the on in (3).

2.3 Some Terminology and Context

The two approximations in (3) and (4) are given w.r.t. a regular grid of points. The one in (3) involves three points x_{i-1}, x_i, x_{i+1} to approximate the second derivative of Ψ at x_i. The one in (4) involves five points $x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2}$. Such neighborhoods around a point x_i are also called stencils. As the stencil in (3) involves three points, its is called a three-point stencil; as the one in (4) involves five points, its is called a five-point stencil.

There also is a connection to signal processing: The two sets of coefficients $1/\delta x^2 \cdot \big\{1, -2, 1\big\}$ and $1/12\,\delta x^2 \cdot \big\{-1, 16, -30, 16, -1\big\}$ in (3) and (4) are sometimes called convolution kernels. Indeed, if we define the discrete functions $\Psi[i] = \Psi(x + i\cdot\delta x), i \in \mathbb{Z}$ and

$$g_3[i] = \begin{cases} -\frac{2}{\delta x^2} & \text{if } i = 0 \\ \frac{1}{\delta x^2} & \text{if } i = \pm 1 \\ 0 & \text{otherwise} \end{cases}$$

$$g_5[i] = \begin{cases} -\frac{30}{12\,\delta x^2} & \text{if } i = 0 \\ \frac{16}{12\,\delta x^2} & \text{if } i = \pm 1 \\ -\frac{1}{12\,\delta x^2} & \text{if } i = \pm 2 \\ 0 & \text{otherwise} \end{cases}$$

we have

$$(3) \Leftrightarrow \big(\Psi * g_3\big)[i] = \sum_{j=-1}^{1} \Psi[i]\, g_3[i-j]$$

$$(4) \Leftrightarrow \big(\Psi * g_5\big)[i] = \sum_{j=-2}^{2} \Psi[i]\, g_5[i-j]$$

2.4 Back to Solving the Schrödinger Equation

If we use a grid of $N > 1$ equally spaced points $-l/2 \leq x_i \leq +l/2$ to discretize the position variable x of a quantum harmonic oscillator, we have $\delta x = x_i - x_{i-1} = 1/N{-}1$ and can discretize the Schrödinger equation in 1.

To this end, we introduce an N-dimensional vector $\boldsymbol{\psi}$ whose components are given by $\psi_i = \Psi(x_i)$ and represent the Hilbert space operator $\hat{H} = \hat{T} + \hat{V}$ in terms of an $N \times N$ matrix. Just as we did in [1], we represent the potential energy operator $\hat{V} = 1/2\, x^2$ as a diagonal matrix

$$\boldsymbol{V} = \frac{1}{2}\begin{bmatrix} x_1^2 & & \\ & \ddots & \\ & & x_N^2 \end{bmatrix} \tag{11}$$

Contrary to [1] where we used the 3-point stencil in (3) to represent the kinetic energy operator $\hat{T} = -d^2/2\,dx^2$ as a tridiagonal matrix[2]

$$\boldsymbol{T}_3 = -\frac{1}{2\,\delta x^2}\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix} \tag{12}$$

we will now use the 5-point stencil in (4) to represent $\hat{T}$ in terms of a pentadiagonal matrix

$$\boldsymbol{T}_5 = -\frac{1}{24\,\delta x^2}\begin{bmatrix} -30 & 16 & -1 & & & & \\ 16 & -30 & 16 & -1 & & & \\ -1 & 16 & -30 & 16 & -1 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & -1 & 16 & -30 & 16 & -1 \\ & & & -1 & 16 & -30 & 16 \\ & & & & -1 & 16 & -30 \end{bmatrix} \tag{13}$$

Using these definitions, the action of the Hamiltonian $\hat{H}$ of our quantum harmonic oscillator can approximated as

$$\hat{H}\,\Psi(x) \approx \boldsymbol{H}\,\boldsymbol{\psi} \tag{14}$$

where $\boldsymbol{H} = \boldsymbol{T}_5 + \boldsymbol{V}$. By the same token, we can approximate the right hand side of the Schrödinger equation as

$$E\,\Psi(x) \approx E\,\boldsymbol{\psi} \tag{15}$$

so that we once again obtain the discretized Schrödinger equation in (2). Next, we discuss how to solve this eigenvalue / eigenvector problem using *SciPy*'s sparse matrix functionalities.

3 PRACTICE

In [1], we already used *NumPy* to solve the discretized Schrödinger equation in (2) for the eigenenergies and eigenstates of the quantum harmonic oscillator. There, we worked with $\boldsymbol{H} = \boldsymbol{T}_3 + \boldsymbol{V}$ and now could use the same recipe to work with the numerically more accurate matrix $\boldsymbol{H} = \boldsymbol{T}_5 + \boldsymbol{V}$.

However, our approach in [1] was wasteful because, dealing with a grid of size N, we considered a *dense* representation of the $N \times N$ matrix $\boldsymbol{H}$. While it is computationally easy to determine spectral decompositions of band matrices (i.e. matrices whose non-zero elements only occur on the main diagonal and on diagonals on either side), some computers may have difficulties storing a dense $\boldsymbol{H}$ if N is chosen to be large, say, $N \gg 10^5$. Here, we therefore emphasize that band matrices are *sparse*. For example, for the pentadiagonal matrix $\boldsymbol{H} = \boldsymbol{T}_5 + \boldsymbol{V}$, there are only $N + 2(N-1) + 2(N-2) = 5N - 6 \ll N^2$ non-zero elements. This does of course suggest to implement sparse matrix solutions.

[2]Recall that this matrix can be understood as the Laplacian of a weighted line graph. Since Laplacians like this occur in the context of spectral clustering [2–4, 7, 8], there is a close connection to methods known to machine learning experts.

Listing 1: solving the discretized Schrödinger equation (2)

```
 1  l = 12.
 2  N = 1001
 3  num = 11
 4
 5  xs = np.linspace(-l/2, +l/2, N)
 6  vs = 0.5 * xs**2
 7
 8  dx = xs[1]-xs[0]
 9
10  d0 = -30. * np.ones(N)
11  d1 =  16. * np.ones(N-1)
12  d2 = - 1. * np.ones(N-2)
13
14  matT = -sprs.diags([d0, d1, d1, d2, d2],
15                     [ 0, +1, -1, +2, -2]) / (2 * 12 * dx**2)
16
17  matV = sprs.diags([vs], [0])
18
19  matH = matT + matV
20
21  es, psis = sprsla.eigsh(matH, k=num, which='SM')
22
23  psis /= np.sqrt(np.sum(psis**2, axis=0))
24
25  dens = np.abs(psis)**2
```

Our recipe in Listing 1 therefore involves functions available in the `scipy.sparse` module. In lines 1 and 2, we again initialize the parameters l and N. Line 3 sets a new parameter `num` whose role will be discussed soon.

Just as in [1], we then initialize two *NumPy* arrays `xs` and `vs` which represent grid points x_i and corresponding potential energies $V(x_i) = \frac{1}{2}\,x_i^2$. Line 8, too, is already known from our previous solution and sets the grid point distance δx.

The content of lines 10–12 is new. Here we initialize thrre *NumPy* arrays `d0`, `d1`, and `d2` which represent the main diagonal and the ± 1 and ± 2 diagonals of the pentadiagonal matrix $\boldsymbol{T}_5$ in (13).

To implement this matrix as a sparse matrix, we apply the *SciPy* function `diags`. In lines 14 and 15, we cal this function with two parameters: The first is a list of *NumPy* arrays containing numbers to be put on diagonals and the second is a list of integers indicating which diagonals are to be filled.

Method `diags` reoccurs in line 17 where we use it to implement matrix $\boldsymbol{V}$ in (11) as a sparse matrix as well.

Since `matT` and `matV` now contain sparse representation of matrices of commensurable sizes, we may simply add them, as we do in line 19, to obtain a sparse representation `matH` of the problem Hamiltonian $\boldsymbol{H}$.

Once `matH` is available, we can compute its spectral decomposition. Since `matH` represents a sparse matrix, we use a function available in `scipy.spase.linalg`; since `matH` also represents a Hermitian matrix, this function is `eigsh`. Line 21 demonstrates its use: The first parameter passed to `eigsh` is the matrix whose eigenvalues and eigenvectors are to be determined. The second

Table 1: Analytical- and numerical eigenenergies of a QHO

n	E_n (analytical)	E_n (numerical) 3-point stencil	E_n (numerical) 5-point stencil
0	0.5	0.499995	0.500000
1	1.5	1.499977	1.500000
2	2.5	2.499941	2.500000
3	3.5	3.499887	3.500000
4	4.5	4.499815	4.500000
5	5.5	5.499726	5.500000
6	6.5	6.499618	6.500000
7	7.5	7.499493	7.500001
8	8.5	8.499355	8.500008
9	9.5	9.499231	9.500045
10	10.5	10.499231	10.500227

parameter `k` indicates how many eigenvalues / eigenvectors are to be computed. Here, we pass `num`, the parameter we initialized in line 3. The third parameter `which` indicated which eigenvalues / eigenvectors are to be determined. Here, we set it to `'SM'` which is to say that we are interested in those eigenvalues (and their corresponding eigenvectors) that are smallest in magnitude.

As a result, line 21 produces a 1D array `es` of the `num` smallest eigenvalues E_n of $\boldsymbol{H}$ and a 2D array `psis` of corresponding eigenvectors $\boldsymbol{\psi}_n$. Just as in [1], we finally normalize the latter such that $\|\boldsymbol{\psi}_n\| = 1$ (in line 23) and compute the respective probability densities (in line 25).

In [1], we said that the analytical solutions of the eigenenergies of a quantum harmonic oscillator (QHO) are $E_n = n + 1/2$ [6]. We used this fact to asses the quality of the results we obtained from the numerical solution based on the 3-point stencil matrix $\boldsymbol{H} = \boldsymbol{T}_3 + \boldsymbol{V}$. Now, we can use the code in Listing 1 to include the results obtained from working with the 5-point stencil matrix $\boldsymbol{H} = \boldsymbol{T}_5 + \boldsymbol{V}$ into our comparison.

Table 1 compares analytical and numerical results for the first $0 \leq n \leq 10$ eigenenergies E_n where all numerical results were obtained from working with grids of $N = 1001$ points in the interval $[-l/2, +l/2]$ with $l = 12$. Looking at these numbers, we conclude that the higher order finite difference scheme that invokes a 5-point stencil in approximating the second derivative of the wave function does indeed produce more accurate results than the 3-point stencil version we considered in our last note.

4 SUMMARY AND OUTLOOK

In this note, we revisited the problem of numerically solving the Schrödinger equation for a one-dimensional quantum harmonic oscillator. We extended our previous recipe in two regards: First of all, we considered a more accurate finite difference approximation of the second derivative that features prominently in the Schrödinger equation. Second of all, we pointed out that the discretized version of the Hamiltonian of the system is a sparse matrix and thus applied *SciPy* methods for sparse matrix computations.

In addition to the spectral decomposition methods we considered so far, there are many other numerical approaches towards solving Schrödinger equations. Some of these will be discussed in future notes. This will provide us with opportunities of getting to know, say, more specialized *SciPy* functions for solving differential equations.

ACKNOWLEDGMENTS

This material was produced within the Competence Center for Machine Learning Rhine-Ruhr (**ML2R**) which is funded by the Federal Ministry of Education and Research of Germany (grant no. 01IS18038C). The authors gratefully acknowledge this support.

REFERENCES

[1] C. Bauckhage. 2021. *ML2R Coding Nuggets: Numerically Solving the Schrödinger Equation (Part 1).* Technical Report. MLAI, University of Bonn.
[2] I.S. Dhillon, Y. Guan, and B. Kulis. 2004. Kernel k-means, Spectral Clustering and Normalized Cuts. In *Proc. KDD*. ACM.
[3] J. Kunegis, D. Fay, and C. Bauckhage. 2010. Network Growth and the Spectral Evolution Model. In *Proc. CIKM*. ACM.
[4] J. Kunegis, D. Fay, and C. Bauckhage. 2013. Spectral Evolution in Dynamic Networks. *Knowledge and Information Systems* 37, 1 (2013).
[5] T.E. Oliphant. 2007. Python for Scientific Computing. *Computing in Science & Engineering* 9, 3 (2007).
[6] R. Shankar. 1994. *Principles of Quantum Mechanics* (2nd ed.). Springer.
[7] J. Shi and J. Malik. 2000. Normalized Cuts and Image Segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence* 22, 8 (2000).
[8] U. von Luxburg. 2007. A Tutorial on Spectral Clustering. *Statistics and Computing* 17 (2007).