Computational Physics Homework 9

Cristopher Cerda Puga

Matrix representation of the Hamiltonian operator for the harmonic oscillator

The Hamiltonian operator for the harmonic oscillator is given by

$$\hat{H} = -\frac{\hbar^2}{2m}\frac{\mathrm{d}^2}{\mathrm{d}x^2} + \frac{1}{2}m\omega^2 x^2 \tag{1}$$

The representation of the second derivative, we can use the central difference approximation.

Recall that the first derivative of a function can be a approximated as:

$$f'(x) \approx \frac{f(x) - f(x+h)}{h}$$

or using the first order central difference:

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

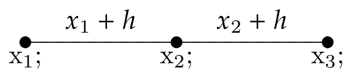
Now, for the second order central difference:

$$\begin{aligned} f(x+h) &= f(x) + f'(x)h + \frac{1}{2}f''(x)h^2 + O(3) \\ f(x-h) &= f(x) - f'(x)h + \frac{1}{2}f''(x)h^2 + O(3) \end{aligned}$$

adding the two expressions above and dividing by h^2 we get

$$f''(x) \approx \frac{f(x-h) - 2f(x) + f(x+h)}{h^2} \tag{2}$$

Let us define h as the spacing between the grid points as shown in the next figure.



Then $f(x_2 - h) = f(x_1)$ and $f(x_2 + h) = f(x_3)$. And we can represent the second derivative as the next three-diagonal matrix:

$$\begin{pmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ 0 & 1 & -2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \end{pmatrix} \tag{3}$$

which acts over the discretized space of points as follows:

$$\begin{pmatrix} f''(x_0) \\ f''(x_1) \\ f''(x_3) \\ \vdots \\ f''(x_n) \end{pmatrix} \approx \frac{1}{h^2} \begin{pmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ 0 & 1 & -2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \end{pmatrix} \begin{pmatrix} f(x_0) \\ f(x_1) \\ f(x_3) \\ \vdots \\ f(x_n) \end{pmatrix}. \tag{4}$$

And therefore the Hamiltonian operator can be represented as (setting $\hbar$ and m equal to 1):

$$\hat{H} \approx \frac{1}{2h^2}\begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ 0 & -1 & 2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & -1 & 2 \end{pmatrix} + \begin{pmatrix} V(x_0) & 0 & 0 & \cdots & 0 \\ 0 & V(x_1) & 0 & \cdots & 0 \\ 0 & 0 & V(x_2) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & V(x_n) \end{pmatrix} \tag{5}$$

where $V(x_i) = \frac{1}{2}m\omega^2 x_i^2$. Implementing this into code:

```python
def construct_hamiltonian(N, hbar, m, omega):
    # defining the grid
    x = np.linspace(-100,100,N)
    # find the spacing between the grid
    h = x[1] - x[0]

    kinetic_energy_matrix = (1/h**2)*(hbar**2/2*m)*(np.diag(-1*np.ones(N-1), k=-1) +
        np.diag(2*np.ones(N),k=0) + np.diag(-1*np.ones(N-1), k=1))
    harmonic_potential_matrix = np.diag((0.5*m*omega**2*x**2)*np.ones(N))
    hamiltonian_matrix = harmonic_potential_matrix + kinetic_energy_matrix
    eigenvalues,eigenvectors = np.linalg.eigh(hamiltonian_matrix)

    return eigenvalues,eigenvectors
```

After defining some spacing we can compare the numerical eigenvalues with the analytical ones:

```python
N = 2000
hbar = 1
omega = 1
m = 1

eigenvalues_list = list(eigenvalues[1:20]) # taking just the first 19 eigenvalues (
    without counting the first).

n = np.linspace(1,19,19)

analytic_eigenvalue = []
# Analytic eigenvalues
for k in n:
    analytic_eigenvalue.append(hbar*omega*(0.5+k))

plt.scatter(n,eigenvalues_list,label='Numerical eigenvalues')
plt.scatter(n,analytic_eigenvalue,color='red',label='Analytic eigenvalues')
plt.xlabel('n')
plt.ylabel('Eigenvalues')
plt.legend()
plt.show()
```

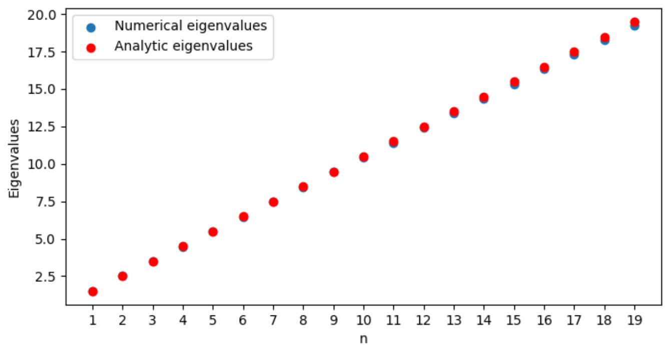


Figure 1: Eigenvalues of the quantum harmonic oscillator.

You can clearly see a small deviation as n increases. This can be seen more strictly in the form of a table.

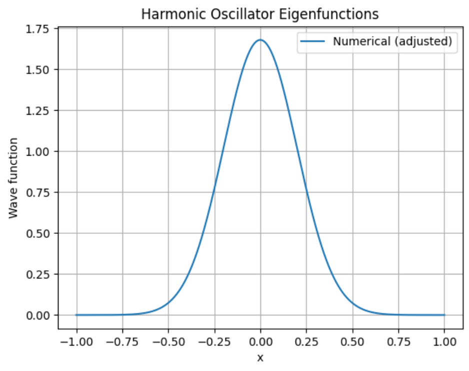
n	Numerical	Analytical
1	1.4984341694607266	1.5
2	2.4959265541491487	2.5
3	3.4921617670804905	3.5
4	4.487137414997147	4.5
5	5.480851089403741	5.5
$\vdots$	$\vdots$	$\vdots$
9	9.443037339638733	9.5
10	10.430404474363677	10.5
$\vdots$	$\vdots$	$\vdots$
19	19.258948867227513	19.5

Table 1: Eigenvalues of the quantum harmonic oscillator.

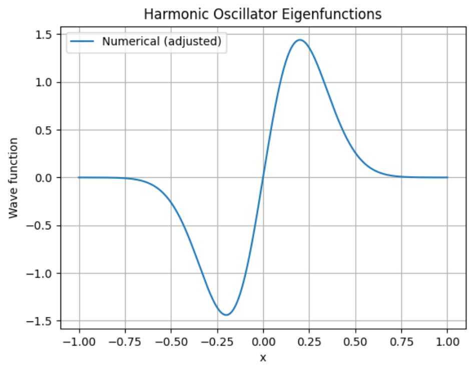
To increase accuracy as n increases, you can increase the number of intervals in the code.

Another thing we may be interested in are the Hamiltonian eigenvectors, which can be found quite easily as shown in the listings above. It can be noted that the eigenvectors resulting from this method could not be in the correct phase, so in order to have a physical meaning for our problem we must adjust the phase.

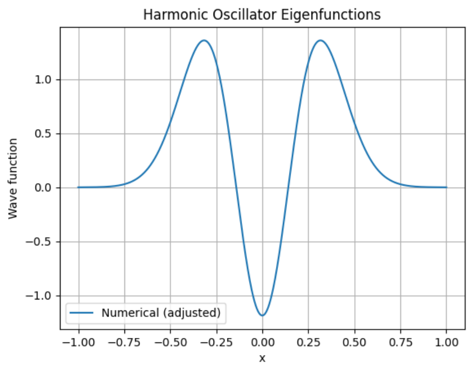
```
eigenvalues,eigenvectors = construct_hamiltonian(N, hbar, m, omega)

# x values for the plot
x_values = np.linspace(-1, 1, N)

plt.plot(x_values,eigenvectors[:,0],label='Numerical (adjusted)')
plt.title('Harmonic Oscillator Eigenfunctions')
plt.xlabel('x')
plt.ylabel('Wave function')
plt.legend()
plt.grid(True)
plt.show()
```



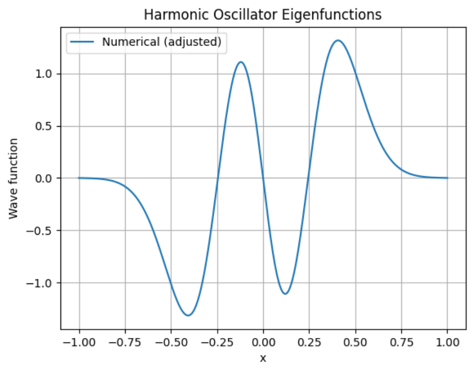
(a) Ground state.



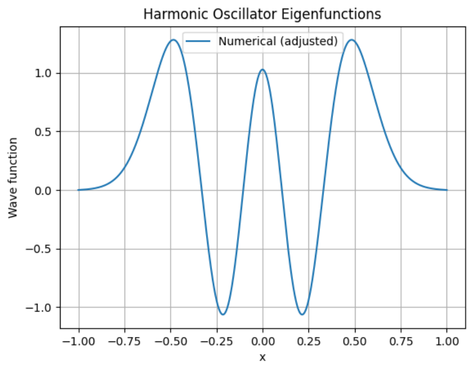
(b) First excited state.



(c) Second excited state.



(d) Third excited state.



(e) Fourth excited state.

Figure 2: Eigenfunctions for the harmonic oscillator.