Project 1

Clayton Seitz
October 29, 2022

1 PART I

1.1 A

Here, we are trying to solve for the solutions to Schrodinger's eigenvalue equation:

$$\hat{H}_0\phi_n = \epsilon_n\phi_n$$

By discretizing ϕ_n, each ϕ_n becomes a finite dimensional vector and we can write $\hat{H}$ explicitly as a matrix. That matrix satisfies

$$\sum_j \langle i|\hat{H}_0|j\rangle \vec{\phi}_{n,j} = \epsilon_n\vec{\phi}_n$$

wheree $\langle i|\hat{H}_0|j\rangle$ is the matrix element $[H_0]_{ij}$. It was shown the Schrodingers wave equation could be expressed in discrete form, as

$$-t(\phi_{n,i+1}+\phi_{n,i-1})+(2t+V_i)\phi_{n,i} = \epsilon_n\phi_{n,i}$$

which gives us a relationship between $\phi_{n,i}$ and the neighboring elements $\phi_{n,i-1}$ and $\phi_{n,i+1}$. The eigenvalues equation can then be written as a matrix multiplication

$$\hat{H}_0\phi_n = \begin{pmatrix} 2t+V_1 & -t & 0 & \dots \\ -t & 2t+V_2 & -t & \dots \\ 0 & -t & 2t+V_3 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}\begin{pmatrix}\phi_{n,1}\\ \phi_{n,2}\\ \phi_{n,3}\\ \vdots\end{pmatrix}$$

To show that the eigenvectors form an orthonormal set, We can define a matrix T such that each column of T is one eigenvector of $\hat{H}_0$. If the eigenvectors are indeed orthonormal, then

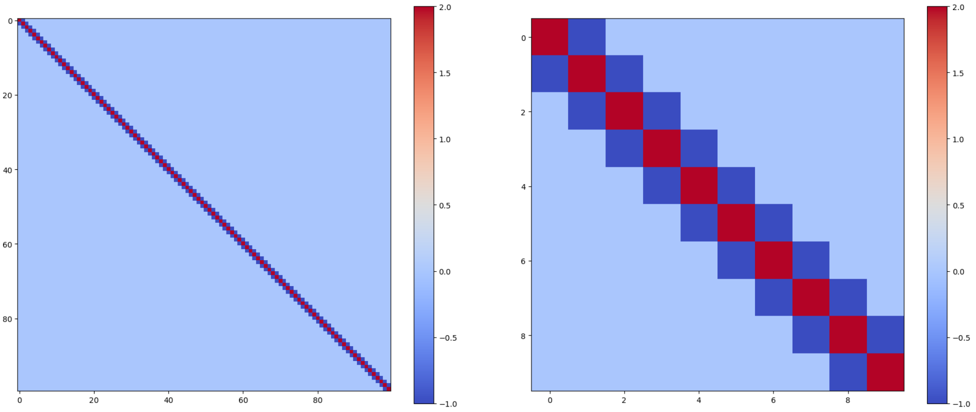

Figure 1.1:

$$T^T T = I_{N\times N}$$

These are the solutions to a free particle. The analytical eigenvalues of that problem should be

1.2 Source Code

```python
import numpy as np

def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
```

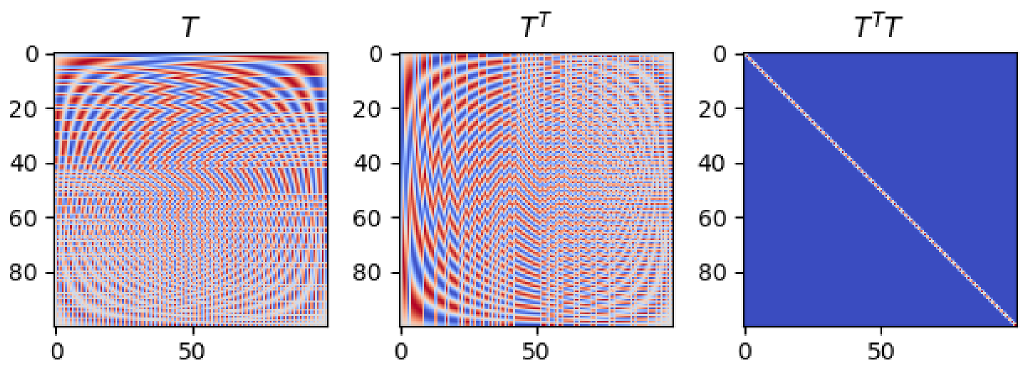

Figure 1.2:

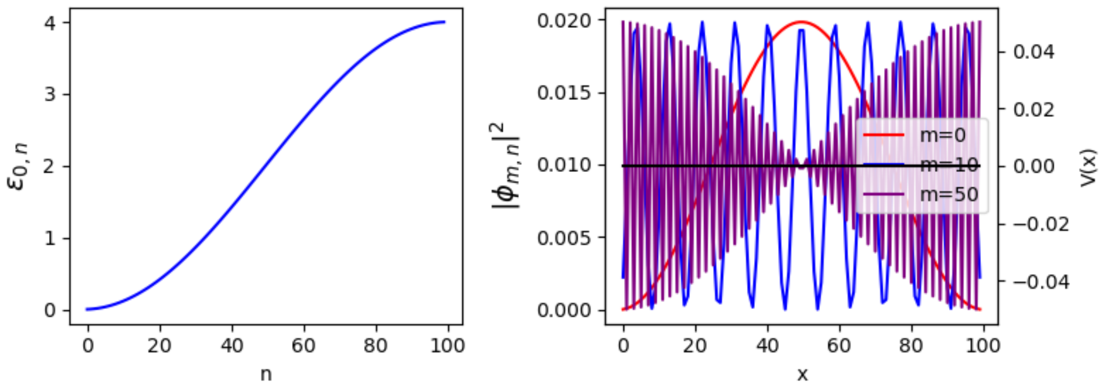

Figure 1.3:

```
    M = None #to become the incidence matrix
    VT = np.zeros((n*m,1), int)  #dummy variable

    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2),1)

    for i in range(m-1):
        for j in range(i+1, m):
            [r,c] = np.where(M2 == M1[i,j])
            for k in range(len(r)):
                VT[(i)*n + r[k]] = 1;
                VT[(i)*n + c[k]] = 1;
                VT[(j)*n + r[k]] = 1;
                VT[(j)*n + c[k]] = 1;

                if M is None:
                    M = np.copy(VT)
                else:
                    M = np.concatenate((M, VT), 1)

                VT = np.zeros((n*m,1), int)

    return M
```

2 Part II