INTRODUCTION TO COMPUTING

**EIGENSTATES OF SCHRODINGER EQUATION**

ADITYA SINGH 2K19/EP/005

horizontal line

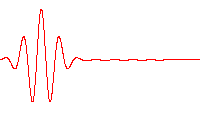
# 

# Introduction

The Schrödinger equation is a linear partial differential equation that describes the wave function or state function of a quantum-mechanical system.

In classical mechanics, Newton's second law (F = ma) is used to make a mathematical prediction as to what path a given physical system will take over time following a set of known initial conditions. Solving this equation gives the position and the momentum of the physical system as a function of the external force F on the system. Those two parameters are sufficient to describe its state at each instant. In quantum mechanics, the analogue of Newton's law is Schrödinger's equation.

# SCHR~~O~~DINGER’S EQUATION



A wave function that satisfies the nonrelativistic Schrödinger equation with V = 0. In other words, this corresponds to a particle traveling freely through empty space. The real part of the wave function is plotted.

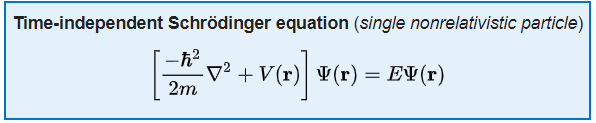
### Time Dependent Equation

### The form of the Schrödinger equation depends on the physical situation (see below for special cases). The most general form is the time-dependent Schrödinger equation (TDSE), which gives a description of a system evolving with time:

To apply the Schrödinger equation, write down the Hamiltonian for the system, accounting for the kinetic and potential energies of the particles constituting the system, then insert it into the Schrödinger equation. The resulting partial differential equation is solved for the wave function, which contains information about the system.

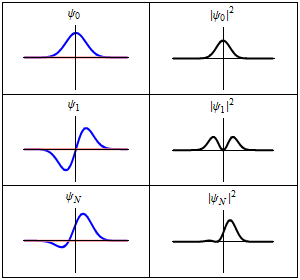
### Time Independent Equation

### The time-dependent Schrödinger equation described above predicts that wave functions can form standing waves, called stationary states.



where the Hamiltonian *function* is the sum of the kinetic and potential energies. That is,

for a single particle in the non-relativistic limit.



Each of these three rows is a wave function which satisfies the time-dependent Schrödinger equation for a harmonic oscillator. Left: The real part (blue) and imaginary part (red) of the wave function. Right: The probability distribution of finding the particle with this wave function at a given position. The top two rows are examples of stationary states, which correspond to standing waves. The bottom row is an example of a state which is not a stationary state. The right column illustrates why stationary states are called "stationary".

# PREREQUISITE MATLAB FUNCTIONS

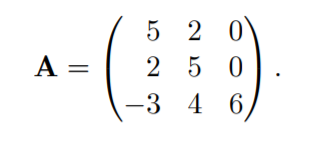
### EIG(A)

* returns a column vector containing the eigenvalues of square matrix A.

[V,D] = eig(A) - returns diagonal matrix D of eigenvalues and matrix V whose columns are the corresponding right eigenvectors, so that A\*V = V\*D.

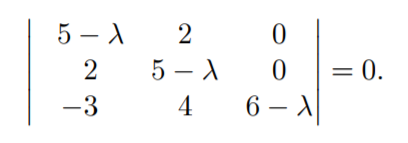
eigs - It finds largest eigenvalues and eigenvectors of a sparse matrix.

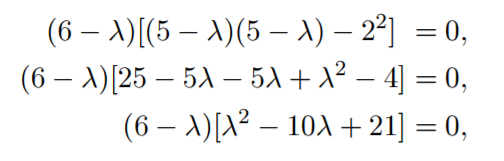
[V,D] = eigs(A) - returns a diagonal matrix D of A's six largest magnitude eigenvalues and a matrix V whose columns are the corresponding eigenvectors.

Example, Find Eigen value of

Mathematically, we would solve like:

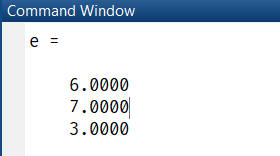
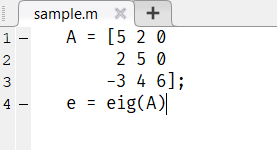








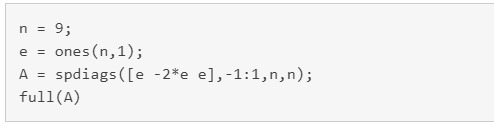
But in Matlab, it's easy as simple as:



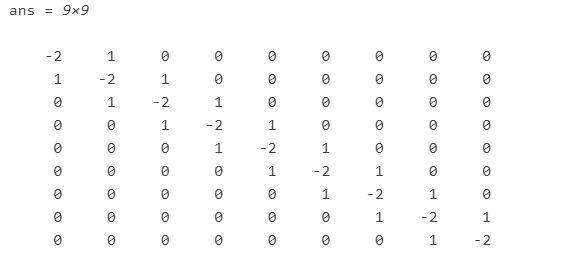
### SPDIAGS(A)

B = spdiags(A) - extracts the nonzero diagonals from m-by-n matrix A and returns them as the columns in min(m,n)-by-p matrix B, where p is the number of nonzero diagonals.

S = spdiags(A ,d ,m ,n) - creates an m-by-n sparse matrix S by taking the columns of A and placing them along the diagonals specified by d.



Create a 9-by-1 vector of ones, and then create a tridiagonal matrix using the vector.



### 

### REPMAT(A)

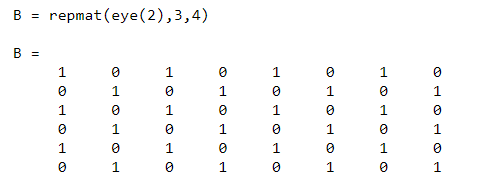
Syntax:

B = repmat(A,n) returns an array containing n copies of A in the row and column dimensions. The size of B is size(A)\*n when A is a matrix.

B = repmat(A,m,n) creates a large matrix B consisting of an m-by-n tiling of copies of A. The statement repmat(A,n) creates an n-by-n tiling.

Example:

In this, repmat replicates 12 copies of the second-order identity matrix, resulting in a "checkerboard" pattern.



The statement N = repmat(NaN,[2 3]) creates a 2-by-3 matrix of NaNs.

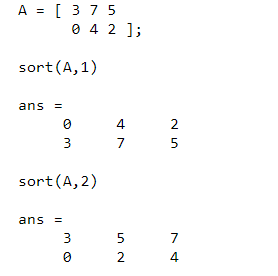
### SORT(A)

* sort elements in ascending order

B = sort(A) sorts the elements along different dimensions of an array, and arranges those elements in ascending order.

|  |  |
| --- | --- |
| **If A is a ...** | **sort(A) ...** |
| Vector | Sorts the elements of A in ascending order. |
| Matrix | Sorts each column of A in ascending order. |
| Multidimensional array | Sorts A along the first non-singleton dimension, and returns an array of sorted vectors. |
| Cell array of strings | Sorts the strings in ASCII dictionary order. |

Example:

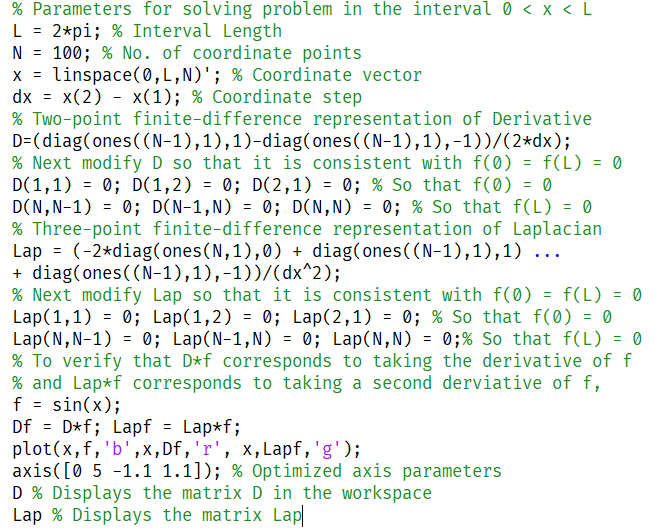


### Write differential operator as a matrix

First we will see how to Calculate first and second derivative numerically, showing how to write the differential operator as a matrix.

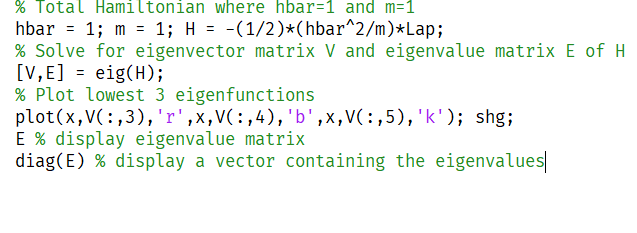
The command ones(a, b) generates an a × b matrix of ones. The command diag(A, n) generates a matrix with the elements of the vector A placed along the n th diagonal and zeros everywhere else. The central diagonal corresponds to n = 0, the diagonal above the center one corresponds to n = 1, etc...).

Matlab Code is as follows:

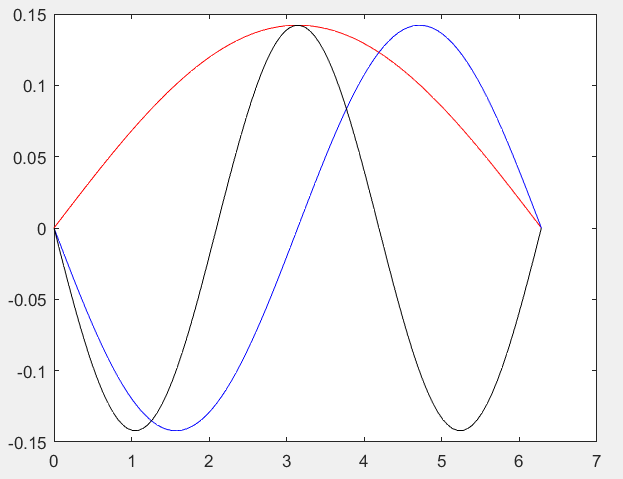


### Matrix representation of differential operators

Below is the program for solving for Eigenvectors & Eigenvalues of Infinite Square Well.

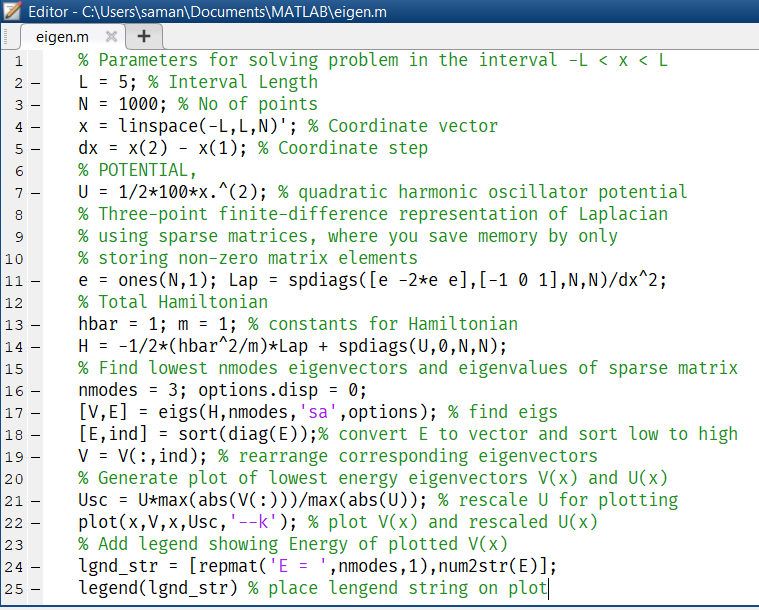


In MATLAB, the command [V, E] = eig(H) does precisely this: it generates two matrices. The first matrix V has as its columns the eigenvectors |ΨE(0). The second matrix E is a diagonal matrix with the eigenvalues Ei corresponding to the eigenvectors |ΨEi (0) placed along the central diagonal. We can use the command E = diag(E) to convert this matrix into a column vector. For brevity, we omit the commands setting the parameters L, N, x, and dx.



# EIGEN ENERGY OF SCHRODINGER EQUATION

This Matlab code solves the time independent Schrodinger equation for finite square well potentials, the harmonic oscillator potential and even for potentials that can only be solved numerically which is U(x) = ½100\*(x2) . In order to minimize the amount of RAM required, the codes shown make use of sparse matrices, where only the non-zero elements of the matrices are stored. The commands for sparse matrices are very similar to those for non-sparse matrices. For example, the command [V, E] = eigs(H, nmodes..) provides the nmodes lowest energy eigenvectors V of the sparse matrix H.



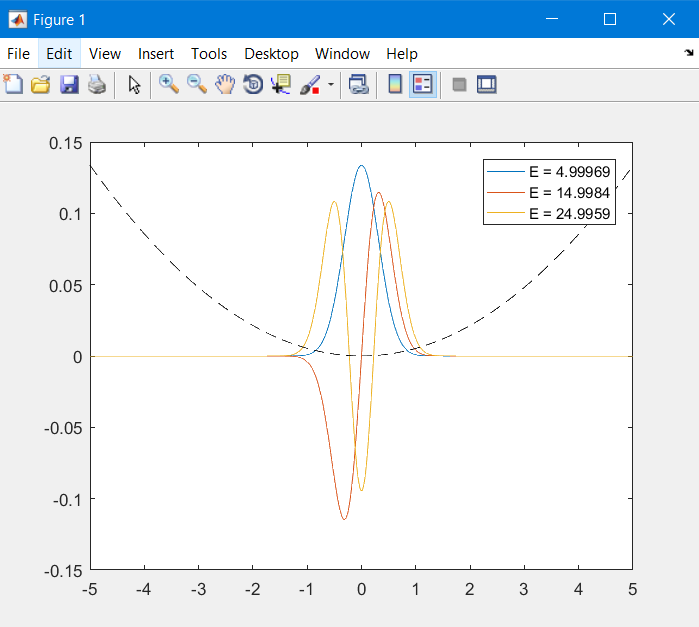


Fig. shows the plot obtained from Program for the potential U = 1/2 · 100 ·x2. Note that the 3 lowest energies displayed in the figure are just as expected due to the analytic formula E=hw(n+½) with n = integer and ω = sqrt(k/m) = 10 rad/s.

The corresponding energies displayed within the figure legend, 4.99969, 14.9984 and 24.9959, are, within rounding error, precisely those expected from the Energy equation for the three lowest-energy modes.

**THANK YOU**