Computational Physics (PHYS6350)

Lecture 21: Problems in quantum mechanics

- Matrix method for eigenenergies and eigenstates
- Time-dependence

April 11, 2023

Instructor: Volodymyr Vovchenko (vvovchenko@uh.edu)

Course materials: https://github.com/vlvovch/PHYS6350-ComputationalPhysics

Finding the eigenenergies

Time-independent Schroedinger equation reads

$$\left[-\frac{\hbar^2}{2m}\frac{d^2}{dx^2} + V(x)\right]\psi(x) = E\psi(x).$$

e.g. in a box of length L with boundary conditions $\psi(-L/2) = \psi(L/2) = 0.$

In the case of (an)harmonic oscillator we learned how to use the shooting method to find the eigenenergies by combining a root finder (bisection or secant method) with an ODE integrator (such as RK4).

This involved discretizing the space on a grid.

The problem can also be tackled efficiently by linear algebra methods.

Matrix method for eigenenergies and eigenstates

By discretizing the space into N intervals, we can represent the wave function $\psi(x)$ as an $(N+1)$-dimensional vector $\psi = (\psi_0, \dots, \psi_N)$ such that

$$\psi_k = \psi(x_k), \qquad x_k = -L/2 + kdx, \qquad dx = L/N.$$

Due to boundary conditions we have $\psi_0 = \psi_N = 0$, thus effectively we deal with $(N-1)$-dimensional space.

Each operator becomes a $(N-1)$x$(N-1)$ matrix. By discretizing $\frac{d^2}{dx^2}$ by the central difference we get

$$\frac{d^2}{dx^2}\psi_n \approx \frac{\psi_{n+1} - 2\psi_n + \psi_{n-1}}{dx^2}.$$

There, the Hamiltonian, has the following matrix representation

$$H_{nm} = -\frac{\hbar^2}{2m}\left[\delta_{m,n+1}\psi_{n+1} - 2\delta_{m,n}\psi_n + \delta_{m,n-1}\psi_{n-1}\right] + \delta_{m,n}V(x_n),$$

i.e. H is a tridiagonal symmetric matrix.

Therefore finding the energies and wave function of the system corresponds to the matrix eigenvalue problem for the matrix H.

Let us apply the method to (an)harmonic oscillator we had before.

Matrix method

```
# Constants
me = 9.1094e-31    # Mass of electron
hbar = 1.0546e-34  # Planck's constant over 2*pi
e = 1.6022e-19     # Electron charge
V0 = 50*e
a  = 1e-11
N = 1000
L = 20*a
dx = L/N

# Potential functions
def Vharm(x):
    return V0 * x**2 / a**2

def Vanharm(x):
    return V0 * x**4 / a**4

# Construct the Hamiltonian matrix
def HamiltonianMatrix(V):
    H = (-hbar**2 / (2*me*dx**2)) * (np.diag((N-2)*[1],-1) + np.diag((N-1)*[-2],0) + np.diag((N-2)*[1],1))
    H += np.diag([V(-0.5 * L + dx*(k+0.5)) for k in range(1,N)],0)
    return H
```

Matrix method for harmonic oscillator

We can use QR decomposition to solve the eigenvalue problem

Since our matrix is real symmetric, we can use a straightforward implementation of the QR algorithm. We thus expect to obtain representation of H in form

$$H = Q^T A Q$$

where A is diagonal and contains the energies, while Q is orthogonal and has eigenvectors (wave functions) in its columns.

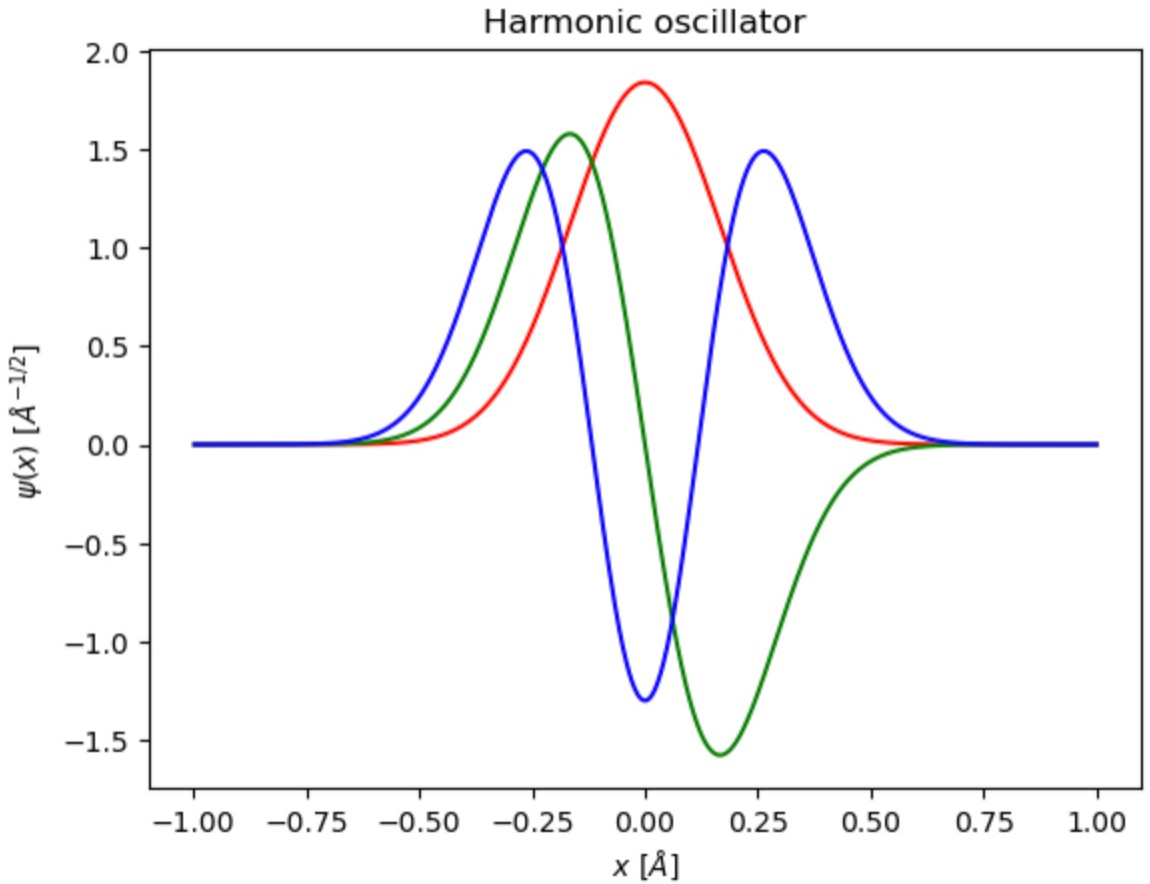
```
def eigen_qr_simple(A, iterations=100):
    Ak = np.copy(A)
    n = len(A[0])
    QQ = np.eye(n)
    for k in range(iterations):
        Q, R = np.linalg.qr(Ak)
        Ak = np.dot(R,Q)
        QQ = np.dot(QQ,Q)
    return Ak, QQ
```

```
# Harmonic oscillator
Vpot = Vharm
Vlabel = "Harmonic oscillator"
A, Q = eigen_qr_simple(HamiltonianMatrix(Vpot),50)
indices = np.argsort(np.diag(A))
eigenvalues = np.diag(A)[indices]
eigenvectors = [Q[:,indices[i]] for i in range(len(indices))]
Nprint = 10
print("First",Nprint,"eigenenergies of", Vlabel, "are")
for n in range(Nprint):
    print("E_",n,"=",eigenvalues[n]/e,"eV")
```

```
First 10 eigenenergies of Harmonic oscillator are
E_ 0 = 138.0227220181584 eV
E_ 1 = 414.0656659872072 eV
E_ 2 = 690.1036097526343 eV
E_ 3 = 966.136554028328 eV
E_ 4 = 1242.1646545118429 eV
E_ 5 = 1518.1925691076508 eV
E_ 6 = 1794.264990851771 eV
E_ 7 = 2070.579770340522 eV
E_ 8 = 2347.6427052950403 eV
E_ 9 = 2626.3110902945514 eV
```

Matrix method for harmonic oscillator

Eigenstates are encoded in the columns of matrix Q

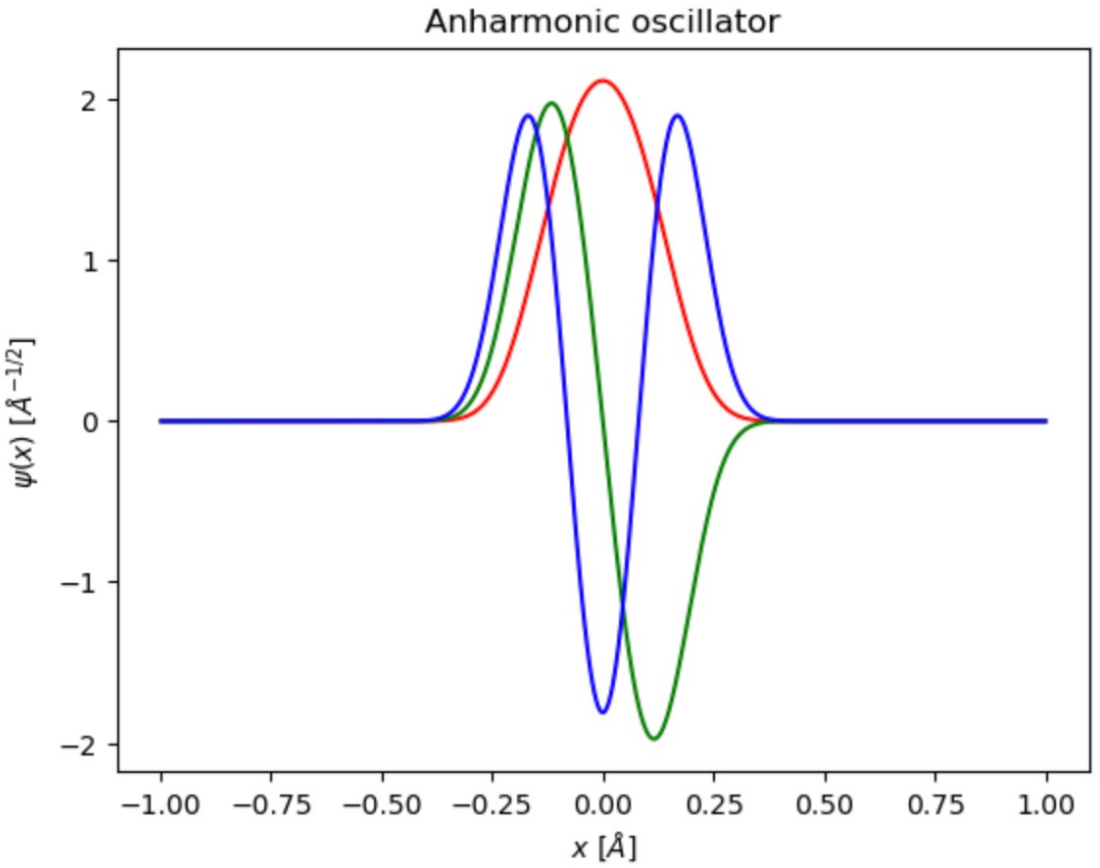


Normalization

```python
# Compute the normalisation factor with trapezoidal rule
def integral_psi2(psi, dx):
    N = len(psi) - 1
    ret = 0

    for k in range(N):
        ret += psi[k] * np.conj(psi[k]) + psi[k+1] * np.conj(psi[k+1])

    ret *= 0.5 * dx

    return ret
```

Matrix method for anharmonic oscillator

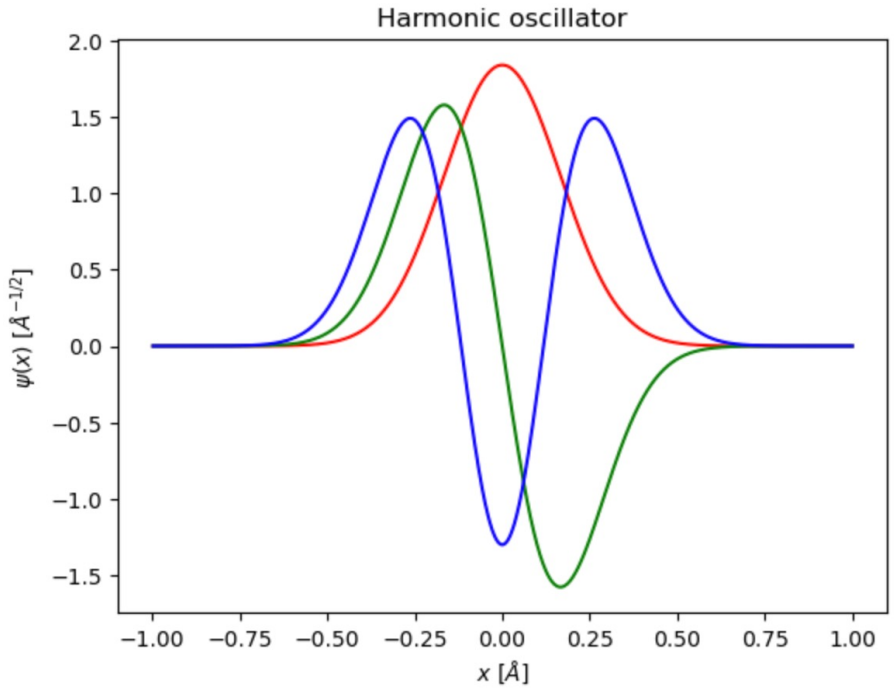
```
First 10 eigenenergies of Anharmonic oscillator are
E_ 0 = 205.3022520064578 eV
E_ 1 = 735.6578481328587 eV
E_ 2 = 1443.4564627068437 eV
E_ 3 = 2254.386094620506 eV
E_ 4 = 3148.0999787331502 eV
E_ 5 = 4111.305594420894 eV
E_ 6 = 5135.101294752596 eV
E_ 7 = 6213.029170837947 eV
E_ 8 = 7340.383787818462 eV
E_ 9 = 8514.323805747972 eV
```

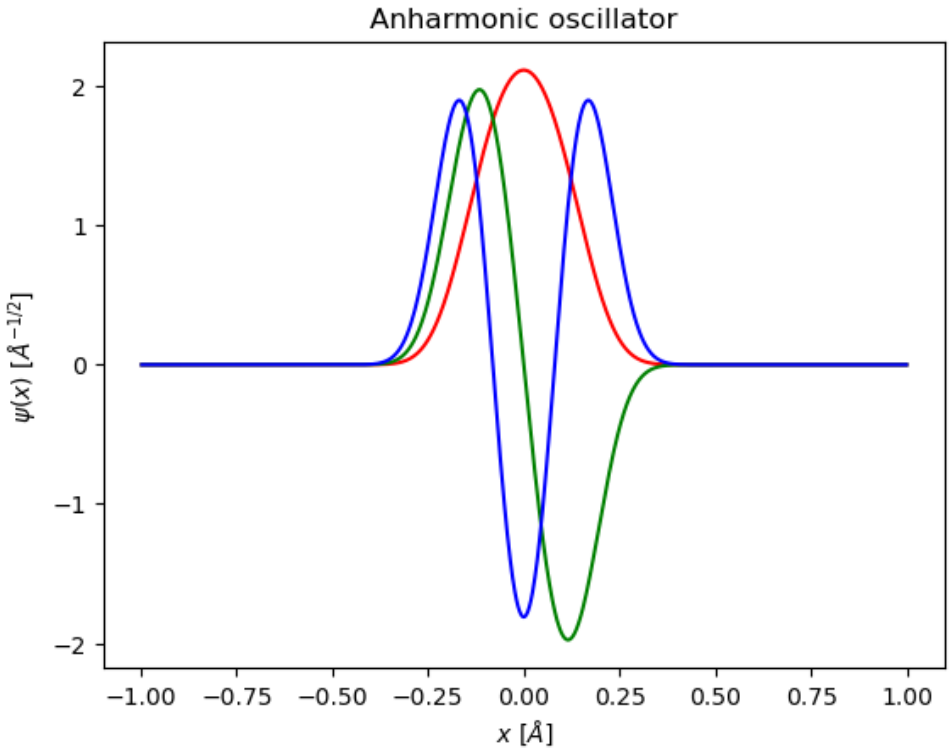


Matrix method

One can also use efficient implementations of the eigenvalue problem in numpy

```
Vpot = Vharm
Vlabel = "Harmonic oscillator"
eigenvalues, eigenvectors = np.linalg.eigh(HamiltonianMatrix(Vpot))
Nprint = 10
print("First",Nprint,"eigenenergies of", Vlabel, "are")
for n in range(Nprint):
    print("E_",n,"=",eigenvalues[n]/e,"eV")
```





Time-dependent Schroedinger equation

The time-dependent Schroedinger equation reads

$$\hat{H}\psi = i\hbar\frac{\partial\psi}{\partial t} \ .$$

e.g. for a free particle it reads

$$-\frac{\hbar^2}{2m}\frac{\partial^2\psi}{\partial x^2} = i\hbar\frac{\partial\psi}{\partial t} \ .$$

Formal solution be written as

$$\psi(t) = e^{-\frac{it}{\hbar}\hat{H}}\psi(0),$$

with $\hat{U}(t) = e^{-\frac{it}{\hbar}\hat{H}}$ being the time evolution operator. It is a unitary operator $U^\dagger U = \hat{I}$, therefore, the norm of the wave function $|\psi|^2$ is conserved.

Time integration

We can study time evolution by succesively applying approximate $\hat{U}(\Delta t)$ over small time intervals.

- FTCS scheme

  $$\hat{U}(\Delta t) = e^{-\frac{i\Delta t}{\hbar}\hat{H}} \approx 1 - \frac{i\Delta t}{\hbar}\hat{H}$$

  Such an operator is not unitary since

  $$U^{\dagger}(\Delta t) = 1 + \frac{i\Delta t}{\hbar}\hat{H} \neq \hat{U}(\Delta t).$$

  If U is applied to an energy eigenstate ψ_l, such that $\hat{H}\psi_l = E_k\psi_l$, we get

  $$\psi_l(N\Delta t) = \lambda_l^N \psi_l(0),$$

  where $\lambda_l = \left[1 - \frac{i\Delta t E_l}{\hbar}\right]$. Since $|\lambda_l| > 1$, the method is unstable.
- Implicit scheme

We apply the approximation of the FTCS scheme to the inverse of $\hat{U}(\Delta t)$. This implies

$$\hat{U}(\Delta t) = e^{-\frac{i\Delta t}{\hbar}\hat{H}} \approx \frac{1}{1 + \frac{i\Delta t}{\hbar}\hat{H}}.$$

The operator is also non-unitary. The method is stable because

$$|\lambda_l| = \left|\frac{1}{1 + \frac{i\Delta t}{\hbar}E_L}\right| < 1,$$

but the method does not conserve the norm of the wave function.

Time integration: Crank-Nicholson scheme

explicit

$$\hat{U}(\Delta t) = e^{-\frac{i\Delta t}{\hbar}\hat{H}} \approx 1 - \frac{i\Delta t}{\hbar}\hat{H}$$

implicit

$$\hat{U}(\Delta t) = e^{-\frac{i\Delta t}{\hbar}\hat{H}} \approx \frac{1}{1 + \frac{i\Delta t}{\hbar}\hat{H}}.$$

- Crank-Nicholson scheme

Crank-Nicholson scheme takes the combitation of FTCS and implicit schemes. This correspons to a rational approximation of $\hat{U}(\Delta t)$:

$$\hat{U}(\Delta t) = e^{-\frac{i\Delta t}{\hbar}\hat{H}} \approx \frac{1 - \frac{i\Delta t}{\hbar}\hat{H}}{1 + \frac{i\Delta t}{\hbar}\hat{H}}.$$

This operator is unitary (for an hermitian $\hat{H}$), $U^\dagger U = UU^\dagger$, and conserves the norm of the wave function.

Free particle in a box

Let us consider the particle in a box of length L. We thus have boundary conditions $\psi(0) = \psi(L) = 0$.

Given some initial wave function $\psi(x)$, we can numerically integrate the Schroedinger equation to study the time evolution of the wave function.

Let us write the equation in a form:

$$\frac{\partial \psi}{\partial t} = \frac{i\hbar}{2m}\frac{\partial^2 \psi}{\partial x^2}.$$

We can now use the central difference approximation for $\partial^2\psi/\partial x^2$:

$$\frac{\partial \psi}{\partial t} \approx \frac{i\hbar}{2ma^2}\left[\psi(x+a,t) - 2\psi(x,t) + \psi(x-a,t)\right].$$

This gives us discretized wave function in coordinate space, in full analogy to the heat equation that we studied before. The only difference is that now we are dealing with complex-valued functions.

- FTCS scheme

$$\psi_k^{n+1} = \psi_k^n + h\frac{i\hbar}{2ma^2}(\psi_{k+1}^n - 2\psi_k^n + \psi_{k-1}^n).$$

- Implicit scheme

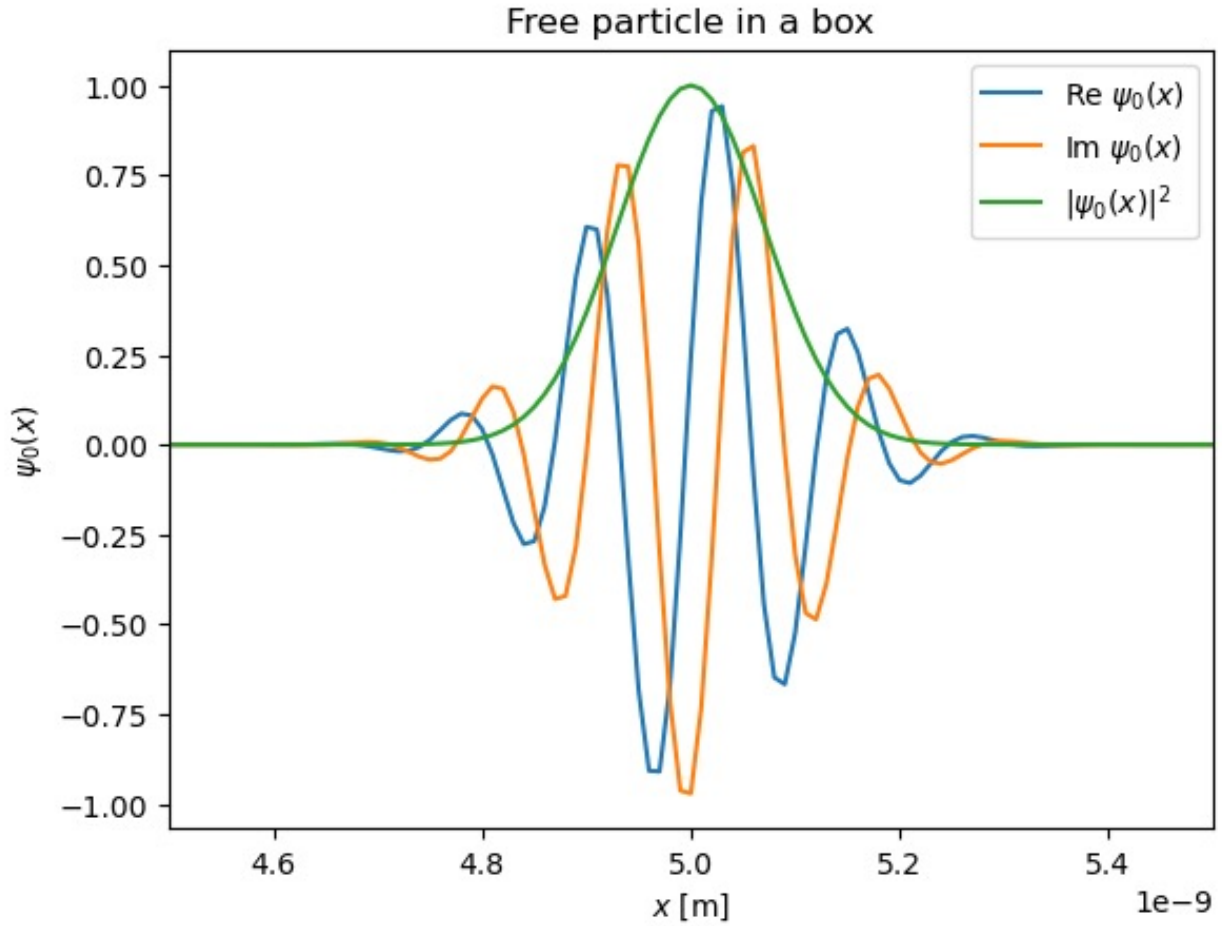
$$\psi_k^{n+1} = \psi_k^n + h\frac{i\hbar}{2ma^2}(\psi_{k+1}^{n+1} - 2\psi_k^{n+1} + \psi_{k-1}^{n+1}).$$

- Crank-Nicholson scheme

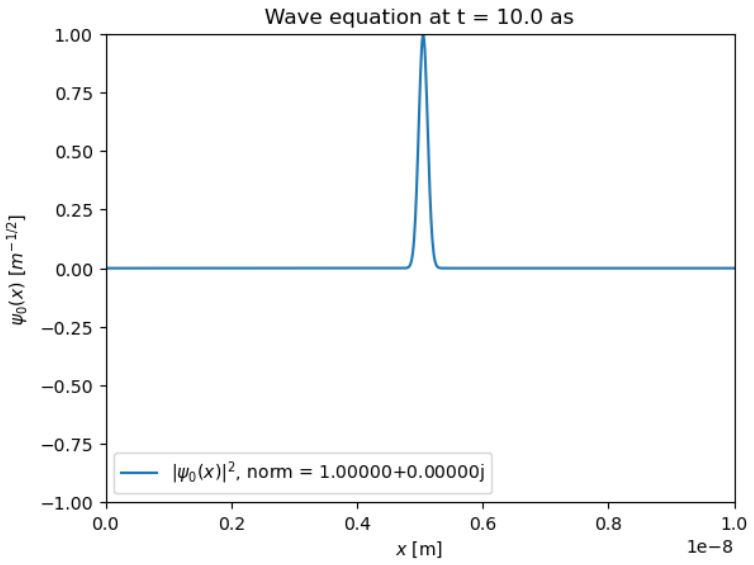
$$\psi_k^{n+1} = \psi_k^n + \frac{h}{2}\frac{i\hbar}{2ma^2}(\psi_{k+1}^n - 2\psi_k^n + \psi_{k-1}^n) + \frac{h}{2}\frac{i\hbar}{2ma^2}(\psi_{k+1}^{n+1} - 2\psi_k^{n+1} + \psi_{k-1}^{n+1}).$$

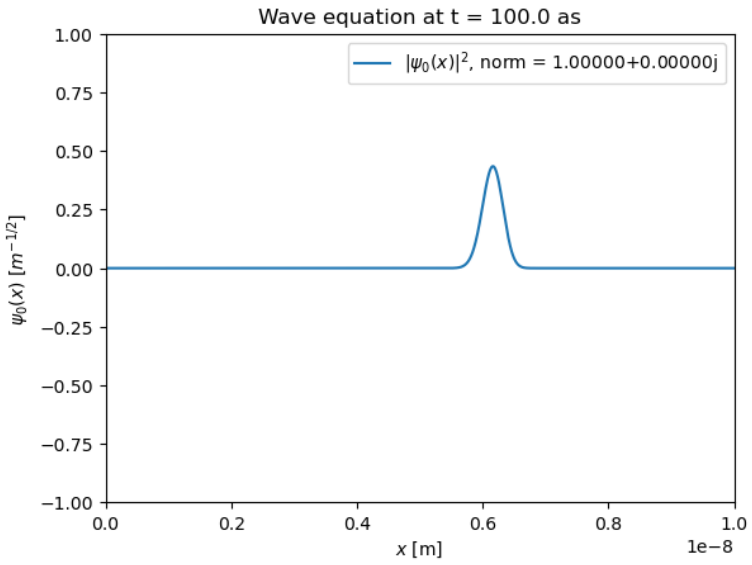
Initial wave function: Gaussian wave packet

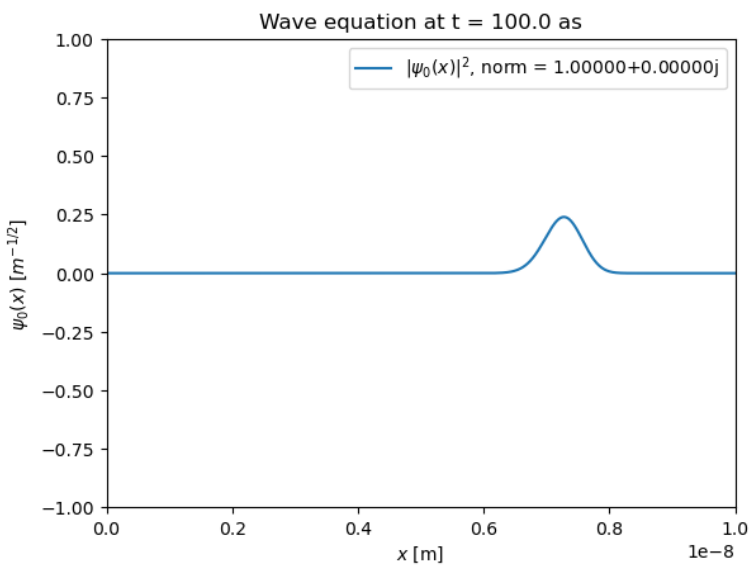
```
def psi0(x):
    return np.exp(-(x-x0)**2/(2*sig**2))*np.exp(1j*kappa*x)
```



Time evolution







Expanding wave packet (towards plane wave), norm is conserved