# TDSE-v1

March 12, 2024

## 0.1 Time-dependent Schroedinger Equation

We consider the problem of finding the evolution of the (complex) wave function $\phi$ in

$$i\hbar\frac{\partial\phi}{\partial t} = -\frac{\hbar^2}{2m}\nabla^2\phi + V\phi = H\phi$$

given its value at some initial time.

Then, for a one-dimensional problem, we have,

$$\frac{\partial\phi}{\partial t} = -i\left(-\frac{\hbar}{2m}\frac{\partial^2\phi}{\partial x^2} + U(x)\right)\phi = -\frac{i}{\hbar}H\phi$$

where,

$$U(x) = V(x)/\hbar$$

For simplicity, let us work with a unit system to have $\hbar = m = 1$. Then we have,

$$\frac{\partial\phi}{\partial t} = -i\left(-\frac{\partial^2\phi}{\partial x^2} + U(x)\right)\phi = -iH(x)\phi$$

A direct implementation using An Euler-like scheme would give us:

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} = -iH\phi^n$$

and hence,

$$\phi^{n+1} = (1 - iH\Delta t)\phi^n$$

This is unstable as the eigenvalues of this operator

$$(1 - i\epsilon_\lambda\Delta t)$$

have modulii

$$(1 + \epsilon_\lambda^2\Delta t^2)^{1/2} > 1.$$

However a half-step approach is stable. It starts from

$$(1 + \frac{i}{2}H\Delta t)\phi^{n+1} = (1 - \frac{i}{2}H\Delta t)\phi^n$$

to yield,

$$\phi^{n+1} = \frac{(1 - \frac{i}{2}H\Delta t)}{(1 + \frac{i}{2}H\Delta t)}\phi^n$$

The above propagator is unitary.

For actual numerical computation, it is efficient to rewrite the above as

$$\phi^{n+1} = \left(\frac{2}{1 + \frac{i}{2}H\Delta t} - 1\right)\phi^n = \chi - \phi^n$$

where, we have

$$(1 + \frac{i}{2}H\Delta t)\chi = 2\phi^n$$

Explicitly the above takes the form

$$-\frac{i\Delta t}{2h^2}\chi_{j+1} + \left(1 + \frac{i\Delta t}{h^2} + \frac{i\Delta t}{2}U_j\right)\chi_j - \frac{i\Delta t}{2h^2}\chi_{j-1} = 2\phi_j^n$$

or

$$\chi_{j+1} + \left(-2 + \frac{2ih^2}{\Delta t} - h^2U_j\right)\chi_j + \chi_{j-1} = \frac{4ih^2}{\Delta t}\phi_j^n$$

or

$$A\chi = \frac{4ih^2}{\Delta t}\phi^n$$

and hence

$$\chi = A^{-1}\frac{4ih^2}{\Delta t}\phi^n$$

This yields

$$\phi^{n+1} = \left(A^{-1}\frac{4ih^2}{\Delta t} - 1\right)\phi^n$$

where,

$$A = \begin{pmatrix} -2 + (k - U_1)h^2 & 1 & 0 & & & & \\ 1 & -2 + (k - U_2)h^2 & 1 & & & & \\ 0 & 1 & -2 + (k - U_3)h^2 & \ddots & & & \\ & & \ddots & \ddots & & & \\ & & & -2 + (k - U_{n-2})h^2 & 1 & & \\ & & & 1 & -2 + (k - U_{n-1})h^2 & 1 & \\ & & & & 1 & -2 + (k - & \end{pmatrix}$$

[40]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[41]:
```
# Grid points
N = 201

# Create the grid
xs, h = np.linspace(0.0,10.0,N,retstep=True)
```

```
# Timestep
delt = 0.01

# Given potential
U = np.zeros(N)

# Initial condition
phi0 = np.sin(5*np.pi*xs/10);

# Boundary condition
p0 = 0.0
pN = 0.0

# A parameter used in expressions
k = 2.0j/delt
```

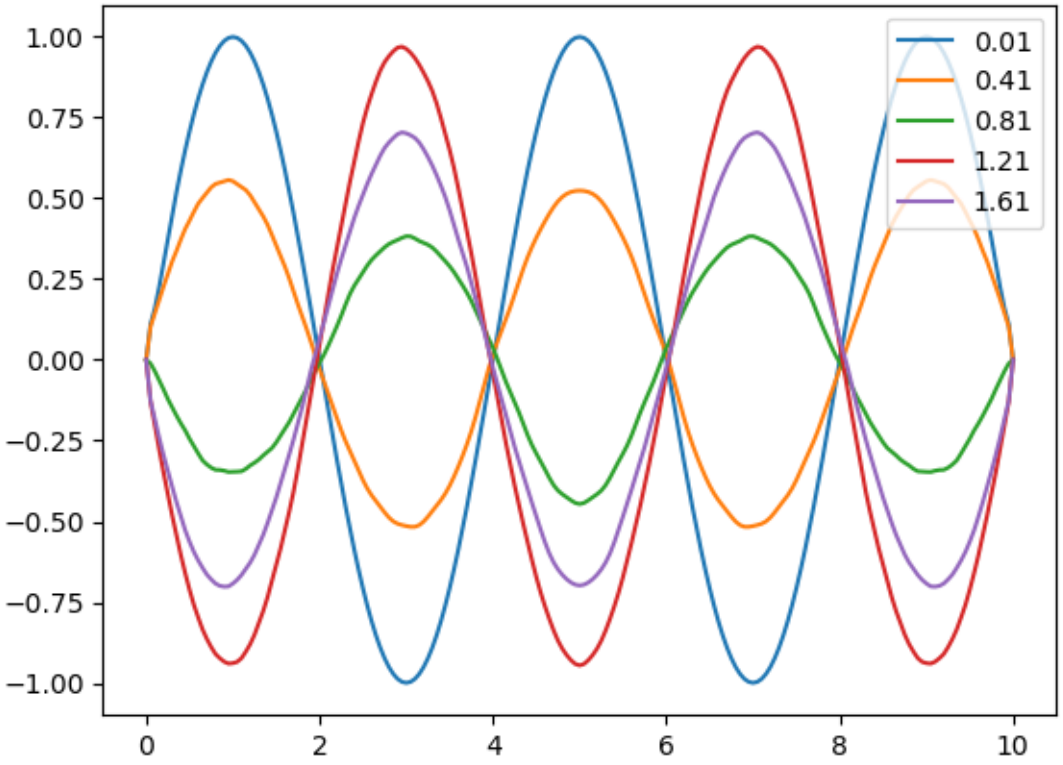
[47]:
```
# Construct A
A = (k*np.eye(N) - np.diag(U))*h**2 - 2*np.eye(N) + np.eye(N,k=1) + np.
 ↪eye(N,k=-1)

# Find inverse of A
Ainv = np.linalg.inv(A)

# Calculate the propagator
Uprop = Ainv*(4*1j*h**2)/delt - np.eye(N)
```

[49]:
```
d,v = np.linalg.eig(Uprop);
abs(d[0])
```

[49]:
```
1.0000000000000009
```

[29]:
```
# Begin from the initial condition
phi = phi0

# set initial and final times
t = 0.0
tfinal = 2.0

# k keeps track of iterations
k = 0

while t<tfinal:
    # Evolve
    phi = np.dot(Uprop,phi)
    # Set boundary
    phi[0] = p0
```

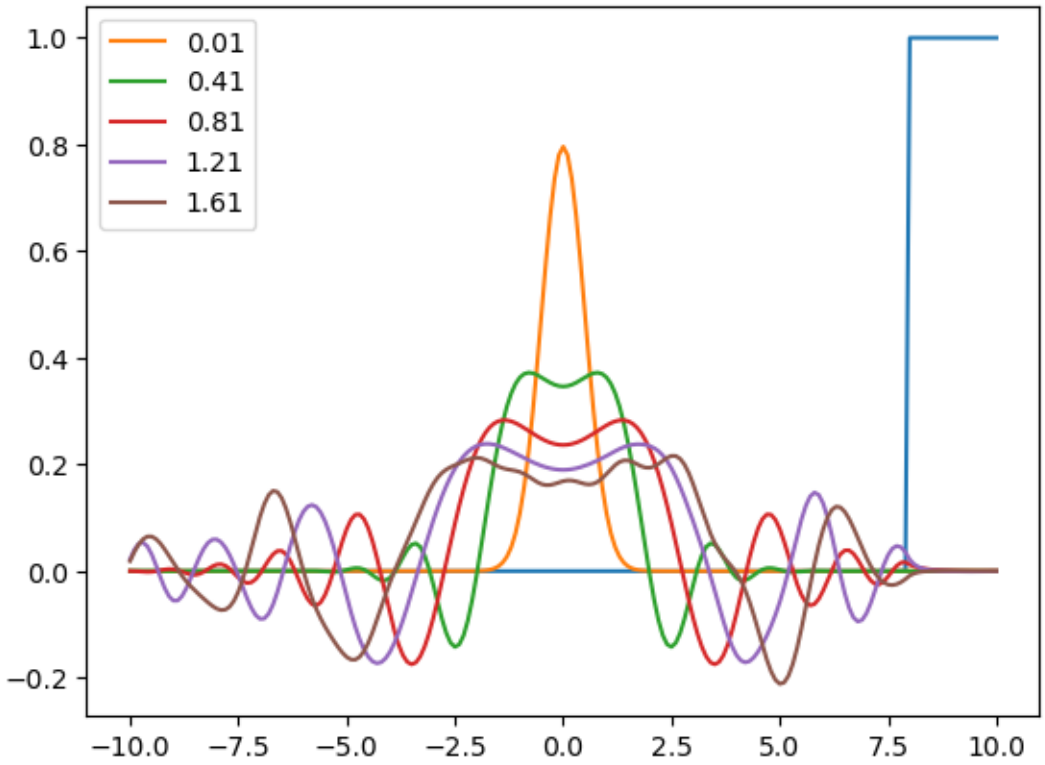
```
    phi[-1] = pN
    # Increment t
    t += delt
    # plot after every 40 iteration
    if k%40==0:
        plt.plot(xs,np.real(phi),label=str(round(t,2)))
        plt.legend()
    # increment k
    k += 1
```




[36]:
```
# Grid points
N = 201

# Create the grid
xs, h = np.linspace(-10.0,10.0,N,retstep=True)

# Timestep
delt = 0.01

# Given potential
U = np.zeros(N)
U[180:] = 40.0
```

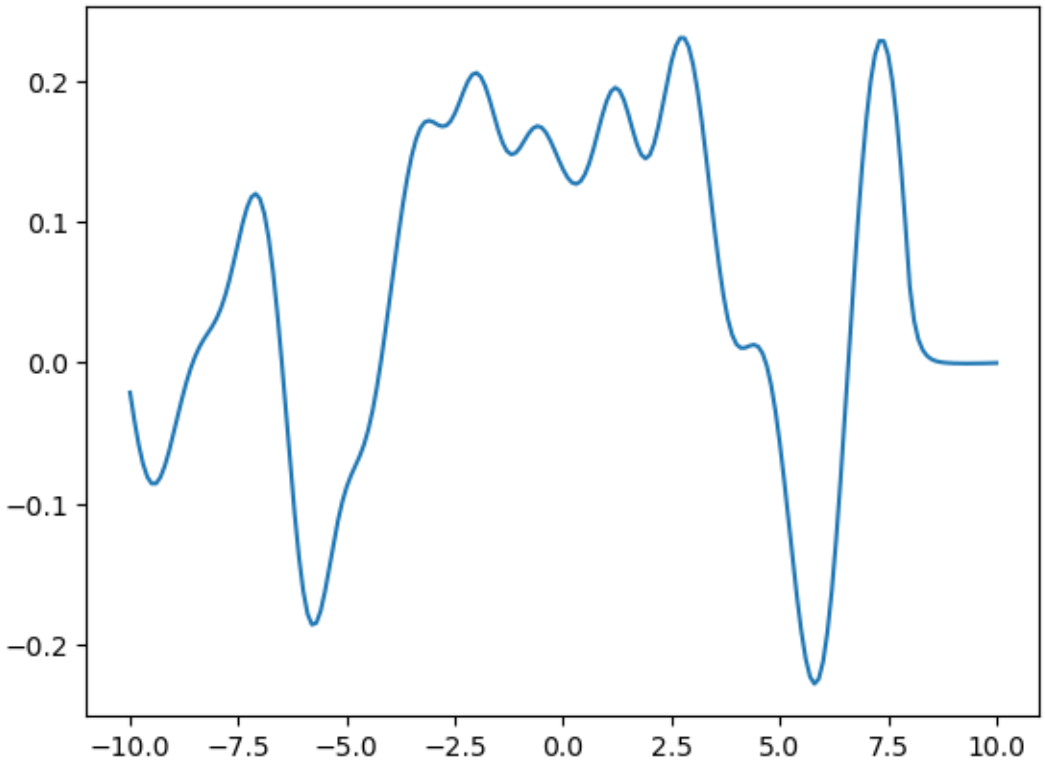
```
# Initial condition
s=0.5
phi0 = np.exp(-xs**2/2/s**2)/np.sqrt(2*np.pi)/s

# Boundary condition
#p0 = 0.0
#p1 = 0.0

# A parameter used in expressions
k = 2.0j/delt

# Construct A matrix
A = (k*np.eye(N) - np.diag(U))*h**2 - 2*np.eye(N) + np.eye(N,k=1) + np.
 ↪eye(N,k=-1)

# Calculate the inverse
Ainv = np.linalg.inv(A)

# Calculate the propagator
Uprop = Ainv*(4*1j*h**2)/delt - np.eye(N)
```

[38]:
```
# Begin from the initial condition
phi = phi0

# set initial and final times
t = 0.0
tfinal = 2.0

# k keeps track of iterations
k = 0

plt.plot(xs, U/np.max(U))
while t<tfinal:
    # Evolve
    phi = np.dot(Uprop,phi)
    # Increment t
    t += delt
    # plot after every 40 iteration
    if k%40==0:
        plt.plot(xs,np.real(phi),label=str(round(t,2)))
        plt.legend()
    # increment k
    k += 1
```




[39]:
```
plt.plot(xs, phi)
```

[39]: [<matplotlib.lines.Line2D at 0x7f696410ed50>]


0.2
0.1
0.0
−0.1
−0.2
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

[ ]: