Untitled

May 28, 2024

0.1 Split Step Method

If we look at bit closer at the two Schrödinger equations above, we recognize that there is some symmetry in the two Schrödinger equations, which we can use to calculate the time-dependence of the wave function. This type of method is called the split step method.

We may substitute in the right side of the position Schrödinger equation

$$\hat{D} = \frac{-\hbar^2}{2m}\frac{\partial^2}{\partial x^2} \tag{1}$$

and

$$\hat{N} = V(x,t) \tag{2}$$

such that

$$i\hbar\frac{\partial\Psi(x,t)}{\partial t} = [\hat{D}+\hat{N}]\Psi$$

with the solution

$$\Psi(x,t) = e^{-i(\hat{D}+\hat{N})t/\hbar}\Psi(x,0)$$

If we only make a small timestep dt, we can write the latter equation also as

$$\Psi(x,t+dt) = e^{-i\hat{D}dt/\hbar}e^{-i\hat{N}dt/\hbar}\Psi(x,t)$$

We may now turn to momentum space by taking the Fourier transform F

$$\tilde{\Psi}(k,t+dt) = F\left[e^{-i\hat{D}dt/\hbar}e^{-i\hat{N}dt/\hbar}\right]\tilde{\Psi}(k,t)$$

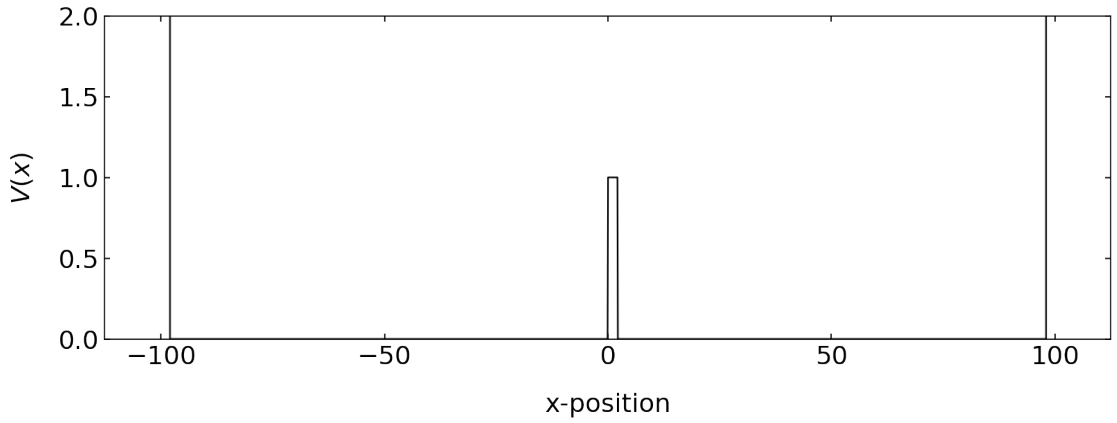
What we know now from the momentum Schrödinger equation is that the operator $\hat{D}$ will just turn into a multiplication with $\hbar k^2/2m$ in momentum space and therefore

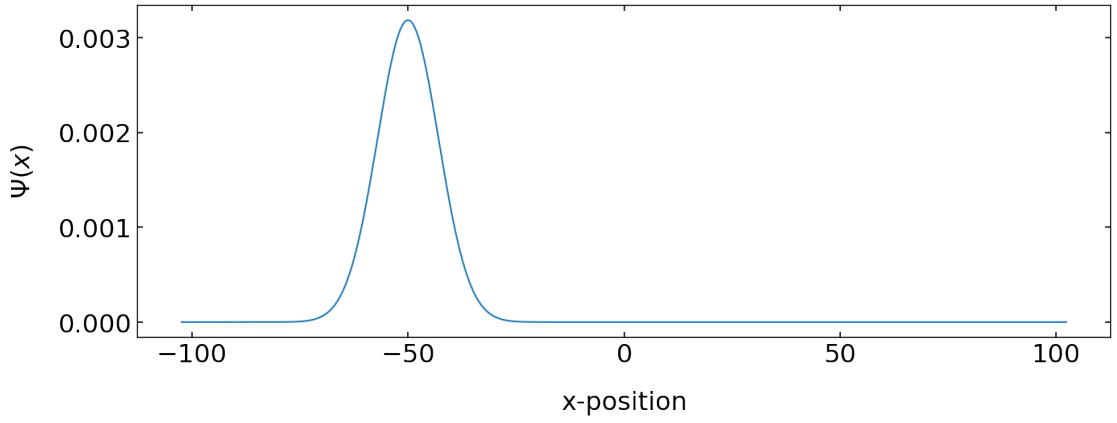
$$\tilde{\Psi}(k,t+dt) = e^{i\frac{\hbar k^2}{2m}dt}F\left[e^{-i\hat{N}dt/\hbar}\right]\tilde{\Psi}(k,t)$$

Thus if we just do the inverse Fourier transform of that, we obtain

$$\Psi(x, t+dt) = F^{-1}\left[e^{i\frac{\hbar k^2}{2m}dt} F\left[e^{-i\hat{N}dt/\hbar}\Psi(x,t)\right]\right]$$

This is the receipe we want to solve the time dependent Schrödinger equation.






```
Canvas(height=380, width=800)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-635-5827bc4795d2> in <module>
      3     ax.draw_artist(points)
      4     for j in range(100):
----> 5         tmp=ifft(phase_k*fft(psi_modx*phase_x))
      6         psi_modx=tmp
      7     points.set_data(x,1e5*np.abs(psi_modx)**2)
```

```
~/opt/anaconda3/envs/JupyterLab/lib/python3.7/site-packages/scipy/fftpack/basic
 ↪py in fft(x, n, axis, overwrite_x)
     87
     88     """
---> 89     return _pocketfft.fft(x, n, axis, None, overwrite_x)
     90
     91

~/opt/anaconda3/envs/JupyterLab/lib/python3.7/site-packages/scipy/fft/_pocketff /
 ↪basic.py in c2c(forward, x, n, axis, norm, overwrite_x, workers)
     28     out = (tmp if overwrite_x and tmp.dtype.kind == 'c' else None)
     29
---> 30     return pfft.c2c(tmp, (axis,), forward, norm, out, workers)
     31
     32

KeyboardInterrupt:
```

0.2 Old Split Step Method

If we look at bit closer at the two Schrödinger equations above, we recognize that there is some symmetry in the two Schrödinger equations, which we can use to calculate the time-dependence of the wave function. This type of method is called the split step method.

We may substitute in the right side of the position Schrödinger equation

$$\hat{D} = \frac{-\hbar^2}{2m}\frac{\partial^2}{\partial x^2} \tag{3}$$

and

$$\hat{N} = V(x,t) \tag{4}$$

such that

$$i\hbar\frac{\partial\Psi(x,t)}{\partial t} = [\hat{D}+\hat{N}]\Psi$$

with the solution

$$\Psi(x,t) = e^{-i(\hat{D}+\hat{N})t/\hbar}\Psi(x,0)$$

If we only make a small timestep dt, we can write the latter equation also as

$$\Psi(x,t+dt) = e^{-i\hat{D}dt/\hbar}e^{-i\hat{N}dt/\hbar}\Psi(x,t)$$

We may now turn to momentum space by taking the Fourier transform F

$$\tilde{\Psi}(k, t+dt) = F\left[e^{-i\hat{D}dt/\hbar}e^{-i\hat{N}dt/\hbar}\right]\tilde{\Psi}(k,t)$$

What we know now from the momentum Schrödinger equation is that the operator $\hat{D}$ will just turn into a multiplication with k^2 in momentum space and therefore

$$\tilde{\Psi}(k, t+dt) = e^{ik^2dt}F\left[e^{-i\hat{N}dt/\hbar}\right]\tilde{\Psi}(k,t)$$

Thus if we just do the inverse Fourier transform of that, we obtain

$$\Psi(x, t+dt) = F^{-1}\left[e^{ik^2dt}F\left[e^{-i\hat{N}dt/\hbar}\Psi(x,t)\right]\right]$$

This is the receipe we want to solve the time dependent Schrödinger equation.

0.2.1 Setup the simulation

While we did use in the previous calculations a sum over many plane waves to specify a Gaussian wave packet, we use the analytical version

$$\Psi(x) = \frac{1}{\sqrt{\pi}\sigma}e^{-\frac{(x-x_0)^2}{2\sigma^2}+ik_0x}$$

and the corresponding momentum space wavefunction

$$\tilde{\Psi}(k) = \frac{1}{\sqrt{\pi}\sigma}e^{-\frac{(k-k_0)^2}{2\sigma^2}-i(k-k_0)x}$$

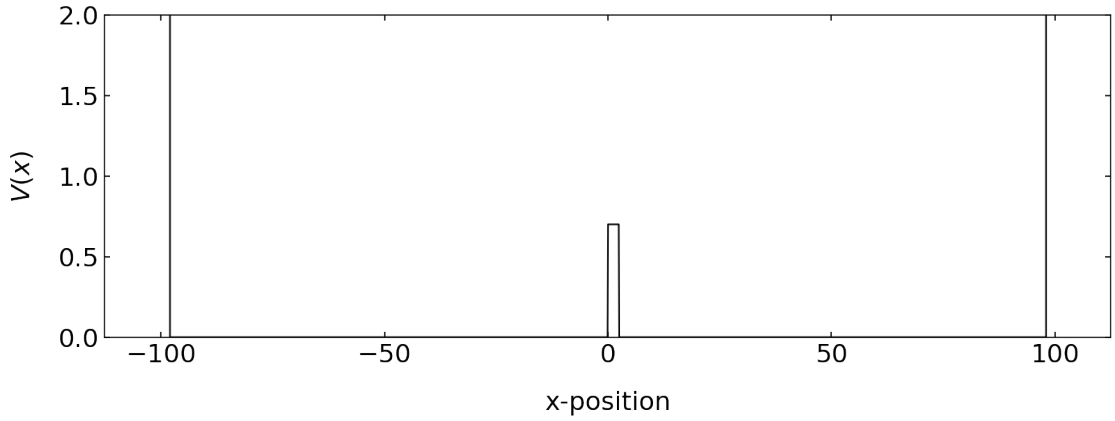
for the Gaussian wavepacket.

We also need to setup some fundamental parameters.

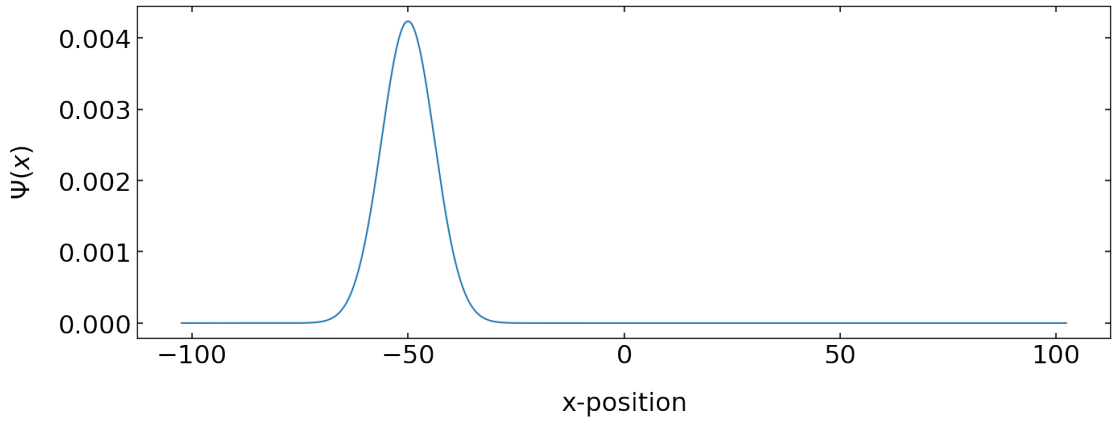
We define our spatial domain with an even number of data points, which is important for the `FFT` method.

We will also define the temporal domain to be covered by our simulations

Our potential energy landscape shall consists of very high boundaries at the edges and a barrier in the center.



Finally, we want to specify our initial wavepacket




```
Canvas(height=380, width=800)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-473-aa3a772b7cb8> in <module>
      2     fig.canvas.restore_region(background)
      3     ax.draw_artist(points)
----> 4     S.time_step(dt, 200)
      5     points.set_data(S.x,8*np.abs(S.psi_x))
      6

<ipython-input-469-1cbcc64ef919> in time_step(self, dt, Nsteps)
     82             self.psi_mod_k *= self.k_evolve
     83             self.compute_x_from_k()
---> 84             self.psi_mod_x *= self.x_evolve
     85
```

```
     86         self.compute_k_from_x()

KeyboardInterrupt:
```