Physics 256

Lecture 15 - The Diffusion Equation

Last Time

- Random walks

Today

- The diffusion equation

In [1]:
```
import numpy as np
import matplotlib as mpl
from matplotlib import pyplot as plt
```

One-dimensional diffusion equation

The one-dimensional diffusion equation can be written as

$$\frac{\partial \rho(x,t)}{\partial t} = D\frac{\partial^2 \rho(x,t)}{\partial x^2},$$

where $\rho(x,t)$ is the local concentration of molecules/particles at a point in time t and position x, and D is the diffusion coefficient of the molecule of interest. We can solve this differential equation by discretizing our system, $rho(x,t) = \rho(i\Delta x, n\Delta t) = \rho(i,n)$, and using finite difference methods. First, we expand $\rho(x,t)$ to a time $t + \Delta t$ keeping only up to first order terms

$$\rho(x, t+\Delta t) \approx \rho(x,t) + \frac{\partial \rho(x,t)}{\partial t}\Delta t$$

to obtain an epxression for the left hand side of the ODE

$$\frac{\partial \rho(x,t)}{\partial t} = \frac{\rho(x,t+\Delta t) - \rho(x,t)}{\Delta t}.$$

We can then expand $\rho(x,t)$ to positions $x \pm \Delta x$ keeping up to second order terms

$$\rho(x+\Delta x, t) \approx \rho(x,t) + \frac{\partial \rho(x,t)}{\partial x}\Delta x + \frac{1}{2}\frac{\partial^2 \rho(x,t)}{\partial x^2}(\Delta x)^2$$
$$\rho(x-\Delta x, t) \approx \rho(x,t) - \frac{\partial \rho(x,t)}{\partial x}\Delta x + \frac{1}{2}\frac{\partial^2 \rho(x,t)}{\partial x^2}(\Delta x)^2$$

and combine them

$$\rho(x+\Delta x,t)+\rho(x-\Delta x,t)=2\rho(x,t)+\frac{\partial^2\rho(x,t)}{\partial x^2}(\Delta x)^2$$

to obtain an epxression for the right hand side of the ODE

$$D\frac{\partial^2\rho(x,t)}{\partial x^2}=D\frac{\rho(x+\Delta x,t)+\rho(x-\Delta x,t)-2\rho(x,t)}{(\Delta x)^2}.$$

Putting all this together in index form we get

$$\frac{\rho(i,n+1)-\rho(i,n)}{\Delta t}=D\frac{\rho(i+1,n)+\rho(i-1,n)-2\rho(i,n)}{(\Delta x)^2}$$

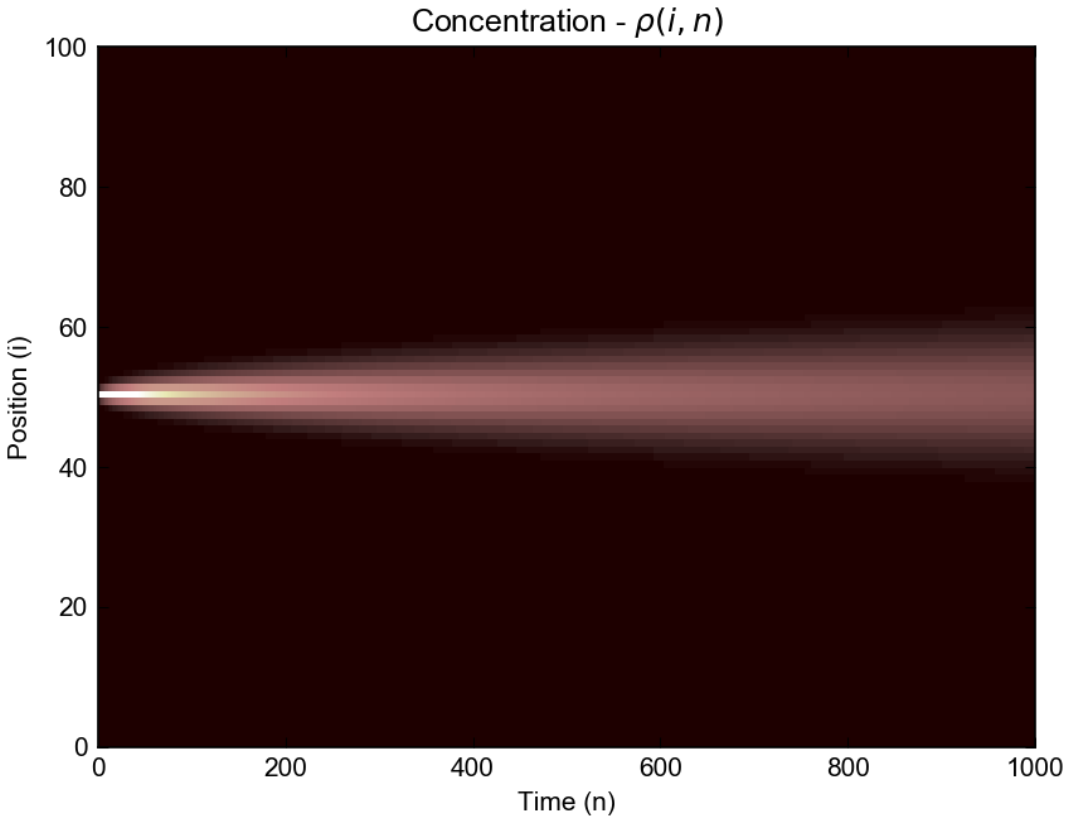
$$\rho(i,n+1)=\rho(i,n)+\frac{D\Delta t}{(\Delta x)^2}\left[\rho(i+1,n)+\rho(i-1,n)-2\rho(i,n)\right].$$

Programming challenge

Write a code that solves the 1-dimensional diffusion equation using the finite difference equation shown above for an initial set of conditions. We need to discretize both time and space, so it will be helpful to create a 2-dimensional array that will store the concentration of particles for every position and time. A simple initial condition ($t=0$) is a point source with a given concentration in the center of the grid ($\rho(i_{\text{center}},0)=1$) and 0 otherwise. A diffusion coefficient of $D=1\text{E-}3$ is appropriate for a spatial grid size of 100 ($\Delta x=1/100$) and 1000 time iterations ($\Delta t=1/1000$). Use periodic boundary conditions to find the particle densities at the edges of the box.

In [4]:
```
           # Diffusion coefficient

e = 100    # size of discretized space in x
           # actual length of space considered
rid_size

000        # number of time iterations
           # time to be explored
t_iter

eros((x_grid_size, t_iter))  # particle concentration at any point in spa

_size//2, 0] = 1.0  # set the initial configuration, t=0, of the system a

ange(t_iter-1):
in range(x_grid_size-1):
o[i, n+1] = rho[i, n] +(D*dt/dx**2)*(rho[i+1, n] + rho[i-1, n] - 2*rho[i,
re using periodic boundary conditions, so we must be careful at the edges
the left edge, i=0, there is no problem because i=-1 gives us the value a
for the right edge, i=x_grid_size-1, the i-1 term must be set to 0 to get
grid_size-1, n+1] = rho[x_grid_size-1, n] +(D*dt/dx**2)*(rho[0, n] + rho[

()
mesh(rho, vmin=0, vmax=0.5, cmap='pink')
('Time (n)')
('Position (i)')
r'Concentration - $\rho (i, n)$')
```

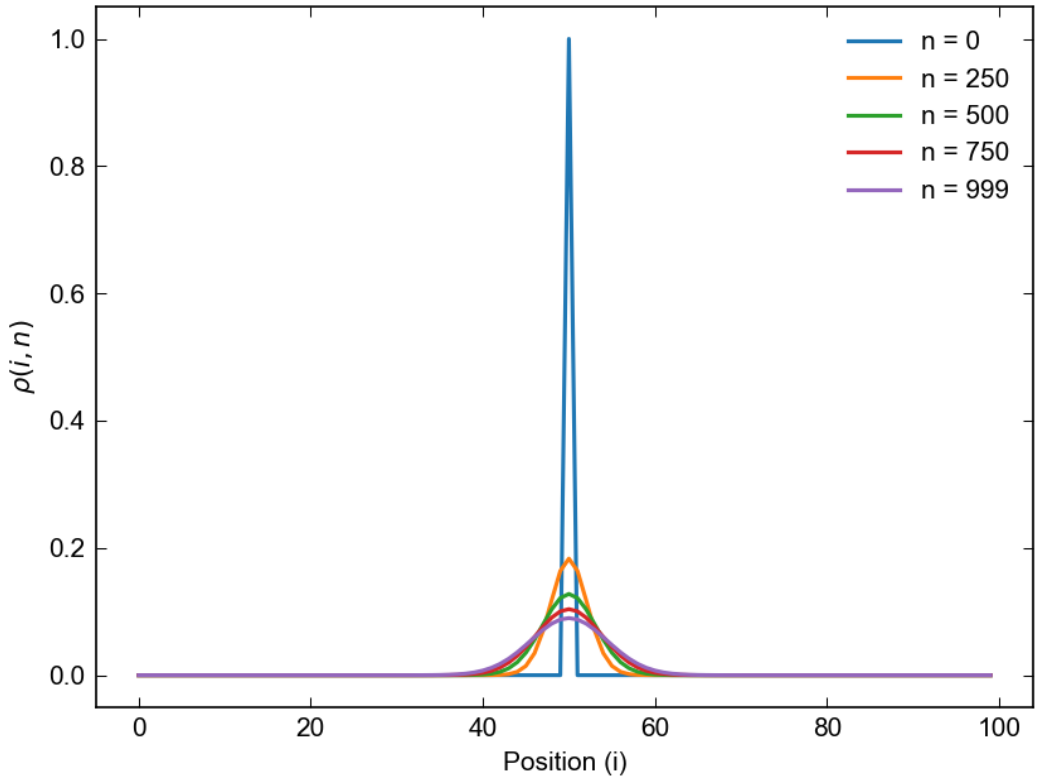
<IPython.core.display.Javascript object>



Out[4]: <matplotlib.text.Text at 0x10b0478d0>

In [5]:
```
plt.figure()
for t in (0, t_iter//4, t_iter//2, 3*t_iter//4, t_iter-1):
    plt.plot(rho[:, t], label = 'n = {0:d}'.format(t))

plt.xlabel('Position (i)')
plt.ylabel(r'$\rho (i,n)$')
plt.legend()
```

```
<IPython.core.display.Javascript object>
```



Out[5]: `<matplotlib.legend.Legend at 0x10b09a8d0>`

Point source and random walker

Last time we talked about the probability of finding a random walker at some position x after taking so many steps, N and obtained

$$P(x;N) = \frac{1}{\sqrt{2\pi N a^2}} e^{-x^2/2Na^2}.$$

The solution to the 1-dimensional diffusion equation for an ideal point source is also a Gaussian function

$$\rho(x,t) = \frac{\rho_0}{\sqrt{4\pi D t}} e^{-x^2/4Dt},$$

where the mean squared displacement is $\left\langle x^2 \right\rangle = 2Dt$.

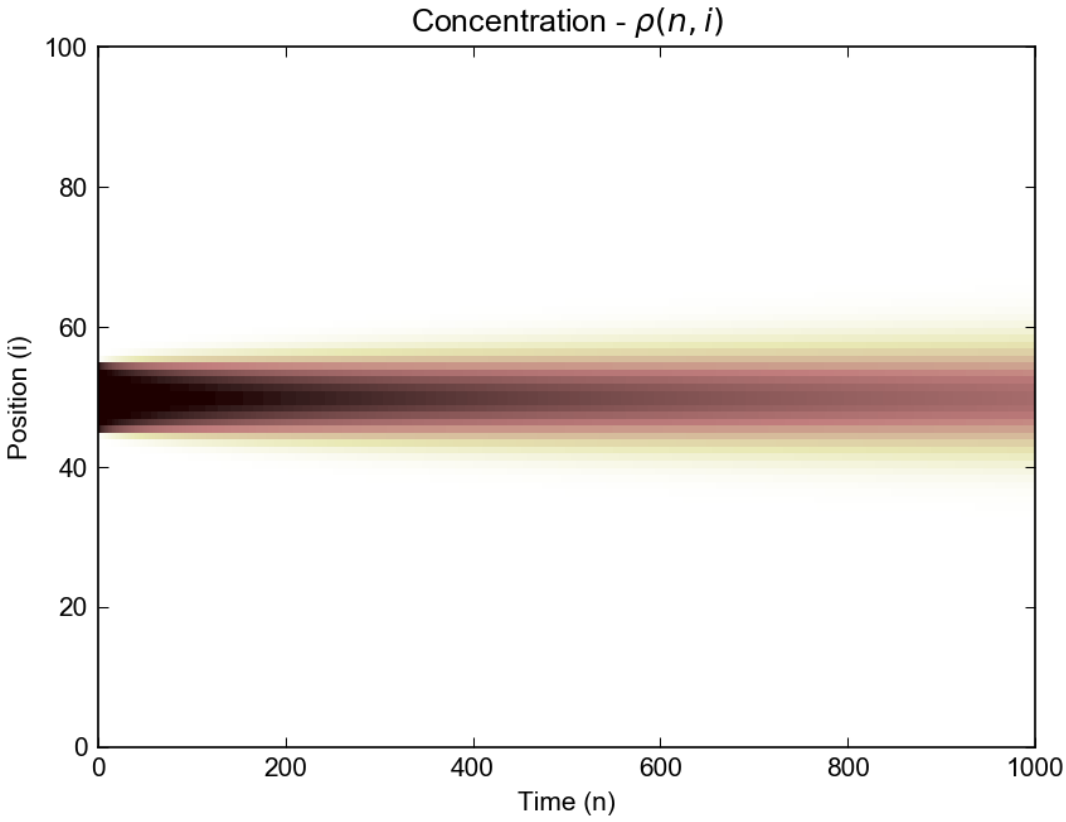
Different initial conditions

Programming challenge

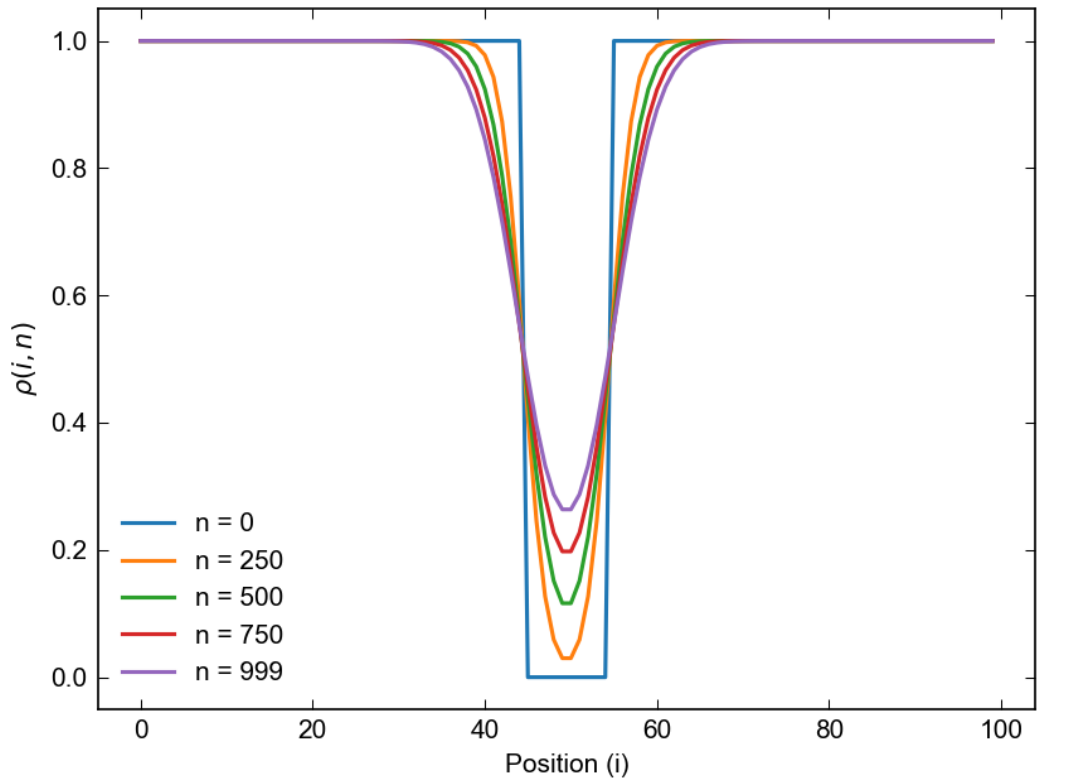
Change the A initial conditions ($t = 0$) so that there is a uniform concentration of particles over the entire box ($\rho(i, 0) = 1$) except for a 'bleached' spot at the center where the concentration is 0 ($\rho(i_{\text{center}} \pm 5, 0) = 0$). A diffusion coefficient of $D = 1\text{E-}3$ is appropriate for a spatial grid size of 100 ($\Delta x = 1/100$) and 1000 time iterations ($\Delta t = 1/1000$). Use periodic boundary conditions to find the particle densities at the edges of the box.

In [6]:
```
= 1E-3             # Diffusion coefficient

grid_size = 100    # size of discretized space in x
= 1.0              # actual length of space considered
 = L/x_grid_size

iter = 1000        # number of time iterations
ax = 1.0           # time to be explored
 = tmax/t_iter

o = np.ones((x_grid_size, t_iter))  # particle concentration at any point

r i in range(x_grid_size//2 - 5, x_grid_size//2 + 5):
  rho[i, 0] = 0.0  # set the initial configuration, t=0, of the system as
                   # spot in the center

r n in range(t_iter-1):
  for i in range(x_grid_size-1):
      rho[i, n+1] = rho[i, n] +(D*dt/dx**2)*(rho[i+1, n] + rho[i-1, n] -
  # we are using periodic boundary conditions, so we must be careful at t
  # for the left edge, i=0, there is no problem because i=-1 gives us the
  # but for the right edge, i=x_grid_size-1, the i-1 term must be set to
  rho[x_grid_size-1, n+1] = rho[x_grid_size-1, n] +(D*dt/dx**2)*(rho[0, n

t.figure()
t.pcolormesh(rho, vmin=0, vmax=1, cmap='pink')
t.xlabel('Time (n)')
t.ylabel('Position (i)')
t.title(r'Concentration - $\rho (n, i)$')

t.figure()
r t in (0, t_iter//4, t_iter//2, 3*t_iter//4, t_iter-1):
  plt.plot(rho[:, t], label = 'n = {0:d}'.format(t))

t.xlabel('Position (i)')
t.ylabel(r'$\rho (i,n)$')
t.legend()
```

<IPython.core.display.Javascript object>




```
<IPython.core.display.Javascript object>
```




```
Out[6]: <matplotlib.legend.Legend at 0x10c3344e0>
```

In [ ]: