PROBLEM 1 - Solution of 1D Fick Law using FTCS method

Gisela Martí Guerrero, ICC 2023

This code implements a simulation of one-dimensional diffusion using the Forward-Time Central-Space (FTCS) method, specifically for the Fick's 2nd law for 1D diffusion diffusion equation:

$$\frac{\delta c}{\delta t} = D\nabla^2 c$$
$$-k\frac{\delta^2 c}{\delta x^2} + \frac{\delta c}{\delta t} = 0$$

The FTCS method is a finite difference scheme commonly used for solving partial differential equations. It can be described with the following equation:

$$c_i^{k+1} = c_i^k + \frac{D\Delta t}{\Delta x^2}[c_{i+1}^k + c_{i-1}^k - 2c_i^k]$$

The program calculates three different scenarios, depending on the initial conditions:

1. "Source" of constant concentration to fill a segment L with one wall end.
2. Concentration step with two wall ends.
3. "Droplet" diffusion in the center of the system with periodic boundary conditions.

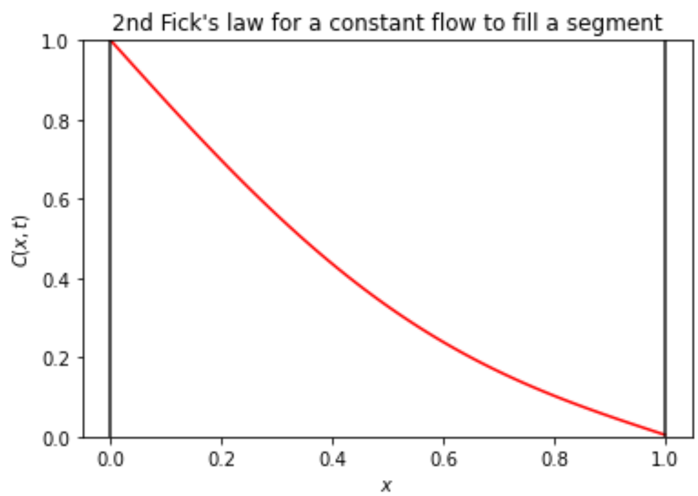
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation,PillowWriter
```

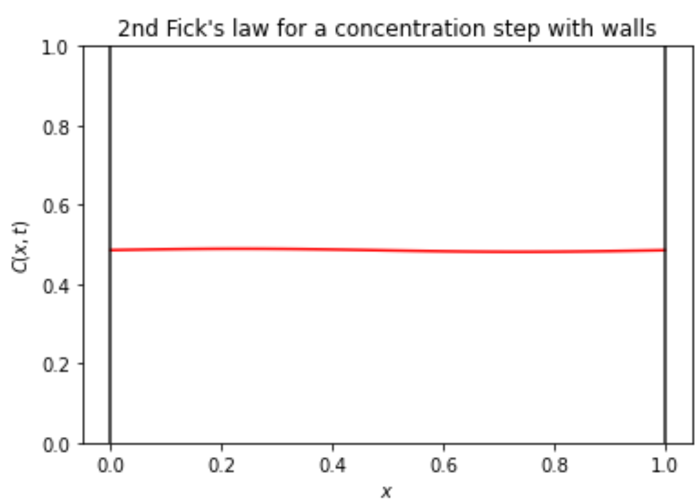
In [2]:
```python
def Fick_FTCS(D,c0,L,trange,pbc,title,animation_name):
    # Initialization
    t0, tf = trange            # Initial and final time
    dx = 0.01                   # Spatial step
    dt = 0.5*dx**2/(2*D)        # Time step
    t = np.arange(t0,tf+dt,dt)  # Time grid
    x = np.arange(0,L+dx,dx)    # Spatial grid

    # Initial conditions
    c_old = np.zeros(len(x))
    c_new = np.zeros(len(x))

    if animation_name == "Fick1.gif":
        c_old[0] = c0
        c_new[0] = c0
    elif animation_name == "Fick2.gif":
        c_old[:int(len(x)/2)] = c0
    else:
        c_old[int(len(x)/2)] = c0

    # Integration Loop
    i_frame = 0
    c_init = c_old.copy()
    ct = [0 for _ in range(animation_frames+1)]

    for i,ti in enumerate(t):
        # Boundary Conditions
        if pbc: # Periodic
            c_new[0] = c_new[0] + D*dt/dx**2 * (c_new[-1] + c_new[1] - 2*c_new[0])
            c_new[-1] = c_new[-1] + D*dt/dx**2 * (c_new[0] + c_new[-2] - 2*c_new[-1])
        else:   # Walls
            c_new[0] = c0
            c_new[-1] = c_new[-1] + D*dt/dx**2 * (0 + c_new[-2] - 2*c_new[-1])

        for j, xi in enumerate(x,start=1):
            try:
                c_new[j] = c_old[j] + D*dt/dx**2 * (c_old[j+1] + c_old[j-1] - 2*c_old[j
                c_old[j] = c_new[j]
            except IndexError: pass
        c_old = c_new

        if i%int(len(t)/animation_frames) == 0 :
            ct[i_frame] = np.array(c_new)
            i_frame +=1
    ct = np.array(ct,dtype=object)

    # Start animation
    fig,ax = plt.subplots()

    def GIF(frame):
        """Function that creates a frame for the GIF."""
        ax.clear()
        ct_frame, = ax.plot(x, ct[frame], c="red")
        wall1 = ax.axvline(L, ymin=0, c="k", alpha=0.9)
        wall2 = ax.axvline(0, ymin=0, c="k", alpha=0.9)
        ax.set_xlabel("$x$")
        ax.set_ylabel("$C(x,t)$")
        ax.set_ylim([0,c0])
        ax.set_title(title)
        return ct_frame, wall1, wall2
```

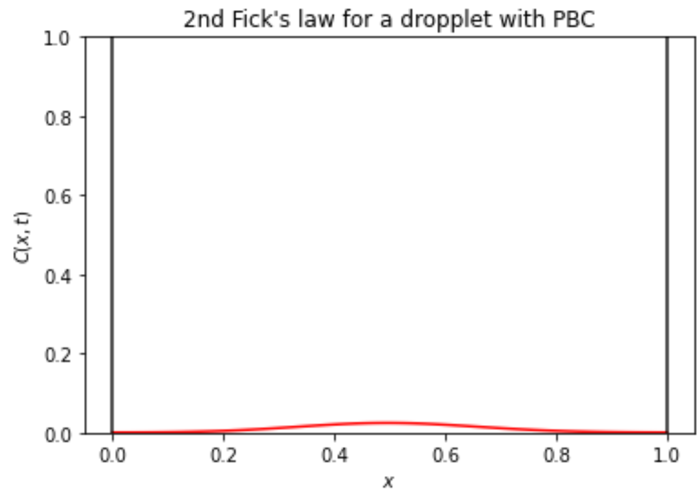
```
    animation = FuncAnimation(fig,GIF,frames=animation_frames,interval=20,blit=True,rep
    animation.save(animation_name,dpi=120,writer=PillowWriter(fps=25))
    plt.show()

    return fig
```

In [3]:

```python
# Simulation
D = 0.01
c0 = 1
L = 1.0
animation_frames = 100

# FIRST CASE
title1 = "2nd Fick's law for a constant flow to fill a segment"
animation_name1 = "Fick1.gif"
trange1 = (0,10)
fig1 = Fick_FTCS(D,c0,L,trange1,False,title1,animation_name1)
plt.show()

# SECOND CASE
title2 = "2nd Fick's law for a concentration step with walls"
animation_name2 = "Fick2.gif"
trange2 = (0,10)
fig1 = Fick_FTCS(D,c0,L,trange2,True,title2,animation_name2)
plt.show()

# THIRD CASE
title3 = "2nd Fick's law for a dropplet with PBC"
animation_name3 = "Fick3.gif"
trange3 = (0,1)
fig3 = Fick_FTCS(D,c0,L,trange3,True,title3,animation_name3)
plt.show()
```







All simulations are initialized with the same conditions. We can see the following results:

1. For the first case, we cans see that the flux begins to fill the box.
2. For the econd case, we can see that the high concentrations in the first section diffuse into all the box, ending in an averaged value for the concentration in all the box ($C(x, t) = 0.5$).
3. For the third case, we can see that the dropplet diffues through the entire lengths of the box, also ending in an averaged value.