Finite Difference 1 - Diffusion

November 16, 2018

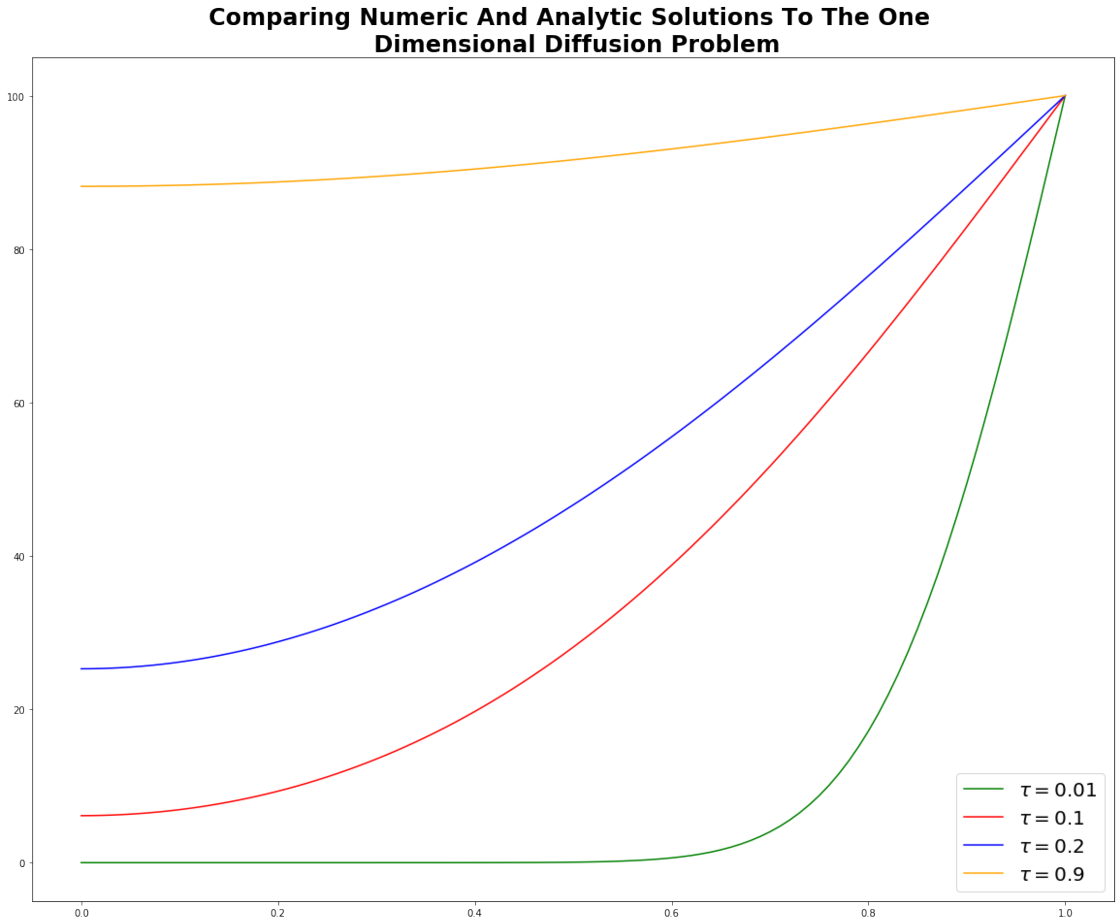
0.1 Analytic Solution

In this section we implement a numeric solution to the diffusion equation. We have already found the analytic solution, which is

$$C_-(x,0) = C_b 1 - \frac{4}{\pi} \sum_n \frac{(-1)^m}{(2m+1)} \exp - \frac{(2n+1)\pi}{2\delta}^2 \frac{D\_t}{\delta^2} \cos \frac{(2m+1)\pi}{2} \frac{x}{\delta}. \tag{1}$$

In this section we will compute the diffusion equation numerically and add the chemical reaction as a border condition.

```
In [6]: import analytic
        import matplotlib.pyplot as plt
        import numpy as np

        M = 100
        xi = np.linspace(0,1, M+1)

        Cm = analytic.C_an
        mw = 4
        fs = 24
        n = 1000
        fig = plt.figure(figsize=(20,16))
        plt.title('Comparing Numeric And Analytic Solutions To The One \n Dimensional Diffusion
        plt.plot(xi, Cm(xi, 0.01), 'g-', label=r'$\tau=0.01$')
        plt.plot(xi, Cm(xi, 0.1), 'r-', label=r'$\tau = 0.1$')
        plt.plot(xi, Cm(xi, 0.2), 'b-', label = r'$\tau=0.2$')
        plt.plot(xi, Cm(xi, 0.9), '-', color="orange", label = r'$\tau=0.9$')
        plt.legend(fontsize = fs-4)
        plt.savefig('../../../img/concentration-diffusiononly-comparison.eps', format='eps', dpi
        plt.show()
```



0.2 Numeric Solution

We will use an implicit scheme to find the numerical solution. This means that, in approaching the finite difference method, we will compute the spacial derivative at time step $n+1$. Consider the one dimensional diffusion equation

$$\frac{\partial C}{\partial t} = D\frac{\partial^2 C}{\partial x^2}. \tag{2}$$

We will define $x = \delta\xi$ where δ is the width of the laminar flow sheet.

$$\frac{\partial C}{\partial t} = \frac{D}{\delta^2}\frac{\partial^2 C}{\partial \xi^2}. \frac{\partial C}{\partial \tau} = \frac{\partial^2 C}{\partial \xi^2}. \tag{3}$$

$$\tag{4}$$

We will discretize the derivative as follows,

$$\frac{\partial C}{\partial t}^{n+1,k} = \frac{C^{n+1,k} - C^{n,k}}{\Delta t}, \tag{5}$$

$$\frac{\partial^2 C}{\partial \xi^2}^{n+1,k} = \frac{C^{n+1,k-1} - 2C^{n,k} + C^{n+1,k+1}}{\Delta \xi^2}. \tag{6}$$

Replacing int equation 2 we get

$$-\alpha\rho^{n+1,k-1} + (1+2\alpha)\rho^{n+1,k} - \alpha\rho^{n+1,k+1} = \rho^{n+1,k} \tag{7}$$

In particular, for a given n value, the equations for $k = 0$ and $k = m$ are

$$-\alpha\rho^{n+1,-1} + (1+2\alpha)\rho^{n+1,0} - \alpha\rho^{n+1,1} = \rho^{n+1,0}, \tag{8}$$

$$-\alpha\rho^{n+1,m-1} + (1+2\alpha)\rho^{n+1,m} - \alpha\rho^{n+1,m+1} = \rho^{n+1,m}. \tag{9}$$

The border conditions for our system (in discretized form) are

$$\rho^{n+1,-1} = \rho^{n+1,0}, \tag{10}$$

$$\rho^{n+1,m} = 0. \tag{11}$$

Therefore, equations 9 yield

$$(1+\alpha)\rho^{n+1,0} - \alpha\rho^{n+1,1} = \rho^{n+1,0}, \tag{12}$$

$$-\alpha\rho^{n+1,m-1} + (1+2\alpha)\rho^{n+1,m} = \rho^{n+1,m}. \tag{13}$$

We want to put these equations in matrix form. Let

$$\rho^n = \begin{bmatrix} \rho^{n,0} \\ \rho^{n,1} \\ \vdots \\ \rho^{n,m-1} \\ \rho^{n,m} \end{bmatrix} \tag{14}$$

$$\underline{\mathbf{A}} = \begin{bmatrix} (1+\alpha) & -\alpha & 0 & 0 & \cdots & 0 \\ -\alpha & (1+2\alpha) & -\alpha & \cdots & 0 & 0 \\ \vdots & \cdots & \ddots & \ddots & \ddots & \\ \vdots & \cdots & 0 & -\alpha & (1+2\alpha) & -\alpha \\ 0 & \cdots & 0 & 0 & -\alpha & (1+2\alpha) \end{bmatrix} \tag{15}$$

Equations 7 can be expressed as

$$\underline{\mathbf{A}}\mathbf{æ^n} = \mathbf{æ^n}. \tag{16}$$

Considering the initial conditions **??**, we get

$$\rho^{0,k} = -C_b, \tag{17}$$
$$k \in [0, 1, ..., m]. \tag{18}$$

Now we are ready to start iterating this matrix equation to get the time evolution.
We will use the parameters $\xi = x/\delta$ and $\tau = t/\delta^2$ as the parameters of the equation.

In [7]:
```
#imports
import numpy as np
import matplotlib.pyplot as plt
from scipy.sparse import diags

# Define grid parameters
N = 100
tau = np.linspace(0,1, N+1)

dt = 1/(N+1)
dx = 1/(M+1)
a = dt / dx ** 2

# Define the coefficient matrix
di = ( 1 + 2 * a ) * np.ones(M+2)
di[0] = ( 1 + a )
A = diags(np.array([- a * np.ones(M+2), di, - a * np.ones(M+2)]), [-1, 0, 1], shape=(M+1
A_inv = np.asarray(np.linalg.inv(A))


# Set up initial conditions for \rho
Cb = 100

rho = np.zeros([N+1, M+1])
rho[0, :] = - Cb


for n in range(0, N):
    rho[n+1, :] = np.matmul(A_inv, rho[n, :])
```

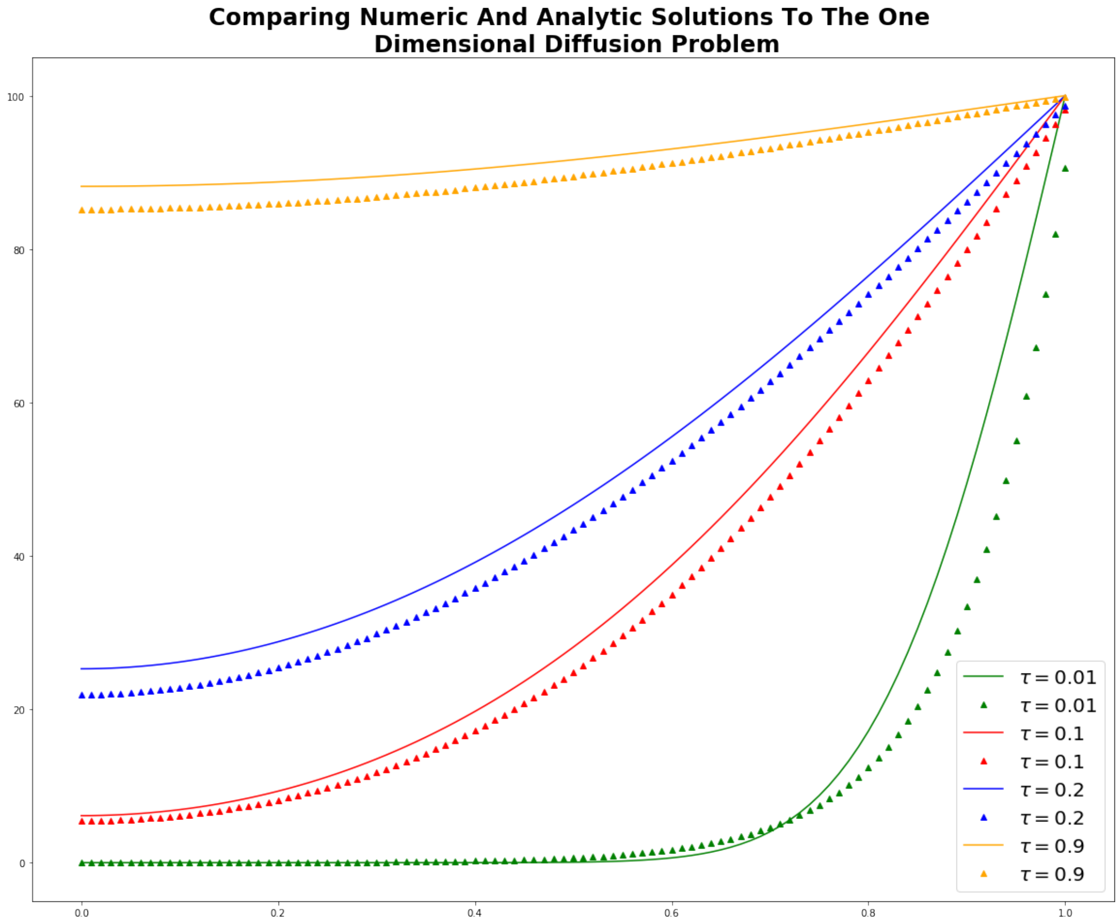
In [8]:
```
def C(t):


    rho = np.zeros([N+1, M+1])
    rho[0, :] = - Cb
```

```
    for n in range(0, N):
        rho[n+1, :] = np.matmul(A_inv, rho[n, :])

    n = int(t/dt)

    return Cb * np.ones(M+1) + rho[n, :]
```

1 Comparison With Numerical Results

We will import the analytical solution and compare it with the numerical results for eacht τ

```
In [9]: #Cm is the imported analytical solution

        mw = 4
        fs = 24
        n = 1000
        fig = plt.figure(figsize=(20,16))
        plt.title('Comparing Numeric And Analytic Solutions To The One \n Dimensional Diffusion
        plt.plot(xi, Cm(xi, 0.01), 'g-', label=r'$\tau=0.01$')
        plt.plot(xi, C(0.01), 'g^', label=r'$\tau=0.01$')
        plt.plot(xi, Cm(xi, 0.1), 'r-', label=r'$\tau = 0.1$')
        plt.plot(xi, C(0.1), 'r^', label=r'$\tau = 0.1$')
        plt.plot(xi, Cm(xi, 0.2), 'b-', label = r'$\tau=0.2$')
        plt.plot(xi, C(0.2), 'b^', label = r'$\tau=0.2$')
        plt.plot(xi, Cm(xi, 0.9), '-', color="orange", label = r'$\tau=0.9$')
        plt.plot(xi, C(0.9), '^', color="orange", label = r'$\tau=0.9$')
        plt.legend(fontsize = fs-4)
        plt.savefig('../../../img/concentration-diffusiononly-comparison.eps', format='eps', dpi
        plt.show()
```



Δ marks are the numeric values and continuous lines are the analytical values.

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```