Final Rahul Deshmukh (PUID 0030004932)

December 14, 2017

```
In [1]: import numpy as np
        from sympy import *
        import matplotlib.pyplot as plt
```

Problem 1:
The given equation

$$\frac{\partial T}{\partial t} + v\frac{\partial T}{\partial y} = \frac{4h}{\rho c_p d}(T_g(y) - T)$$

is a first order partial differential equation(advection equation) with Boundary condition :

$$T(0,t) = 3$$

and Initial condition:

$$T(y,0) = T_g(y) = (3 + 10e^{\frac{-1}{2y}})$$

the domain of the body is given as

$$0 \leq y \leq 10$$

Now we need to discretize the spatial and time domain using first oder fininte differences. As we know the boundary term at y=0, using a Backward difference for spatial coordinate is suitable and a forward difference for time.

The discretized equation can be written as:

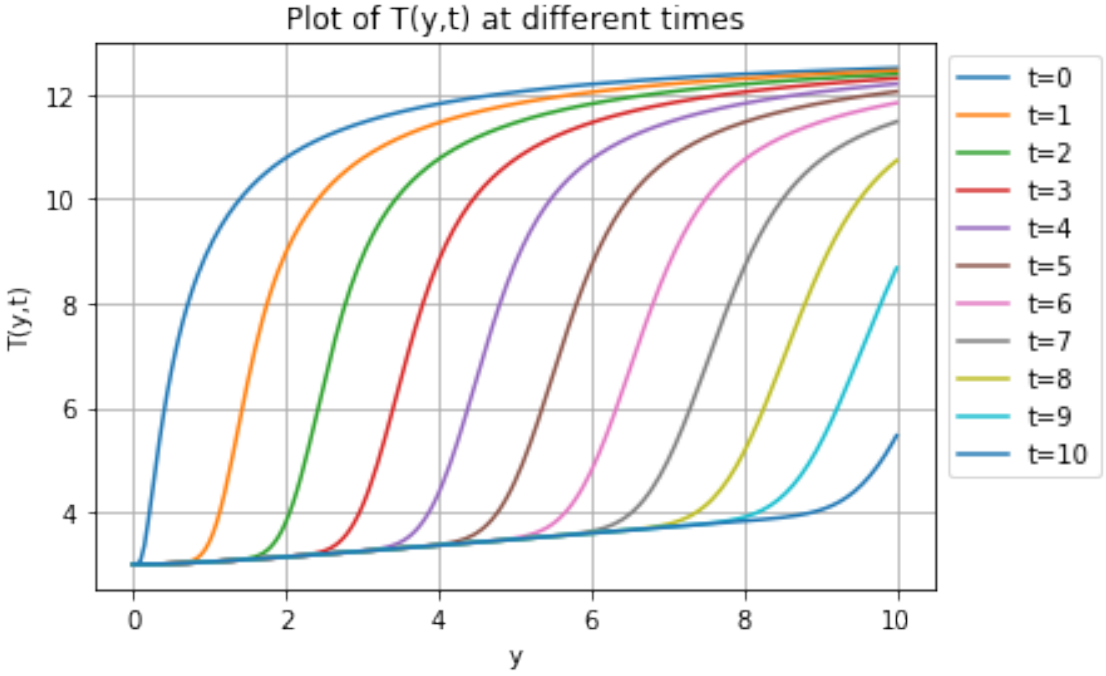
$$\frac{T_i^{k+1} - T_i^k}{\Delta t} + v\frac{T_i^k - T_{i-1}^k}{\Delta x} = \frac{4h}{\rho c_p d}(T_g(y_i) - T_i^k)$$

and we need to solve for T_i^{k+1} by advancing in time with this three point stencil

```
In [2]: def solve_advection_PDE(a,b,tf,dx,dt,v,c,ua,ut0,f):#c=4h/pcd and f is the driving force
            N=int((b-a)/dx); M=int((tf)/dt);
            u=np.zeros(((N+1),(M+1)))#u(y,t)
            for i in range(0,N+1):
                u[i,0]=ut0(i*dx)#assinging the initial condition
            for k in range(0,M+1):
                u[0,k]=ua(k*dt)#assinging the BC at y=a
            for k in range(1,M+1):
                for i in range(1,N+1):
                    u[i,k]=u[i,k-1]+dt*(c*(f(i*dx)-u[i,k-1])-(v/dx)*(u[i,k-1]-u[i-1,k-1]))
            return(u)
```

In [3]:
```
a=0;b=10;
v=1;h=990;p=1000;cp=4180;d=70/1000;
c=(4*h)/(p*cp*d);
dy=0.05; dt=0.005
def Tg(y):
    if y==0:
        return(3)
    else:
        return((3+10*np.e**(-1/(2*y))))
def Ta(t):
    return(3)
tf=10
T=solve_advection_PDE(a,b,tf,dy,dt,v,c,Ta,Tg,Tg)
```

In [4]:
```
y = np.linspace(a,b,int((b-a)/dy)+1)
t = np.linspace(0,tf,int(tf/dt)+1)
for i in range(0,tf+1):
    z=int(i/dt)
    plt.plot(y,T[:,z],label='t='+str(i))
plt.legend(bbox_to_anchor=(1,1),loc=0)
plt.xlabel('y')
plt.ylabel('T(y,t)')
plt.title('Plot of T(y,t) at different times')
plt.grid('on')
plt.show()
```



Problem 2:
PDE:

$$C\frac{\partial u}{\partial x} = D\frac{\partial^2 u}{\partial x^2}$$

the above PDE is similar to Heat equation with x analogous to t
BC:

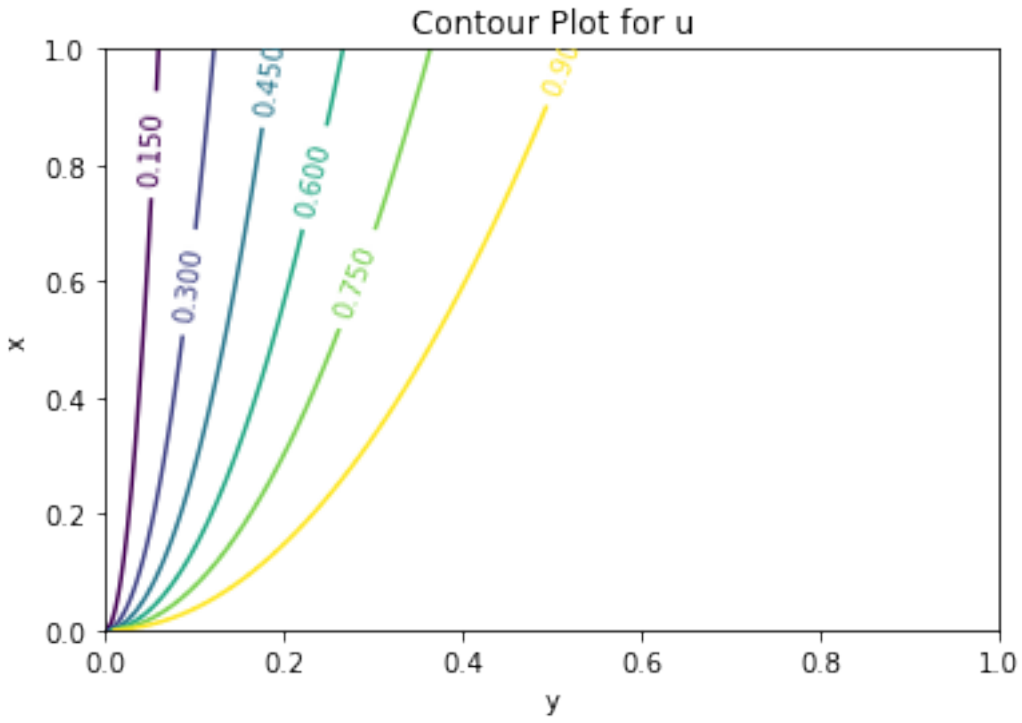
$$u(x, y = 0) = 0$$
$$u(x, y = 1) = 1$$
$$u(x = 0, y) = 1$$

a)

```
In [5]: def solve_heat_PDE(a,b,tf,dx,dt,c,ua,ub,ut0):#c is constant appearing in the PDE Ut=c*Ux
            if dt>dx**2/(2*c):
                print('The choice of element size and time step is not stable')
            N=int((b-a)/dx); M=int((tf)/dt);
            u=np.zeros(((N+1),(M+1)))#u(x,t)
            for i in range(0,N+1):
                u[i,0]=ut0(i*dx)#assinging the initial condition
            for k in range(0,M+1):
                u[0,k]=ua(k*dt)#assining the BC at x=a
                u[-1,k]=ub(k*dt)#assining the BC at x=b
            for k in range(1,M+1):
                for i in range(1,N):
                    u[i,k]=u[i,k-1]+c*(dt/(dx**2))*(u[i+1,k-1]-2*u[i,k-1]+u[i-1,k-1])
            return(u)

In [6]: a=0;b=1;xf=1;
        dx=0.001;dy=0.01;
        C=1;D=0.05;
        c=D/C;
        def ua(x):
            return(0)
        def ub(x):
            return(1)
        def ut0(y):
            return(1)
        u=solve_heat_PDE(a,b,xf,dy,dx,c,ua,ub,ut0)

In [7]: x = np.linspace(a,b,int((b-a)/dy)+1)
        y = np.linspace(0,xf,int(xf/dx)+1)
        X, Y = np.meshgrid(x,y)
        cp=plt.contour(X,Y,np.transpose(u))
        plt.clabel(cp, inline=True,fontsize=10)
        plt.title('Contour Plot for u')
        plt.xlabel('y')
        plt.ylabel('x')
        plt.show()
```



b) **Finding the analytical solution:**

Let us convert the above PDE into a form of heat equation (for convenience) by replacing x with t and y with x we get:

$$\frac{\partial u}{\partial t} = c^2 \frac{\partial^2 u}{\partial x^2}$$

where $c^2 = C/D = 0.05$

we have the following **boundary conditions** for the problem:

$$u(0,t) = 0$$

$$u(1,t) = 1$$

and the **initial condition**:

$$u(x,0) = 1$$

Now lets contruct the steady state solution for this problem:

for steady state we have:

$$\frac{\partial u}{\partial t} = 0 = c^2 \frac{\partial^2 u}{\partial x^2}$$

$$\Rightarrow \frac{\partial^2 u}{\partial x^2} = 0$$

therefore $u_\infty = ax + b$

Now u_∞ needs to satisy the Boundary conditions Therefore $u_\infty(0) = u(0,t) = 0 = a * 0 + b \Rightarrow b = 0$

Also $u_\infty(1) = u(1,t) = 1 = a*1 \Rightarrow a = 1 \Rightarrow u_\infty = x$
Now, lets say that $\bar{u}(x,t) = u(x,t) - u_\infty$
we can observe that $\frac{\partial \bar{u}}{\partial t} = \frac{\partial u}{\partial t}$ and $\frac{\partial^2 \bar{u}}{\partial x^2} = \frac{\partial^2 u}{\partial x^2}$ i.e. $\bar{u}(x,t)$ satisfies the PDE
The **boundary conditions** for $\bar{u}(x,t)$ will be:

$$\bar{u}(0,t) = u(0,t) - u_\infty(0) = 0 - 0 = 0$$

$$\bar{u}(1,t) = u(1,t) - u_\infty(1) = 1 - 1 = 0$$

And **initial condition** $\bar{u}(x,t)$ is:

$$\bar{u}(x,0) = u(0,t) - u_\infty(0) = 1 - x$$

Using Separation of variables we have $\bar{u}(x,t) = F(x)G(t)$ Using the PDE we get

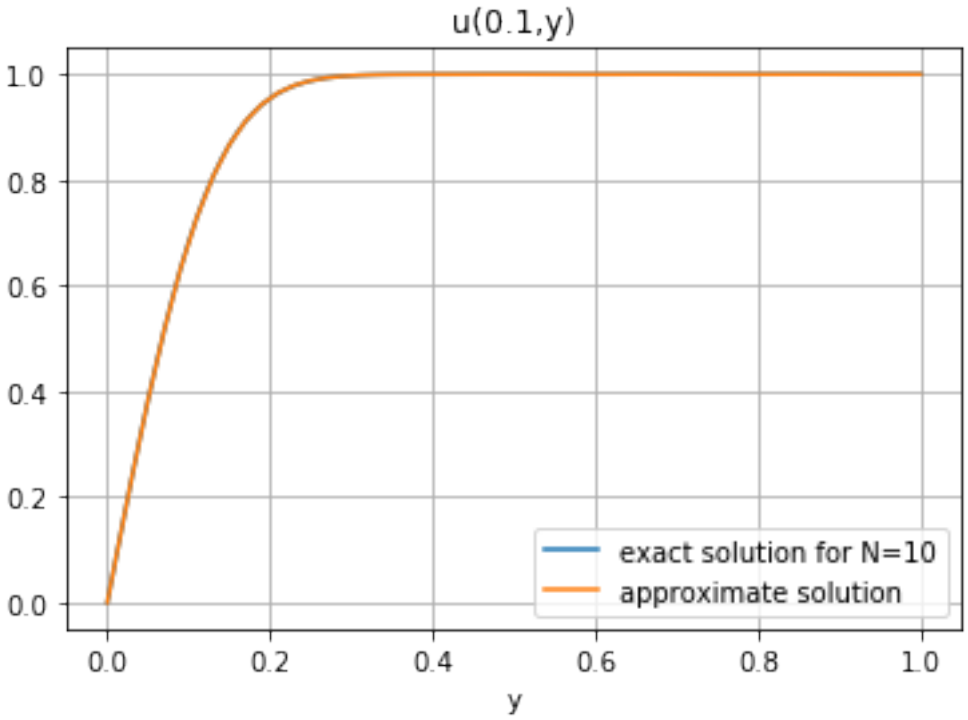
$$\frac{\dot{G}}{c^2 G} = \frac{F''}{F} = k$$

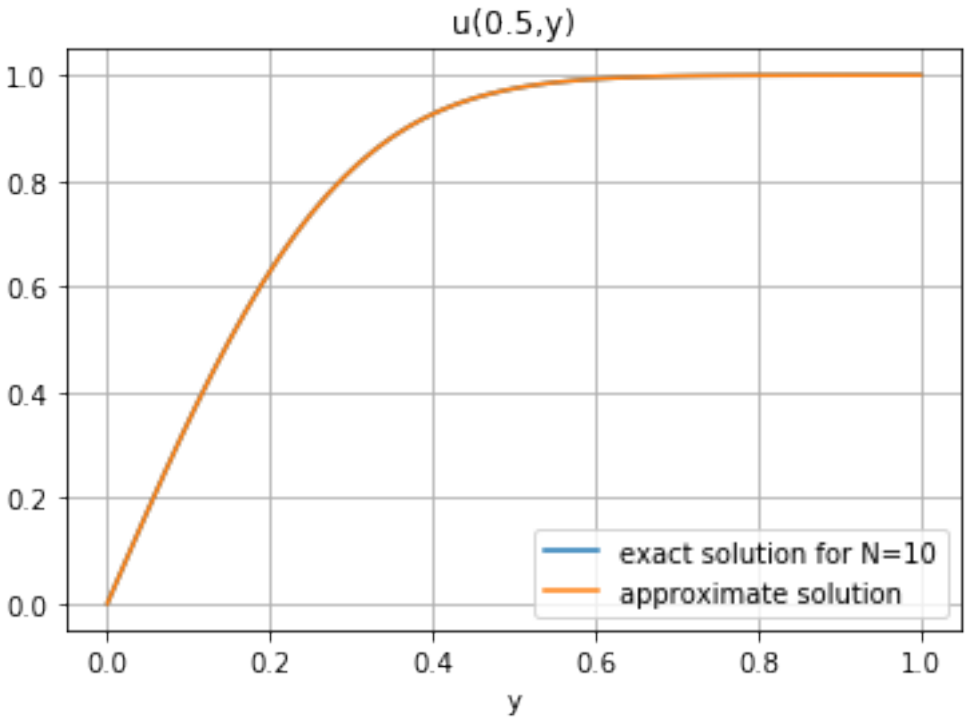
If $k \geq 0$ we will obtain a trivial solution for F(x)
For $k = -p^2$:
$F(x) = Acos(px) + Bsin(px)$
using BCs we get $F_n(x) = sin(n\pi x)$ and $p = n\pi$ similarly we get $G_n(t) = B_n e^{-(n\pi c)^2 t}$
$\bar{u}(x,t) = \sum_{n=1}^{\infty} B_n (sin(n\pi x)) e^{-(n\pi c)^2 t}$
Now using initial condition we have $\bar{u}(x,0) = 1 - x$
Therefore, B_n are the coefficients of the Fourier sine series of odd periodic extension of $1 - x$
$B_n = \frac{2}{1}\left(\int_0^1 (1-x) sin(n\pi x) dx \right) = \frac{2}{1}\left((1-x)\frac{(-cos(n\pi x))}{n\pi}|_0^1 - (-1)\frac{(-sin(n\pi x))}{(n\pi)^2}|_0^1 \right) = \frac{2}{n\pi}$
$\bar{u}(x,t) = \sum_{n=1}^{\infty} \frac{2}{n\pi} (sin(n\pi x)) e^{-(n\pi c)^2 t}$
$u(x,t) = \bar{u}(x,t) + u_\infty$
$\Rightarrow u(x,t) = x + \sum_{n=1}^{\infty} \frac{2}{n\pi} (sin(n\pi x)) e^{-(n\pi c)^2 t}$
Now let us change back to our initial variables for the given equaition
$\Rightarrow u(y,t) = y + \sum_{n=1}^{\infty} \frac{2}{n\pi} (sin(n\pi y)) e^{-(n\pi c)^2 x}$

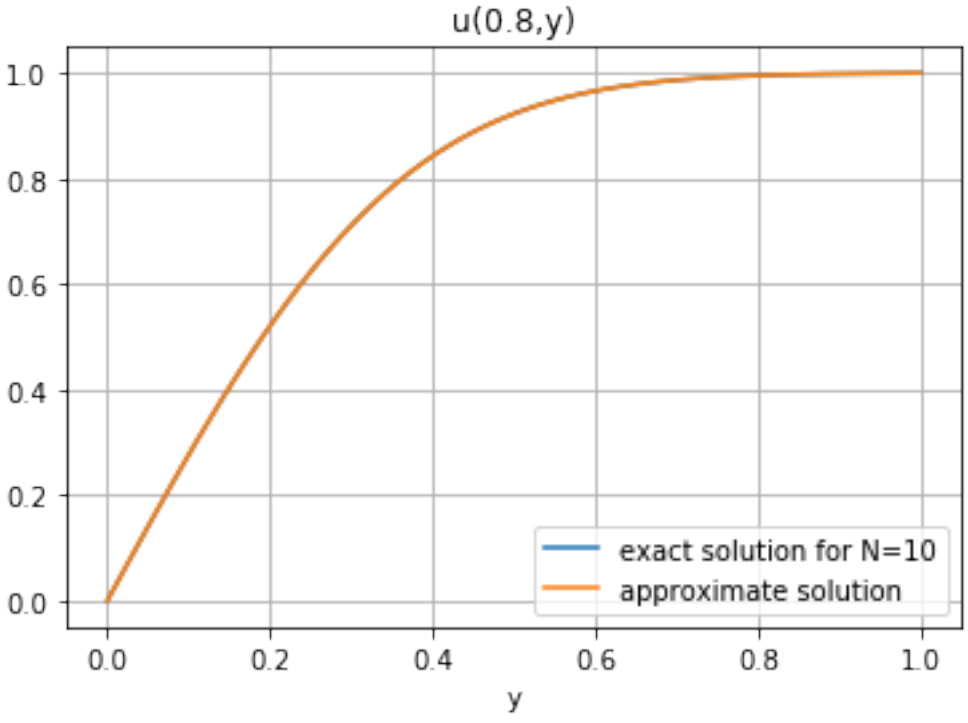
```
In [8]: def exact(x,t,c,N):
            s=0
            for i in range(1,N+1):
                s=s+(2/(i*np.pi))*np.sin(i*np.pi*x)*(np.e**(-((i*np.pi*c)**2)*t))
            s=s+x
            return(s)
```

```
In [9]: x1=np.array([0.1,0.5,0.8])
        ey=np.zeros((int((b-a)/dy)+1,len(x1)))
        N=10 # number of terms in Fourier sine series
        Y = np.linspace(a,b,int((b-a)/dy)+1)
        for i in range(0,len(x1)):
            for j in range(0,int((b-a)/dy)+1):
                ey[j,i]=exact(Y[j],x1[i],np.sqrt(c),N)
            p1,=plt.plot(Y,ey[:,i],label='exact solution for N='+str(N))
            p2,=plt.plot(Y,u[:,int(x1[i]/dx)],label='approximate solution')
            plt.legend(handles=[p1,p2])
```

```
plt.title('u('+str(x1[i])+',y)')
plt.xlabel('y')
plt.grid('on')
plt.show()
```




u(0.5,y)
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
y
exact solution for N=10
approximate solution


u(0.8,y)
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
y
exact solution for N=10
approximate solution

As we can observe from the above plots, the analytical solution and the numerical solution are **coincident**. Therefore we have a good approximation