Assignment 2 ME5102

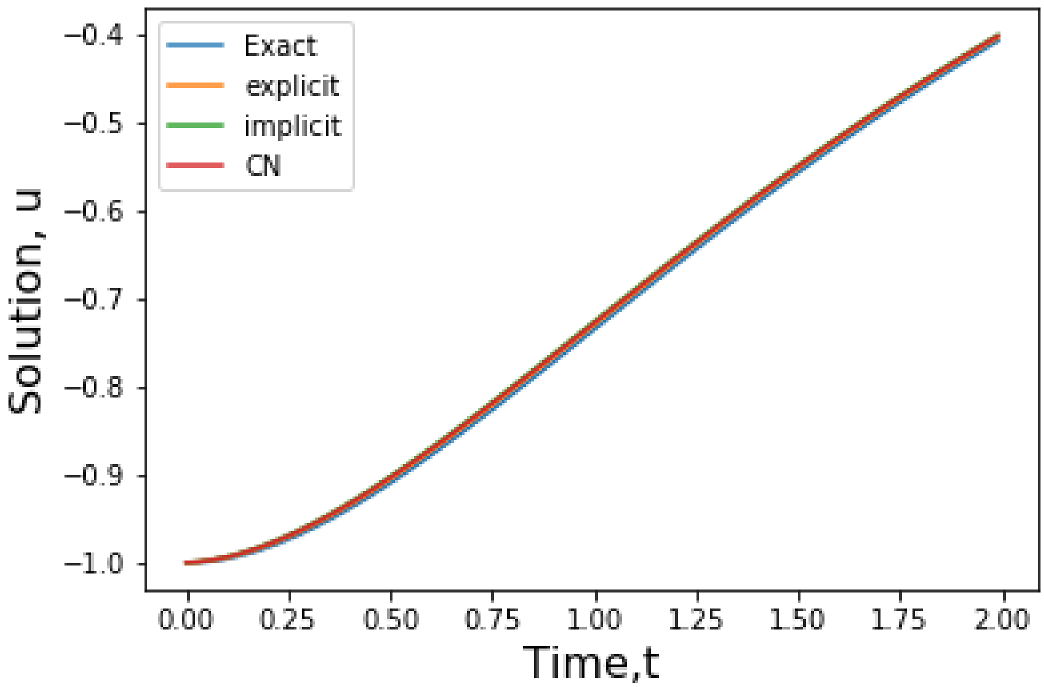
Aakash
Email: me16b001@iittp.ac.in

February 6, 2019

Problem 3: Explicit, implicit and CN time stepping schemes

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
```

```python
import math
# explicit scheme
u=[]
t=0.01
for i in range(200):
    if i==0:
        u.append(-1)
    else:
        u.append(u[i-1]+(0.01)*t*np.exp(-t))
    t=t+0.01
# implicit scheme
ub=[]
t=0.01
for i in range(200):
    if i==0:
        ub.append(-1+ (0.01)*(t+0.01)*np.exp(-(t+0.01)))
    else:
        ub.append(ub[i-1]+(0.01)*(t+0.01)*np.exp(-(t+0.01)))
    t=t+0.01
# CN scheme
uc=[]
t=0.01
for i in range(200):
    if i==0:
        uc.append(-1+(0.5)*(0.01)*(t*np.exp(-t)+(t+0.01)*np.exp(-(t+0.01))))
    else:
        uc.append(uc[i-1]+(0.5)*(0.01)*(t*np.exp(-t)+(t+0.01)*np.exp(-(t+0.01))))
    t=t+0.01
# CN scheme end
t = np.arange(0, 2, 0.01)
u_exact = -1*(t+1)*np.exp(-t)

plt.plot(t, u_exact, label="Exact")
plt.plot(t, u, label="explicit")
plt.plot(t, ub, label="implicit")
plt.plot(t, uc, label="CN")
plt.legend()
plt.xlabel('Time,t', fontsize=16)
plt.ylabel('Solution, u', fontsize=16)
plt.show()
```

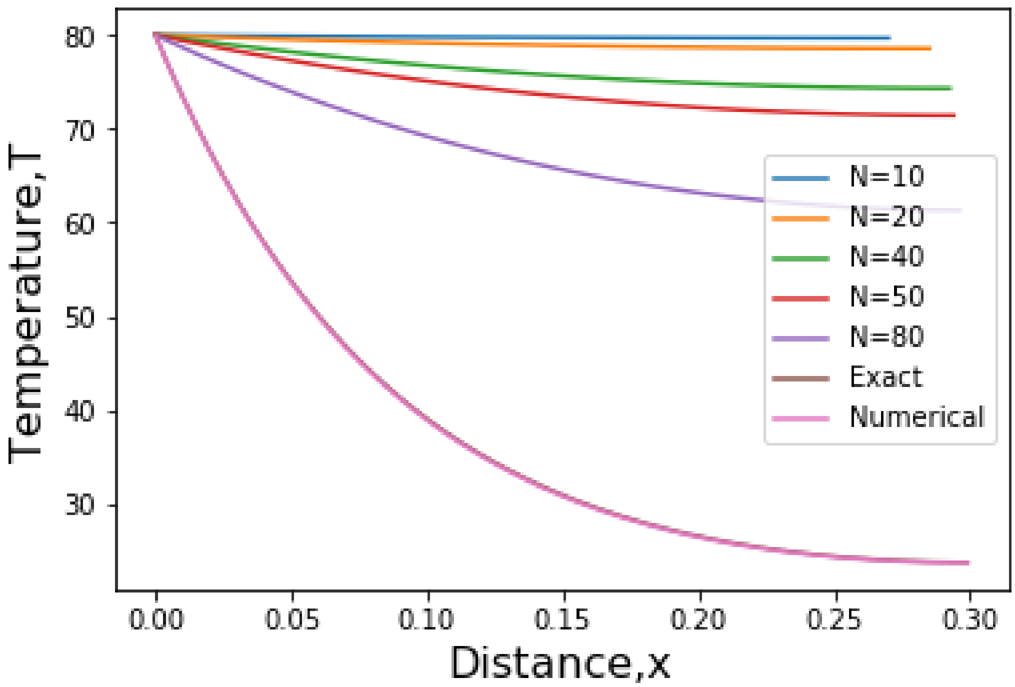


Exact
explicit
implicit
CN
Solution, u
Time,t
−0.4
−0.5
−0.6
−0.7
−0.8
−0.9
−1.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

Problem 4: 1-D Heat loss through circular fin

```
In [4]: from sympy import *
        import matplotlib.pyplot as plt
        import numpy as np
        from numpy import *
        from array import *
        from scipy.sparse import *

        def TDMA(T,d, n):
                a=[]
                b=[]
                c=[]
                cs=[]
                ds=[]
                u=[]

                for i in range(len(T)):
                        for j in range(len(T[0])):
                                if (j-i)==1:
                                        c.append(T[i][j])
                                if (i-j==1):
                                        a.append(T[i][j])
                                if i==j:
                                        b.append(T[i][j])
                # acccomodating n nodes
                tb=b[len(b)-1]
                ta=a[len(a)-1]
                a.pop()
                b.pop()
                for i in range(n-4):
                        a.append(a[len(a)-1])
                        b.append(b[len(b)-1])
                        c.append(c[len(c)-1])
                        d.append(d[len(d)-1])
                a.append(ta)
                b.append(tb)
                T = diags([b,a,c], [0,-1, 1]).todense()
                #print(T)
                #print("T",len(T))

                for i in range(len(d)):
                        u.append(0)

                for i in range(len(c)):
                        if i==0:
                                cs.append(c[i]/b[i])
                        elif i!=0:
```

```
                        cs.append(c[i]/(b[i]-a[i-1]*cs[i-1]))

        for i in range(len(d)):
                if i==0:
                        ds.append(d[i]/b[i])
                elif i!=0:
                        ds.append((d[i]-a[i-1]*ds[i-1])/(b[i]-a[i-1]*cs[i-1]))

        for i in range(len(d)-1,-1,-1):
                if i == len(d)-1:
                        u[i]=ds[i]
                elif i!= len(d)-1:
                        u[i]=ds[i]-cs[i]*u[i+1]
        return(u)

n=299
m=0.3/(n-1)
c=(400/3)*m*m
# Matrix equation AX=B, n is the number of nodes
A = [[1, 0, 0,0,0],[1,-(c+2),1,0,0],[0,1,-(c+2),1,0],[0,0,1,-(c+2),1],[0,0,0,2,-(c+2)]]
B=[80,-c*20,-c*20,-c*20,-c*20]
# plot of exact and Numerical


NN=[10,20,40,50,80]
for N in NN:
        l=np.arange(0,0.3,0.3/N)
        plt.plot(l, TDMA(list(A),list(B),N-1),label="N="+str(N))
x = np.arange(0, 0.3, 0.001)
T_exact = 0.058728*np.exp((20*x)/(np.sqrt(3)))+59.9412719*np.exp((-20*x)/(np.sqrt(3)))+2

plt.plot(x, T_exact, label="Exact")
plt.plot(x, TDMA(list(A),list(B),int(n)),label="Numerical")
plt.legend()
plt.xlabel('Distance,x', fontsize=16)
plt.ylabel('Temperature,T', fontsize=16)
plt.show()
```



80
70
60
50
40
30
Temperature,T
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Distance,x
N=10
N=20
N=40
N=50
N=80
Exact
Numerical


```
In [ ]:
```