hw8_exA

April 22, 2020

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
from mpl_toolkits.mplot3d import Axes3D
# relaxa - Program to solve the Laplace equation using
# Jacobi, Gauss-Seidel and SOR methods on a square grid
# modified to record total cpu time
# this version makes an animation as it iterates
# clear all help relaxa  # Clear memory and print header

#* Initialize parameters (system size, grid spacing, etc.)
method = int(input('Numerical Method: 1-Jacobi, 2-Gauss-Seidel, 3-SOR: '))
N = int(input('Enter number of grid points on a side: '))
animate=int(input(' Enter a 1 to animate while iterating:'))
bnd_conds = int(input("Enter 1 for Dirchlet Boundary Conditions, 2 for Neumann␣
↪Boundary Conditions:"))


if bnd_conds == 1:
    L = 1           # System size (length)
    h = L/(N-1)     # Grid spacing
    x = np.arange(0,N)*h   # x coordinate
    y = np.arange(0,N)*h # y coordinate
    yy,xx = np.meshgrid(x,y) # for plotting, note the reversal in x and y
    plot_interval =50 # interval to plot animation, setting it smaller slows␣
↪the program down alot

    #* Select over-relaxation factor (SOR only)
    if( method == 3 ):
      omegaOpt = 2./(1.+np.sin(np.pi/N))  # Theoretical optimum
      print('Theoretical optimum omega = ',omegaOpt)
      omega = float(input('Enter desired omega: '))

    #* Set initial guess as first term in separation of variables soln.
    phi0 = 1      # Potential at y=L
    # phi = phi0 * 4/(np.pi*np.sinh(np.pi)) * np.outer(np.sin(np.pi*x/L),np.
↪sinh(np.pi*y/L))
```

```
    phi=np.ones((N,N)) # try this to see it evolve better

    #* Set boundary conditions
    # first index is the row and second index is column
    phi[0,:] = 0
    phi[-1,:] = 0
    phi[:,0] = 0
    phi[:,-1] = phi0*np.ones(N)
    print('Potential at y=L equals ',phi0)
    print('Potential is zero on all other boundaries')

    #plt.ion()

    #* Loop until desired fractional change per iteration is obtained
    # start_time=cputime     # Reset the cputime counter
    newphi = np.copy(phi)           # Copy of the solution (used only by Jacobi)
    iterMax = N**2          # Set max to avoid excessively long runs
    changeDesired = 1.0e-4   # Stop when the change is given fraction
    print('Desired fractional change = ',changeDesired)
    change = np.array([])

    for iter in range(0,iterMax):
      changeSum = 0.0

      if( method == 1 ):      ## Jacobi method ##
        for i in range(1,N-1):        # Loop over interior points only
         for j in range(1,N-1):
           newphi[i,j] = .25*(phi[i+1,j]+phi[i-1,j]+ phi[i,j-1]+phi[i,j+1])
           changeSum = changeSum + abs(1-phi[i,j]/newphi[i,j])
        phi = np.copy(newphi)

      elif( method == 2 ):  ## G-S method ##
        for i in range(1,N-1):        # Loop over interior points only
         for j in range(1,N-1):
           newphi = .25*(phi[i+1,j]+phi[i-1,j]+ phi[i,j-1]+phi[i,j+1])
           changeSum = changeSum + abs(1-phi[i,j]/newphi)
           phi[i,j] = newphi

      else:                   ## SOR method ##
        for i in range(1,N-1):        # Loop over interior points only
         for j in range(1,N-1):
           newphi = 0.25*omega*(phi[i+1,j]+phi[i-1,j]+ phi[i,j-1]+phi[i,j+1]) ␣
↪+  (1-omega)*phi[i,j]
           changeSum = changeSum + abs(1-phi[i,j]/newphi)
           phi[i,j] = newphi

      #* Check if fractional change is small enough to halt the iteration
```

```
      change = np.append(change,changeSum/(N-2)**2)
      if( iter%10 < 1 ):
        print('After %d iterations, fractional change = %f'%( iter,change[-1]))

      if( change[-1] < changeDesired ):
        print('Desired accuracy achieved after %d iterations'%iter)
        print('Breaking out of main loop')
        break
    # animate
      if(animate ==1 and iter%plot_interval<1):
          fig = plt.figure(2)   # Clear figure 2 window and bring forward
          plt.clf()
          ax = fig.gca(projection='3d')
          surf = ax.plot_surface(xx, yy, phi, rstride=1, cstride=1, cmap=cm.
↪jet,linewidth=0, antialiased=False)
          ax.set_xlabel('x')
          ax.set_ylabel('y')
          ax.set_zlabel('potential after '+str(iter)+' iterations')
          plt.draw()
          plt.show()
          plt.pause(0.1)

    # total_time = cputime - start_time # get the total cpu time

    #* Plot final estimate of potential as contour and surface plots

    #plt.ioff()

    plt.figure(1);plt.clf()
    contourLevels = np.arange(0,1,0.1) #
    plt.contour(xx,yy,phi,contourLevels)  # Contour plot
    # clabel(cs,contourLabels)  # Add labels to selected contour levels
    plt.xlabel('x')
    plt.ylabel('y')
    plt.title(r'$\Phi(x,y)$')

    fig = plt.figure(2)   # Clear figure 2 window and bring forward
    plt.clf()
    ax = fig.gca(projection='3d')
    surf = ax.plot_surface(xx, yy, phi, rstride=1, cstride=1, cmap=cm.
↪jet,linewidth=0, antialiased=False)
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.set_zlabel('potential after '+str(iter)+' iterations')

    #* Plot the fractional change versus iteration
    plt.figure(3);
```

```
    plt.clf()
    plt.semilogy(change)
    plt.xlabel('Iteration')
    plt.ylabel('Fractional change')
    if method==1:
        title=' Jacobi method'
    elif(method==2):
        title =' Gauss-Seidel method'
    elif(method==3):
        title=' SOR method, $\Omega$ ='+str(omega)

    plt.title(r'Iterations ='+str(iter)+title+' N='+str(N))
    plt.grid(True)

    plt.show()


elif bnd_conds == 2:
    N = N+1 # Add one size to boundary
    L = 1           # System size (length)
    h = L/(N-1)     # Grid spacing
    x = np.arange(0,N)*h   # x coordinate
    y = np.arange(0,N)*h # y coordinate
    yy,xx = np.meshgrid(x,y) # for plotting, note the reversal in x and y
    plot_interval =50 # interval to plot animation, setting it smaller slows␣
 ↪the program down alot



    #* Select over-relaxation factor (SOR only)
    if( method == 3 ):
          omegaOpt = 2./(1.+np.sin(np.pi/N))  # Theoretical optimum
          print('Theoretical optimum omega = ',omegaOpt)
          omega = float(input('Enter desired omega: '))

    #* Set initial guess as first term in separation of variables soln.
    phi0 = 1      # Potential at y=L
    # phi = phi0 * 4/(np.pi*np.sinh(np.pi)) * np.outer(np.sin(np.pi*x/L),np.
 ↪sinh(np.pi*y/L))
    phi=np.ones((N,N)) # try this to see it evolve better

    #* Set boundary conditions
    # first index is the row and second index is column
    phi[0,:] = 0
    phi[-1,:] = 0
    phi[1,:] = phi[0,:] # Set direchlet boundary conds on x=0 and x=1
    phi[-2,:] = phi[-1,:]
```

```
    phi[:,0] = 0
    phi[:,-1] = phi0*np.ones(N)
    print('Potential at y=L equals ',phi0)
    print('Potential is zero on all other boundaries')

    #plt.ion()

    #* Loop until desired fractional change per iteration is obtained
    # start_time=cputime     # Reset the cputime counter
    newphi = np.copy(phi)           # Copy of the solution (used only by Jacobi)
    iterMax = N**2          # Set max to avoid excessively long runs
    changeDesired = 1.0e-4   # Stop when the change is given fraction
    print('Desired fractional change = ',changeDesired)
    change = np.array([])

    for iter in range(0,iterMax):
        changeSum = 0.0

        if( method == 1 ):      ## Jacobi method ##
              for i in range(1,N-1):        # Loop over interior points only
                  for j in range(1,N-1):
                       newphi[i,j] = .25*(phi[i+1,j]+phi[i-1,j]+␣
↪phi[i,j-1]+phi[i,j+1])
                       changeSum = changeSum + abs(1-phi[i,j]/newphi[i,j])
              phi = np.copy(newphi)

        elif( method == 2 ):  ## G-S method ##
              for i in range(1,N-1):        # Loop over interior points only
                   for j in range(1,N-1):
                        newphi = .25*(phi[i+1,j]+phi[i-1,j]+␣
↪phi[i,j-1]+phi[i,j+1])
                        changeSum = changeSum + abs(1-phi[i,j]/newphi)
                        phi[i,j] = newphi

        else:                   ## SOR method ##
              for i in range(1,N-1):        # Loop over interior points only
                   for j in range(1,N-1):
                        newphi = 0.25*omega*(phi[i+1,j]+phi[i-1,j]+␣
↪phi[i,j-1]+phi[i,j+1])  +  (1-omega)*phi[i,j]
                        changeSum = changeSum + abs(1-phi[i,j]/newphi)
                        phi[i,j] = newphi

        #* Check if fractional change is small enough to halt the iteration
        change = np.append(change,changeSum/(N-2)**2)
        if( iter%10 < 1 ):
            print('After %d iterations, fractional change = %f'%(␣
↪iter,change[-1]))
```

```
        if( change[-1] < changeDesired ):
            print('Desired accuracy achieved after %d iterations'%iter)
            print('Breaking out of main loop')
            break
        # animate
        if(animate ==1 and iter%plot_interval<1):
            fig = plt.figure(2)   # Clear figure 2 window and bring forward
            plt.clf()
            ax = fig.gca(projection='3d')
            surf = ax.plot_surface(xx, yy, phi, rstride=1, cstride=1, cmap=cm.
↪jet,linewidth=0, antialiased=False)
            ax.set_xlabel('x')
            ax.set_ylabel('y')
            ax.set_zlabel('potential after '+str(iter)+' iterations')
            plt.draw()
            plt.show()
            plt.pause(0.1)

    # total_time = cputime - start_time # get the total cpu time

    #* Plot final estimate of potential as contour and surface plots

    #plt.ioff()

    plt.figure(1);plt.clf()
    contourLevels = np.arange(0,1,0.1) #
    plt.contour(xx,yy,phi,contourLevels)  # Contour plot
    # clabel(cs,contourLabels)  # Add labels to selected contour levels
    plt.xlabel('x')
    plt.ylabel('y')
    plt.title(r'$\Phi(x,y)$')

    fig = plt.figure(2)   # Clear figure 2 window and bring forward
    plt.clf()
    ax = fig.gca(projection='3d')
    surf = ax.plot_surface(xx, yy, phi, rstride=1, cstride=1, cmap=cm.
↪jet,linewidth=0, antialiased=False)
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.set_zlabel('potential after '+str(iter)+' iterations')

    #* Plot the fractional change versus iteration
    plt.figure(3);
    plt.clf()
    plt.semilogy(change)
    plt.xlabel('Iteration')
```

```
    plt.ylabel('Fractional change')
    if method==1:
        title=' Jacobi method'
    elif(method==2):
        title =' Gauss-Seidel method'
    elif(method==3):
        title=' SOR method, $\Omega$ ='+str(omega)

    plt.title(r'Iterations ='+str(iter)+title+' N='+str(N))
    plt.grid(True)

    plt.show()


else:
    print("Incorrect boundary condition option, exit and try again")
```

```
Numerical Method: 1-Jacobi, 2-Gauss-Seidel, 3-SOR: 1
Enter number of grid points on a side: 50
 Enter a 1 to animate while iterating:0
Enter 1 for Dirchlet Boundary Conditions, 2 for Neumann Boundary Conditions:2
Potential at y=L equals  1
Potential is zero on all other boundaries
Desired fractional change =  0.0001
After 0 iterations, fractional change = 0.061224
After 10 iterations, fractional change = 0.007176
After 20 iterations, fractional change = 0.005667
After 30 iterations, fractional change = 0.004803
After 40 iterations, fractional change = 0.004234
After 50 iterations, fractional change = 0.003823
After 60 iterations, fractional change = 0.003508
After 70 iterations, fractional change = 0.003258
After 80 iterations, fractional change = 0.003052
After 90 iterations, fractional change = 0.002878
After 100 iterations, fractional change = 0.002730
After 110 iterations, fractional change = 0.002601
After 120 iterations, fractional change = 0.002487
After 130 iterations, fractional change = 0.002387
After 140 iterations, fractional change = 0.002296
After 150 iterations, fractional change = 0.002215
After 160 iterations, fractional change = 0.002141
After 170 iterations, fractional change = 0.002073
After 180 iterations, fractional change = 0.002011
After 190 iterations, fractional change = 0.001954
After 200 iterations, fractional change = 0.001901
After 210 iterations, fractional change = 0.001852
After 220 iterations, fractional change = 0.001806
```

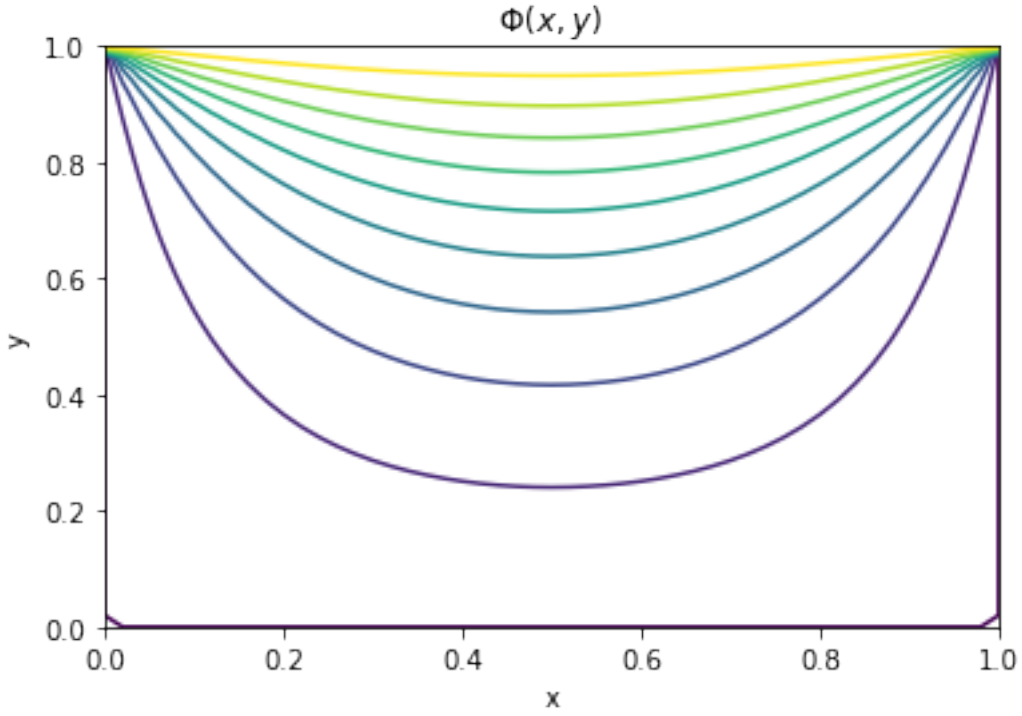
```
After 230 iterations, fractional change = 0.001763
After 240 iterations, fractional change = 0.001724
After 250 iterations, fractional change = 0.001686
After 260 iterations, fractional change = 0.001651
After 270 iterations, fractional change = 0.001618
After 280 iterations, fractional change = 0.001587
After 290 iterations, fractional change = 0.001558
After 300 iterations, fractional change = 0.001530
After 310 iterations, fractional change = 0.001504
After 320 iterations, fractional change = 0.001479
After 330 iterations, fractional change = 0.001456
After 340 iterations, fractional change = 0.001433
After 350 iterations, fractional change = 0.001412
After 360 iterations, fractional change = 0.001391
After 370 iterations, fractional change = 0.001372
After 380 iterations, fractional change = 0.001353
After 390 iterations, fractional change = 0.001335
After 400 iterations, fractional change = 0.001318
After 410 iterations, fractional change = 0.001302
After 420 iterations, fractional change = 0.001286
After 430 iterations, fractional change = 0.001270
After 440 iterations, fractional change = 0.001256
After 450 iterations, fractional change = 0.001241
After 460 iterations, fractional change = 0.001227
After 470 iterations, fractional change = 0.001214
After 480 iterations, fractional change = 0.001201
After 490 iterations, fractional change = 0.001188
After 500 iterations, fractional change = 0.001176
After 510 iterations, fractional change = 0.001163
After 520 iterations, fractional change = 0.001152
After 530 iterations, fractional change = 0.001140
After 540 iterations, fractional change = 0.001129
After 550 iterations, fractional change = 0.001118
After 560 iterations, fractional change = 0.001107
After 570 iterations, fractional change = 0.001096
After 580 iterations, fractional change = 0.001085
After 590 iterations, fractional change = 0.001075
After 600 iterations, fractional change = 0.001065
After 610 iterations, fractional change = 0.001055
After 620 iterations, fractional change = 0.001045
After 630 iterations, fractional change = 0.001035
After 640 iterations, fractional change = 0.001025
After 650 iterations, fractional change = 0.001015
After 660 iterations, fractional change = 0.001006
After 670 iterations, fractional change = 0.000997
After 680 iterations, fractional change = 0.000987
After 690 iterations, fractional change = 0.000978
After 700 iterations, fractional change = 0.000969
```

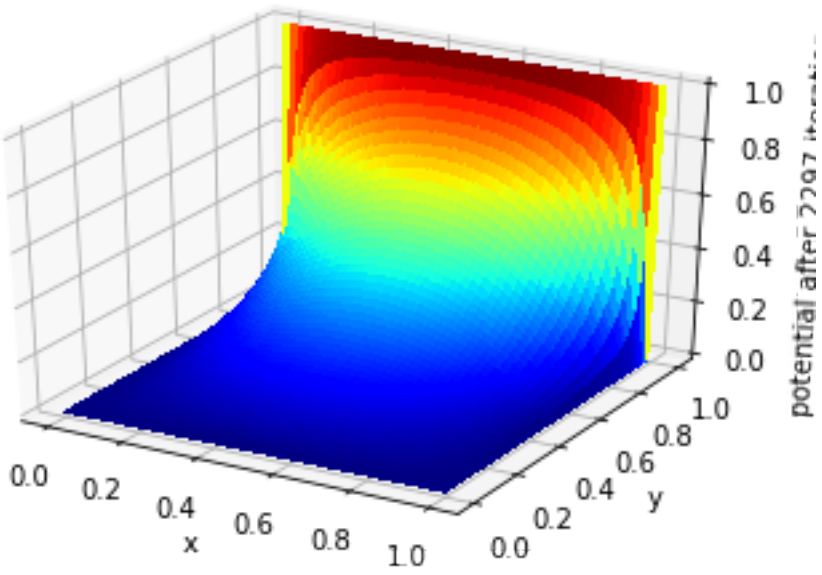
```
After 710 iterations, fractional change = 0.000960
After 720 iterations, fractional change = 0.000951
After 730 iterations, fractional change = 0.000942
After 740 iterations, fractional change = 0.000933
After 750 iterations, fractional change = 0.000924
After 760 iterations, fractional change = 0.000915
After 770 iterations, fractional change = 0.000906
After 780 iterations, fractional change = 0.000897
After 790 iterations, fractional change = 0.000889
After 800 iterations, fractional change = 0.000880
After 810 iterations, fractional change = 0.000872
After 820 iterations, fractional change = 0.000863
After 830 iterations, fractional change = 0.000855
After 840 iterations, fractional change = 0.000846
After 850 iterations, fractional change = 0.000838
After 860 iterations, fractional change = 0.000829
After 870 iterations, fractional change = 0.000821
After 880 iterations, fractional change = 0.000813
After 890 iterations, fractional change = 0.000804
After 900 iterations, fractional change = 0.000796
After 910 iterations, fractional change = 0.000788
After 920 iterations, fractional change = 0.000780
After 930 iterations, fractional change = 0.000772
After 940 iterations, fractional change = 0.000764
After 950 iterations, fractional change = 0.000756
After 960 iterations, fractional change = 0.000748
After 970 iterations, fractional change = 0.000740
After 980 iterations, fractional change = 0.000732
After 990 iterations, fractional change = 0.000724
After 1000 iterations, fractional change = 0.000716
After 1010 iterations, fractional change = 0.000708
After 1020 iterations, fractional change = 0.000700
After 1030 iterations, fractional change = 0.000692
After 1040 iterations, fractional change = 0.000685
After 1050 iterations, fractional change = 0.000677
After 1060 iterations, fractional change = 0.000669
After 1070 iterations, fractional change = 0.000662
After 1080 iterations, fractional change = 0.000654
After 1090 iterations, fractional change = 0.000647
After 1100 iterations, fractional change = 0.000639
After 1110 iterations, fractional change = 0.000632
After 1120 iterations, fractional change = 0.000624
After 1130 iterations, fractional change = 0.000617
After 1140 iterations, fractional change = 0.000610
After 1150 iterations, fractional change = 0.000602
After 1160 iterations, fractional change = 0.000595
After 1170 iterations, fractional change = 0.000588
After 1180 iterations, fractional change = 0.000581
```

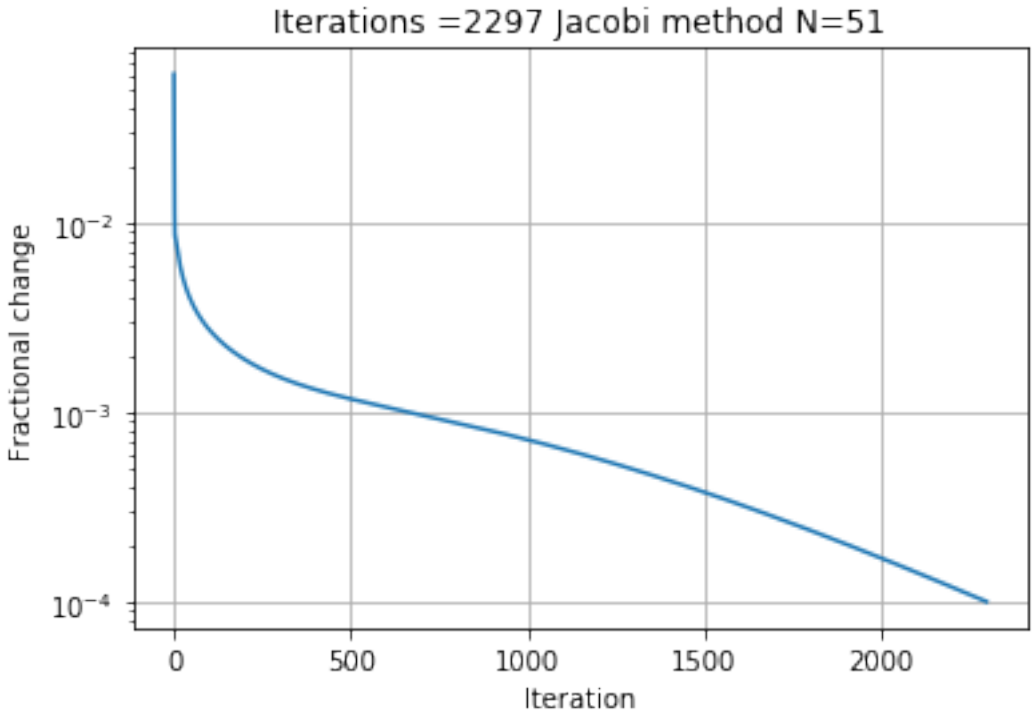
```
After 1190 iterations, fractional change = 0.000574
After 1200 iterations, fractional change = 0.000567
After 1210 iterations, fractional change = 0.000560
After 1220 iterations, fractional change = 0.000553
After 1230 iterations, fractional change = 0.000546
After 1240 iterations, fractional change = 0.000539
After 1250 iterations, fractional change = 0.000532
After 1260 iterations, fractional change = 0.000525
After 1270 iterations, fractional change = 0.000519
After 1280 iterations, fractional change = 0.000512
After 1290 iterations, fractional change = 0.000505
After 1300 iterations, fractional change = 0.000499
After 1310 iterations, fractional change = 0.000492
After 1320 iterations, fractional change = 0.000486
After 1330 iterations, fractional change = 0.000479
After 1340 iterations, fractional change = 0.000473
After 1350 iterations, fractional change = 0.000466
After 1360 iterations, fractional change = 0.000460
After 1370 iterations, fractional change = 0.000454
After 1380 iterations, fractional change = 0.000448
After 1390 iterations, fractional change = 0.000442
After 1400 iterations, fractional change = 0.000436
After 1410 iterations, fractional change = 0.000430
After 1420 iterations, fractional change = 0.000424
After 1430 iterations, fractional change = 0.000418
After 1440 iterations, fractional change = 0.000412
After 1450 iterations, fractional change = 0.000406
After 1460 iterations, fractional change = 0.000400
After 1470 iterations, fractional change = 0.000395
After 1480 iterations, fractional change = 0.000389
After 1490 iterations, fractional change = 0.000383
After 1500 iterations, fractional change = 0.000378
After 1510 iterations, fractional change = 0.000372
After 1520 iterations, fractional change = 0.000367
After 1530 iterations, fractional change = 0.000362
After 1540 iterations, fractional change = 0.000356
After 1550 iterations, fractional change = 0.000351
After 1560 iterations, fractional change = 0.000346
After 1570 iterations, fractional change = 0.000341
After 1580 iterations, fractional change = 0.000336
After 1590 iterations, fractional change = 0.000330
After 1600 iterations, fractional change = 0.000326
After 1610 iterations, fractional change = 0.000321
After 1620 iterations, fractional change = 0.000316
After 1630 iterations, fractional change = 0.000311
After 1640 iterations, fractional change = 0.000306
After 1650 iterations, fractional change = 0.000301
After 1660 iterations, fractional change = 0.000297
```

```
After 1670 iterations, fractional change = 0.000292
After 1680 iterations, fractional change = 0.000288
After 1690 iterations, fractional change = 0.000283
After 1700 iterations, fractional change = 0.000279
After 1710 iterations, fractional change = 0.000274
After 1720 iterations, fractional change = 0.000270
After 1730 iterations, fractional change = 0.000266
After 1740 iterations, fractional change = 0.000262
After 1750 iterations, fractional change = 0.000257
After 1760 iterations, fractional change = 0.000253
After 1770 iterations, fractional change = 0.000249
After 1780 iterations, fractional change = 0.000245
After 1790 iterations, fractional change = 0.000241
After 1800 iterations, fractional change = 0.000237
After 1810 iterations, fractional change = 0.000233
After 1820 iterations, fractional change = 0.000230
After 1830 iterations, fractional change = 0.000226
After 1840 iterations, fractional change = 0.000222
After 1850 iterations, fractional change = 0.000219
After 1860 iterations, fractional change = 0.000215
After 1870 iterations, fractional change = 0.000211
After 1880 iterations, fractional change = 0.000208
After 1890 iterations, fractional change = 0.000204
After 1900 iterations, fractional change = 0.000201
After 1910 iterations, fractional change = 0.000198
After 1920 iterations, fractional change = 0.000194
After 1930 iterations, fractional change = 0.000191
After 1940 iterations, fractional change = 0.000188
After 1950 iterations, fractional change = 0.000185
After 1960 iterations, fractional change = 0.000182
After 1970 iterations, fractional change = 0.000178
After 1980 iterations, fractional change = 0.000175
After 1990 iterations, fractional change = 0.000172
After 2000 iterations, fractional change = 0.000169
After 2010 iterations, fractional change = 0.000167
After 2020 iterations, fractional change = 0.000164
After 2030 iterations, fractional change = 0.000161
After 2040 iterations, fractional change = 0.000158
After 2050 iterations, fractional change = 0.000155
After 2060 iterations, fractional change = 0.000153
After 2070 iterations, fractional change = 0.000150
After 2080 iterations, fractional change = 0.000147
After 2090 iterations, fractional change = 0.000145
After 2100 iterations, fractional change = 0.000142
After 2110 iterations, fractional change = 0.000140
After 2120 iterations, fractional change = 0.000137
After 2130 iterations, fractional change = 0.000135
After 2140 iterations, fractional change = 0.000133
```

```
After 2150 iterations, fractional change = 0.000130
After 2160 iterations, fractional change = 0.000128
After 2170 iterations, fractional change = 0.000126
After 2180 iterations, fractional change = 0.000123
After 2190 iterations, fractional change = 0.000121
After 2200 iterations, fractional change = 0.000119
After 2210 iterations, fractional change = 0.000117
After 2220 iterations, fractional change = 0.000115
After 2230 iterations, fractional change = 0.000113
After 2240 iterations, fractional change = 0.000111
After 2250 iterations, fractional change = 0.000109
After 2260 iterations, fractional change = 0.000107
After 2270 iterations, fractional change = 0.000105
After 2280 iterations, fractional change = 0.000103
After 2290 iterations, fractional change = 0.000101
Desired accuracy achieved after 2297 iterations
Breaking out of main loop
```




1.0
0.8
0.6
0.4
0.2
0.0
potential after 2297 iterati
0.0
0.2
0.4
0.6
0.8
1.0
x
0.0
0.2
0.4
0.6
0.8
1.0
y


Iterations =2297 Jacobi method N=51
Fractional change
10^{-2}
10^{-3}
10^{-4}
0
500
1000
1500
2000
Iteration

[ ]: