دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

Project #1: Solving Linear Systems

Advanced Numerical Analysis

By

Mehras Amirhakimi

Student No.

97126092

Dr B. Baghapour

1 Problem Statement

Consider a steady-state heat diffusion inside a 2D rectangular medium with the prescribed boundary conditions in Fig. 1(a). The governing equation is as follows:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

The temperature distribution for this problem is also depicted in Fig. 1(b). Equation (1) can be discretized using a central-difference scheme inisde the domain. Considering a domain size of (Lx, Ly) = (1.0, 1.0) and grid dimenion of (nx, ny) with a uniform grid size of (Δx, Δy) for each direction i and j, respectively. The algebraic equation for $1 \le i \le nx - 1$ and $1 \le j \le ny - 1$ becomes:

$$a_{i-1.j}T_{i-1.j} + a_{i.j-1}T_{i.j-1} + a_{i.j}T_{i.j} + a_{i+1.j}T_{i+1.j} + a_{i.j+1}T_{i.j+1} = b_{i.j}$$

where the coefficients a’s and b’s are as follows:

$$a_{i-1.j} = a_{i+1.j} = 1.0 \,. \quad a_{i.j-1} = a_{i.j+1} = \left(\frac{\Delta x}{\Delta y}\right)^2$$

$$a_{i.j} = -2.0\left[1.0 + \left(\frac{\Delta x}{\Delta y}\right)^2\right]. \; b_{i.j} = 0$$

Note that in order to reduce the round-of-error, both sides of Eq. (2) was multiplied by $(\Delta x)^2$. To implement Dirchlet and Neumann conditions at boundary nodes, the following relations are used:

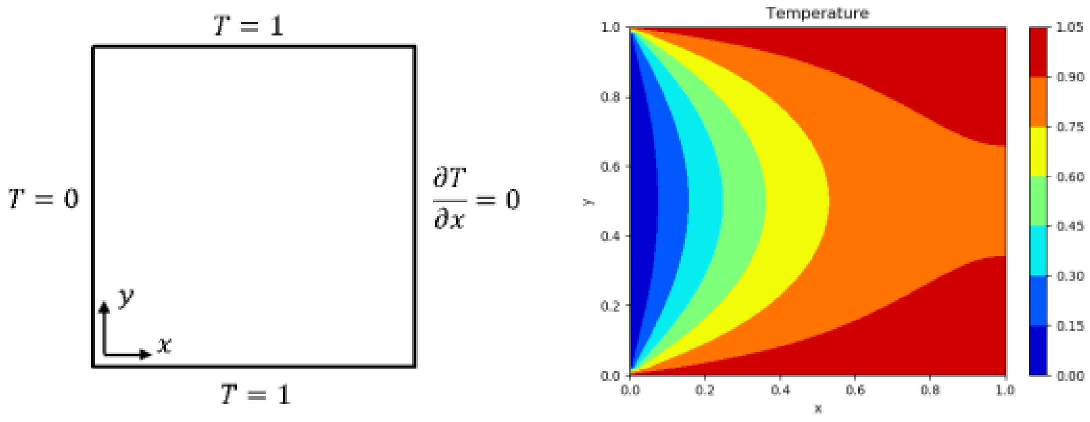
at left side (i=0) : $T_{i.j} = 0.0 \rightarrow a_{i.j} = 1.0. b_{i.j} = 0.0$

at lower side (j=0) : $T_{i.j} = 1.0 \rightarrow a_{i.j} = 1.0. b_{i.j} = 1.0$

at upper side (j=ny) : $T_{i.j} = 1.0 \rightarrow a_{i.j} = 1.0. b_{i.j} = 1.0$

at right side (i=nx) : $T_{i.j} = T_{i-1.j} \rightarrow a_{i-1.j} = -a_{i.j} = 1.0. b_{i.j} = 0.0$

Accordingly, the linear system of equation with size of (N; N), with N = nx × ny, can be formed as AX = B where A = $[A_{m.n}]$ and B = $[B_m]$ ($0 \le m, n \le N - 1$) and X = [Xm] is the solution.



(a) Boundary conditions

b) Temperature distribution

Figure 1: The problem sketch for 2D heat diffusion

2 Methodology

The system of equations described in Sec.1 is going to be solved using sparse matrix format and the uniform mesh with nx = ny = {10,20,50,100} and the residual tolerance of 10^{-5} and $X_0 = 0$. The code has written in Spyder (Python 3.6).

2.1 Constructing A and B for linear solver

In first step we define matrix A in dense form, then we convert it to sparse format using scipy.sparse module's coo_matrix attribute

```
# initialization
import numpy as np
import scipy.sparse as sp
nx = eval(input('x,y grid number: '))
ny = nx
dx = 1/nx
dy = 1/ny
N = (nx+1)*(ny+1)
x = np.linspace(0, 1, nx+1)
y = np.linspace(0, 1, ny+1)

# constructing A and b (coefficient matrix and right hand side vector)
A = np.zeros([N,N])
b = np.zeros(N)
```

```python
for i in range(N):
    if i==0:
        A[i,i]=-4
        A[i,i+1]=1
        A[i,i+nx+1]=1
        b[i]=-1
    elif i==nx:
        A[i,i]=-3
        A[i,i-1]=1
        A[i,2*i+1]=1
        b[i]=-1
    elif i==N-nx-1:
        A[i,i]=-4
        A[i,i-nx-1]=1
        A[i,i+1]=1
        b[i]=-1
    elif i==N-1:
        A[i,i]=-3
        A[i,i-1]=1
        A[i,i-nx-1]=1
        b[i]=-1
    elif i<nx+1:
        A[i,i]=-4
        A[i,i-1]=1
        A[i,i+1]=1
        A[i,i+nx+1]=1
        b[i]=-1
    elif i>N-nx-1:
        A[i,i]=-4
        A[i,i-1]=1
        A[i,i+1]=1
        A[i,i-nx-1]=1
        b[i]=-1
    elif i%(nx+1)==0:
        A[i,i]=-4
        A[i,i-nx-1]=1
        A[i,i+nx+1]=1
        A[i,i+1]=1
    elif (i+1)%(nx+1)==0:
        A[i,i]=-3
        A[i,i-1]=1
        A[i,i-nx-1]=1
        A[i,i+nx+1]=1
    else:
        A[i,i]=-4
        A[i,i-1]=1
        A[i,i+1]=1
        A[i,i-nx-1]=1
        A[i,i+nx+1]=1

Asp=sp.coo_matrix(A)
```

2.2 sketching sparsity matrix for nx = 6

```python
# Sparsity pattern for nx = 6
import matplotlib.pyplot as plt
print(plt.spy(Asp))
```

2.4 Investigate the convergence of SciPy Gmres solver

2.4.1 Without preconditioning:

```
# GMRES (without preconditioning)
import scipy.sparse.linalg as spla
def counter(rk=None):
    counter.niter+=1
    print("# iter {:3d}, residual = {}".format(counter.niter,str(rk)))
x0=np.zeros(N)
counter.niter=0
T = spla.gmres(A,b,x0=x0,callback=counter)
print(T[0])
```

2.4.2 With preconditioning:

```
# GMRES (with preconditioning)
import scipy.sparse.linalg as spla
def counter(rk=None):
    counter.niter+=1
    print("# iter {:3d}, residual = {}".format(counter.niter,str(rk)))
x0=np.zeros(N)
x = spla.gmres(A,b,x0=x0,callback=counter)
M2 = sp.linalg.spilu(Asp,fill_factor=1.0)
Mx = lambda x: M2.solve(x)
M = spla.LinearOperator((N,N),Mx)
counter.niter=0
T = spla.gmres(A,b,M=M,callback=counter,x0=x0)
print(T[0])
```

2.5 Plotting 2D temperature field

```
# Plotting 2D temperature field
T2 = T[0].reshape(ny+1,nx+1)
x,y = np.meshgrid(x,y)
plt.figure(figsize=(8,6))
plt.contourf(x,y,T2,10)
plt.set_cmap('jet')
plt.colorbar()
plt.ylabel('y')
plt.xlabel('x')
plt.show()
```

3 Results and Discussions

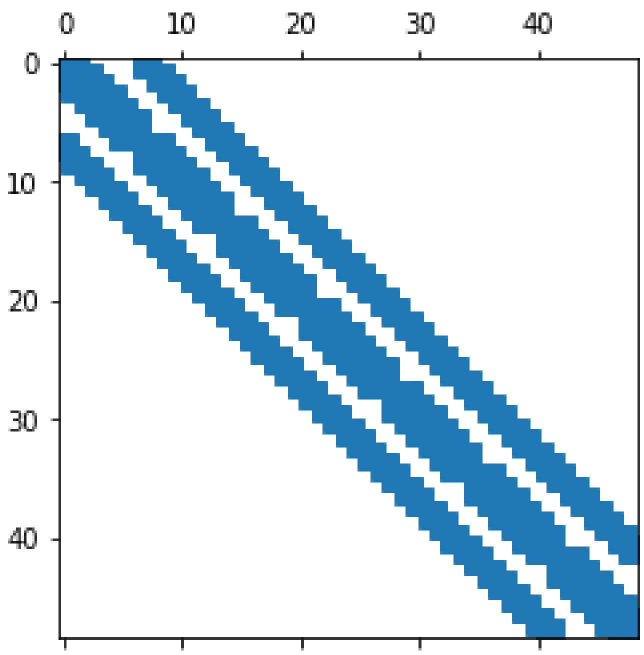
3.1 Matrix A and b

For nx = 3:

```
In [12]: print(A)
[[-4.  1.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 1. -4.  1.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  1. -4.  1.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  1. -3.  0.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 1.  0.  0.  0. -4.  1.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  1.  0.  0.  1. -4.  1.  0.  0.  1.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.  1. -4.  1.  0.  0.  1.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  1.  0.  0.  1. -3.  0.  0.  0.  1.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  1.  0.  0.  0. -4.  1.  0.  0.  1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  1.  0.  0.  1. -4.  1.  0.  0.  1.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  1.  0.  0.  1. -4.  1.  0.  0.  1.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  1. -3.  0.  0.  0.  1.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  0. -4.  1.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  1. -4.  1.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  1. -4.  1.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  1. -3.]]

In [13]: print(b)
[-1. -1. -1. -1.  0.  0.  0.  0.  0.  0.  0.  0. -1. -1. -1. -1.]
```

3.2 Sketching sparsity pattern for nx = 6:



3.4 Investigation of the convergence of SciPy Gmres

Without preconditioning:

nx=10

iter 39, residual = 7.4786858594249985e-06

nx=20

iter 108, residual = 9.704270569453499e-06

nx=50

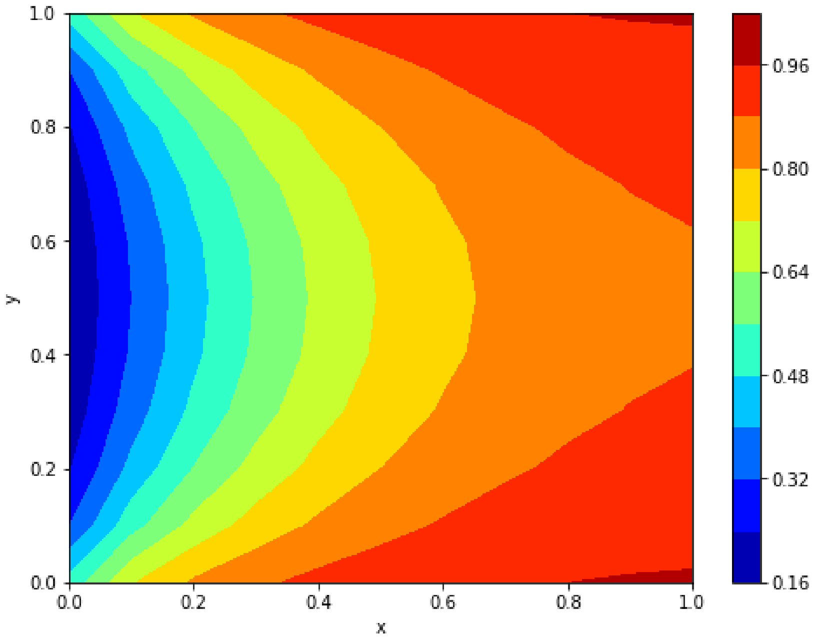
iter 304, residual = 9.839744279751739e-06

nx=100

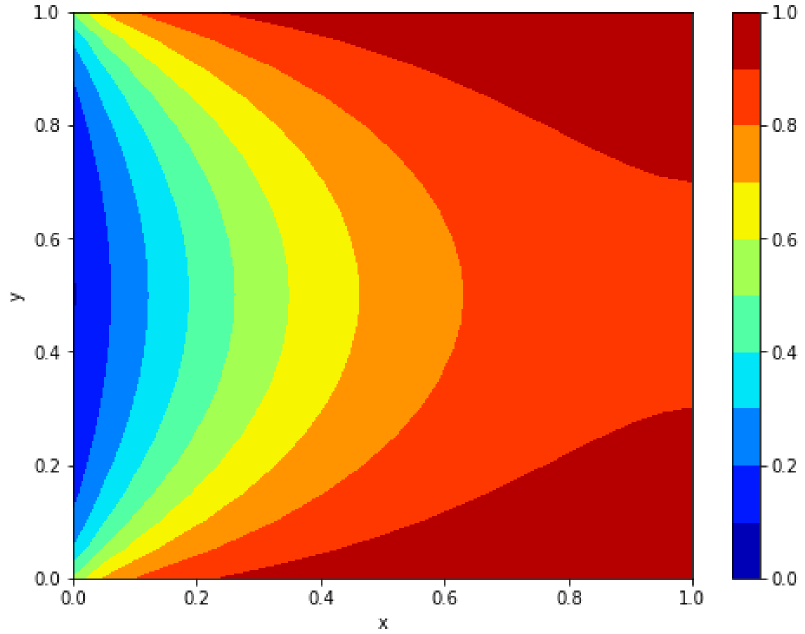
iter 1111, residual = 9.939962381073695e-06

3.6 Plotting 2D temperature field

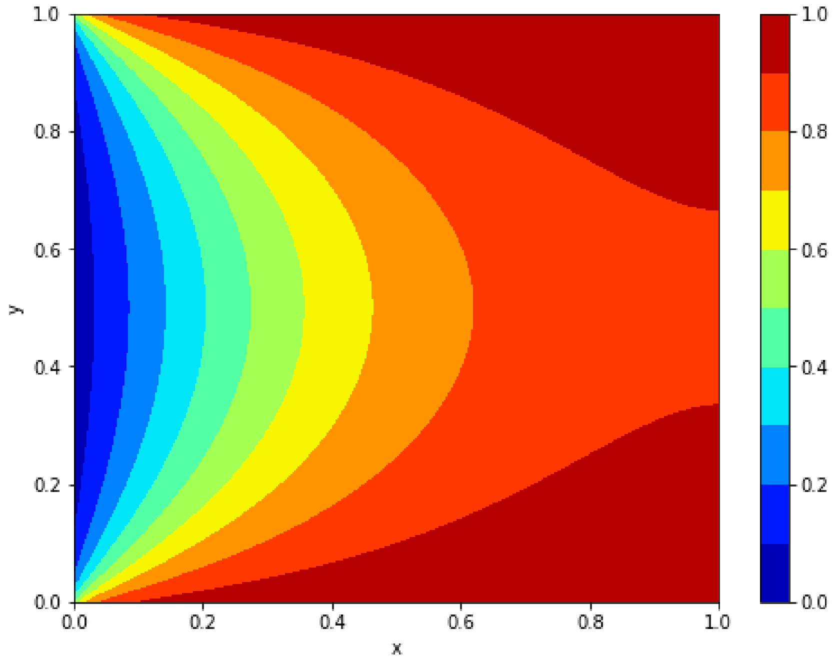
nx=10



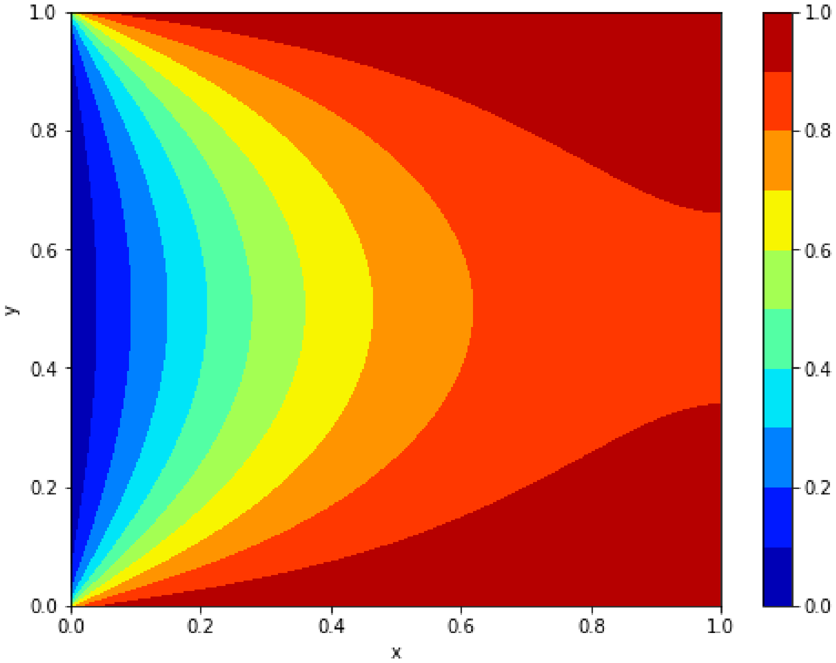
nx=20


1.0
0.8
0.6
0.4
0.2
0.0
y
0.0
0.2
0.4
0.6
0.8
1.0
x
1.0
0.8
0.6
0.4
0.2
0.0

nx=50


1.0
0.8
0.6
0.4
0.2
0.0
y
0.0
0.2
0.4
0.6
0.8
1.0
x
1.0
0.8
0.6
0.4
0.2
0.0

nx=100



By increasing the number of grid points, the temperature field’s contour lines become smoother, temperatures of boundary nodes become closer to prescribed values in problem so solution become more accurate.