AERO97052
Computational Linear Algebra Part I
Coursework Answers
2019–2020

Author: Zijun Fang
Username: zf719
CID: 01811420

Solution to Question 1

(a) For the function derivative:

$$\frac{\partial f}{\partial x} = \frac{f_{tn,xi+\frac{1}{2}} - f_{tn,xi-\frac{1}{2}}}{\Delta x} \qquad \frac{\partial u}{\partial x} = \frac{u_{tn,xi+\frac{1}{2}} - u_{tn,xi-\frac{1}{2}}}{\Delta x}$$

$$\frac{\partial u}{\partial t} = \frac{\partial\left(k\left(\frac{\partial f}{\partial x}\right)\right)}{\partial x} = k\left(\frac{\partial^2 f}{\partial x^2}\right)$$

$$\frac{\Delta\left(\boldsymbol{u^{n+1}} - \boldsymbol{u^{n}}\right)}{\Delta t} - B\left(\frac{-\boldsymbol{u^{n+1}} + \boldsymbol{u^{n}}}{2\Delta x^2}\right) = 0$$

$$\frac{\partial u}{\partial t} - k\left(\frac{\partial^2 f}{\partial x^2}\right) = 0 \qquad \left(\boldsymbol{u^{n+1}} - \boldsymbol{u^{n}}\right) + \left(\boldsymbol{u^{n+1}}\right)\frac{\sigma}{2}B = \left(-\boldsymbol{u^{n}}\right)\frac{\sigma}{2}B$$

$$\left(I + \frac{\sigma}{2}B\right)\boldsymbol{u^{n+1}} = \left(I - \frac{\sigma}{2}B\right)\boldsymbol{u^{n}}$$

For rearranging the previous function, the property:

Which the dimension matrix $\left(I + \frac{\sigma}{2}B\right)$ is marking the function transferring magnitude from the forward condition, due to the system in the material is changed by the distance, the matrix conductivity is the location matrix in B, which the conductivity matrix content inside. For delivering the previous time step heat magnitude to the next section value.

(b) The methods for functions exploit the tridiagonal structure:
In the simulating mainstream script, the finish of the loop appears with the tridiagonal functions to continue the calculations loops to approaching the linear heat transfer magnitude to be worked out through the centred point method. Using the 'Rt_tridiag' function for the output vectors calculations, due to the lower triangular system $R' * \boldsymbol{x} = \boldsymbol{b}$**;** while for the 'Rsolve_tridiag' function for the output results based on the upper triangular system $R * \boldsymbol{x} = \boldsymbol{b}$**.**

(c) With setting the plotting figure situation below:

$$N = 100;$$
$$\Delta t = 10^{-2};$$
$$T = 0, 0.1, 0.4, 1.5;$$

The contour plot result comes with:

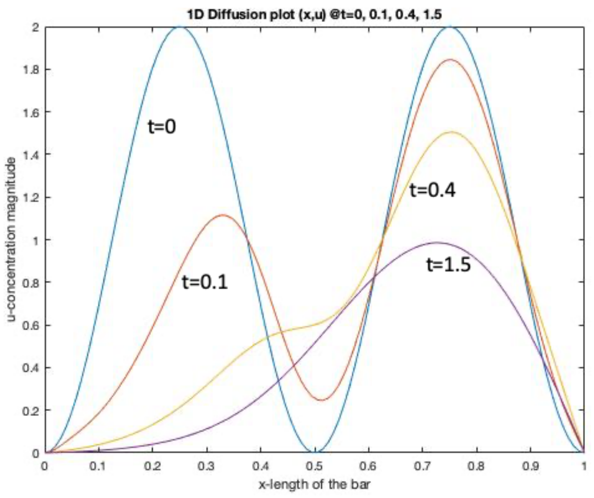


***Figure 1.1**: 1D diffusion matrix plot (x,u) with different time slots*

(d) if the material along the system is both change in time and space, which would not influence the final function output (1.6); due to the second derivatives happen while the PDE function is respected to x, which means that the time variable here could not be differentiated by the function the conductivity also should a constant in one time slot. For the different time slot applies to both sides of the function (1.6):

$$\left(I+\frac{\sigma}{2}*B^{n+1}\right)\boldsymbol{u^{n+1}}=\left(I-\frac{\sigma}{2}*B^{n}\right)\boldsymbol{u^{n}}$$

Solution to Question 2

(a) For the partial pivoting: $PA = LU$

$$\tilde{L}\tilde{U} = A + \Delta A$$

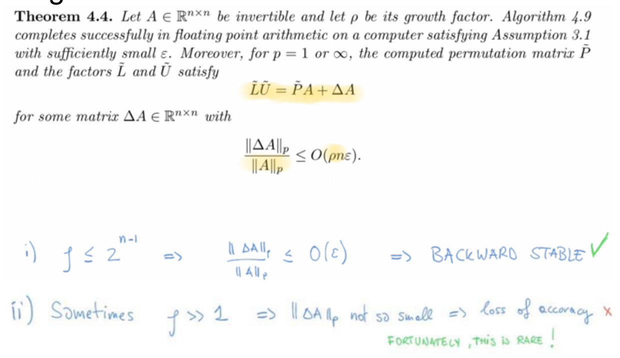
For the output matrix in $\Delta A \in \mathbb{R}^{n*n}$ with:

$$\frac{||\Delta A||p}{||L||p||U||p} \leq o(\varepsilon)$$
$$||\Delta A||p = ||LU||p \leq ||L||p||U||p$$
$$L \leq 1$$

So, for the maximum magnitude of L from p=1 to p=infinity;

$$L_{max} = 1$$

For analysis the constant of growth factor:



Theorem 4.4. *Let $A \in \mathbb{R}^{n\times n}$ be invertible and let ρ be its growth factor. Algorithm 4.9 completes successfully in floating point arithmetic on a computer satisfying Assumption 3.1 with sufficiently small ε. Moreover, for $p = 1$ or ∞, the computed permutation matrix $\tilde{P}$ and the factors $\tilde{L}$ and $\tilde{U}$ satisfy*

$$\tilde{L}\tilde{U} = \tilde{P}A + \Delta A$$

for some matrix $\Delta A \in \mathbb{R}^{n\times n}$ with

$$\frac{\|\Delta A\|_p}{\|A\|_p} \leq O(\rho n \varepsilon).$$

i) $\rho \leq 2^{n-1} \Rightarrow \frac{\|\Delta A\|_p}{\|A\|_p} \leq O(\varepsilon) \Rightarrow$ BACKWARD STABLE ✓

ii) Sometimes $\rho \gg 1 \Rightarrow \|\Delta A\|_p$ not so small $\Rightarrow$ loss of accuracy ✗

FORTUNATELY, THIS IS RARE !

***Figure 2.1**: Prove of Growth Factor*

(b) Here is the fortran output results: n=5

```
A=
  1.00000000     0.00000000     5.20885733E-38  1.43773222E-42  1.00000000
 -1.00000000     1.00000000     1.40129846E-45 -1.38335681E+28  1.00000000
 -1.00000000    -1.00000000     1.00000000     4.59149455E-41  1.00000000
 -1.00000000    -1.00000000    -1.00000000     1.00000000     1.00000000
 -1.00000000    -1.00000000    -1.00000000    -1.00000000     1.00000000
! LU Mehtod
L=
  1.00000000     0.00000000     0.00000000     0.00000000     0.00000000
 -1.00000000     1.00000000     0.00000000     0.00000000     0.00000000
 -1.00000000    -1.00000000     1.00000000     0.00000000     0.00000000
 -1.00000000    -1.00000000    -1.00000000     1.00000000     0.00000000
 -1.00000000    -1.00000000    -1.00000000     1.00000000     1.00000000
 U=
  1.00000000     0.00000000     5.20885733E-38  1.43773222E-42  1.00000000
  0.00000000     1.00000000     5.20885733E-38 -1.38335681E+28  2.00000000
  0.00000000     0.00000000     1.00000000    -1.38335681E+28  4.00000000
  0.00000000     0.00000000     0.00000000    -2.76671363E+28  8.00000000
  0.00000000     0.00000000     0.00000000     0.00000000     0.00000000
 X=
           NaN           NaN           NaN       Infinity       Infinity
 the error of this method is:
           NaN           NaN           NaN           NaN           NaN
! Matrix partial pivoting
L=
   1.00000000      0.00000000      0.00000000      0.00000000      0.00000000
  -1.00000000      1.00000000      0.00000000      0.00000000      0.00000000
  -1.00000000      1.00000000      1.00000000      0.00000000      0.00000000
  -1.00000000     -1.00000000     -0.500000000      1.00000000      0.00000000
  -1.00000000      1.00000000      0.00000000      1.44575854E-28   1.00000000
 U=
   1.00000000      0.00000000      5.20885733E-38   1.43773222E-42   1.00000000
   0.00000000     -1.00000000     -1.00000000      1.00000000      2.00000000
   0.00000000      0.00000000      2.00000000     -1.00000000      0.00000000
   0.00000000      0.00000000      0.00000000     -1.38335681E+28   4.00000000
   0.00000000      0.00000000      0.00000000      0.00000000     -5.78303414E-28
 p=
        1         4         3         2         5
 X=
   4.08207889E+26  -8.16415778E+26 -0.192991555     -0.118034013      -4.08207889E+26
 the error of this method is:
  -1.00000000     -3.68934881E+19  -1.73205078      -2.00000000      -2.23606801

! Complete pivoting
L=
   1.00000000      0.00000000      0.00000000      0.00000000      0.00000000
  -0.00000000      1.00000000      0.00000000      0.00000000      0.00000000
  -1.00000000     -2.00000000      1.00000000      0.00000000      0.00000000
   1.00000000      0.00000000      7.22879268E-29   1.00000000      0.00000000
   1.00000000      0.00000000      1.44575854E-28   7.22879268E-29   1.00000000
 U=
  -1.00000000     -1.00000000      1.00000000     -1.00000000      1.00000000
   0.00000000      1.00000000      1.43773222E-42   5.20885733E-38   1.00000000
   0.00000000      0.00000000     -1.38335681E+28  -1.00000000      4.00000000
   0.00000000      0.00000000      0.00000000      2.00000000     -2.89151707E-28
   0.00000000      0.00000000      0.00000000      0.00000000     -5.78303414E-28
 p=
        4         1         2         3         5
 q=
        2         1         4         3         5
 X=
   4.08207889E+26  -8.16415778E+26 -0.192991614     -0.118034013      -4.08207889E+26
 the error of this method is:
  -1.00000000     -3.68934881E+19  -1.73205078      -2.00000000      -2.23606801
```

(c) None

(d)

Solution to Question 3

(a) With invertible square matrix $A \in \mathbb{R}^{n*n}$
Gauss elimination with partial pivoting: $A^k\boldsymbol{x} = (A * A * \dots\dots * A)\boldsymbol{x} = \boldsymbol{b}$

$$A^0 = A \text{ and } A^k = J_k P_k A^{k-1}$$
$$(J_k P_k \dots\dots J_1 P_1) * A = U$$

ml: lower bandwidth; mu+ml: upper bandwidth
if the matrix size is present as ml and mu+ml;
the nonzero element number after kth matrix A^k: $m_u + ml + k + 1 - i; i = k, \dots\dots, k + ml$
(i: row number)

With all processes after the partial pivoting result:

$$A = (J_k P_k \dots\dots J_1 P_1)^{-1} * U$$
$$A = L * U; A^k = (L * U)^k$$
$$A^k\boldsymbol{x} = (L * U)^k\boldsymbol{x} = L^{k-1} * \boldsymbol{c} = \boldsymbol{b} \qquad flops: o(\boldsymbol{k-1})$$

While replacing the matrix into the specific calculations:

$$U * \boldsymbol{x} = \boldsymbol{c}; \quad \boldsymbol{for\ 'b'\ matrix} \qquad flops\ in\ total:\ o(\boldsymbol{k})$$

(b) With efficiently applying Gauss elimination with partial pivoting:

$$AXA^T = B$$

Which A, B known; X unknown: the solving strategy is for matrix X.

$$L * U = A; A^T = (L * U)^T$$
$$LUXL^T U^T = B$$
$$R^T R = \left(B^{\frac{1}{2}} L^T U^T\right)^T \left(B^{\frac{1}{2}} L^T U^T\right) = X$$

In the order number of matrices is k:

$$R1^T\ R2^T \dots Rk - 2^T Rk - 1^T\ Rk - 1Rk - 2 \dots R2R1 = X \qquad flops: o(\boldsymbol{k})$$

(c) i) Row diagonally dominant:

$$0 < \alpha < 1$$

ii) Strictly row diagonally dominant:

$$|a_{ii}| \geq \sum_{\substack{i=1 \\ j \neq i}}^{n} a_{ij}$$
$$\alpha \leq 1$$

iii) Column diagonally dominant:

$$0 < \alpha < 1$$

iii) Strictly column diagonally dominant:

$$|a_{jj}| \geq \sum_{\substack{j=1 \\ i \neq j}}^{n} a_{ij}$$
$$\alpha \leq 1$$

v) Richard iterative method converge:

$$\varpi^* = \frac{2}{\lambda_{min} + \lambda_{max}} < 1$$
$$\alpha > 0$$

Solution to Question 4

(a) According to the centred finite element approximation in second-order derivatives:

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}}{\Delta x^2} \qquad \frac{\partial^2 T}{\partial y^2} = \frac{T_{i,j-1} - 2T_{i,j} + T_{i,j+1}}{\Delta y^2}$$

The Poisson equation: $q(x, y)$; $T(x, y)$

$$-\frac{\partial^2 T(x,y)}{\partial x^2} - \frac{\partial^2 T(x,y)}{\partial y^2} = q(x,y)$$

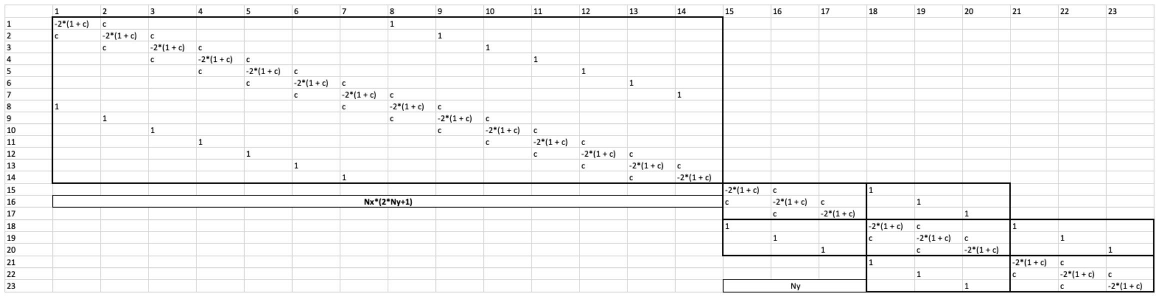
Transfer the Poisson equation into the linear system form: $A\boldsymbol{T} = \boldsymbol{b}$

$$-(T_{i-1,j} - 2T_{i,j} + T_{i+1,j}) - \frac{\Delta x^2}{\Delta y^2}(T_{i,j-1} - 2T_{i,j} + T_{i,j+1}) = \Delta x^2 q(x,y)$$

$$-T_{i-1,j} - T_{i+1,j} - \frac{\Delta x^2}{\Delta y^2}T_{i,j-1} - \frac{\Delta x^2}{\Delta y^2}T_{i,j+1} + 2T_{i,j}\left(1 + \frac{\Delta x^2}{\Delta y^2}\right) = \Delta x^2 q(x,y)$$

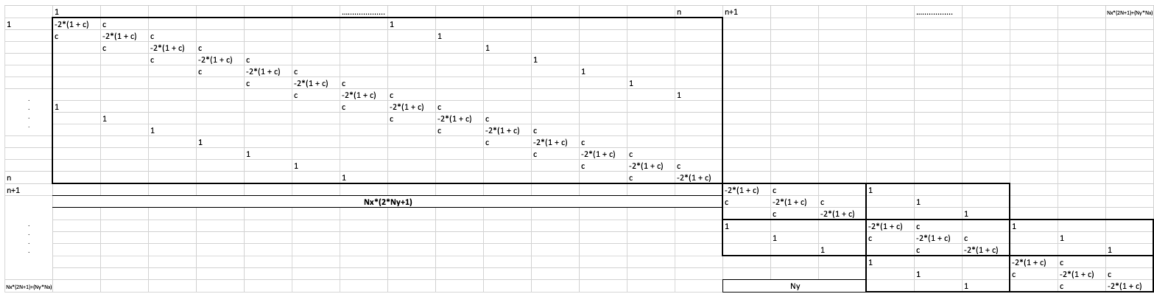
For the elements in Matrix A can be shown as: (for Nx=2 Ny=3)

$$c = \frac{\Delta x^2}{\Delta y^2}$$



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	-2*(1 + c)	c						1															
2	c	-2*(1 + c)	c						1														
3		c	-2*(1 + c)	c						1													
4			c	-2*(1 + c)	c						1												
5				c	-2*(1 + c)	c						1											
6					c	-2*(1 + c)	c						1										
7						c	-2*(1 + c)	c						1									
8	1						c	-2*(1 + c)	c														
9		1						c	-2*(1 + c)	c													
10			1						c	-2*(1 + c)	c												
11				1						c	-2*(1 + c)	c											
12					1						c	-2*(1 + c)	c										
13						1						c	-2*(1 + c)	c									
14							1						c	-2*(1 + c)									
15															-2*(1 + c)	c		1					
16	Nx*(2*Ny+1)														c	-2*(1 + c)	c		1				
17																c	-2*(1 + c)			1			
18															1			-2*(1 + c)	c		1		
19																1		c	-2*(1 + c)	c		1	
20																	1		c	-2*(1 + c)			1
21																		1			-2*(1 + c)	c	
22																			1		c	-2*(1 + c)	c
23															Ny					1		c	-2*(1 + c)

Figure 4.1: *Matrix A sparse structure simulating with Nx=2 Ny=3*



	1						....							n	n+1				....				Nx*(2N+1)+(Ny*Nx)
1	-2*(1 + c)	c						1															
	c	-2*(1 + c)	c						1														
		c	-2*(1 + c)	c						1													
			c	-2*(1 + c)	c						1												
				c	-2*(1 + c)	c						1											
					c	-2*(1 + c)	c						1										
.						c	-2*(1 + c)	c						1									
.	1						c	-2*(1 + c)	c														
.		1						c	-2*(1 + c)	c													
.			1						c	-2*(1 + c)	c												
				1						c	-2*(1 + c)	c											
					1						c	-2*(1 + c)	c										
						1						c	-2*(1 + c)	c									
n							1						c	-2*(1 + c)									
n+1															-2*(1 + c)	c		1					
	Nx*(2*Ny+1)														c	-2*(1 + c)	c		1				
																c	-2*(1 + c)			1			
.															1			-2*(1 + c)	c		1		
.																1		c	-2*(1 + c)	c		1	
.																	1		c	-2*(1 + c)			1
.																		1			-2*(1 + c)	c	
																			1		c	-2*(1 + c)	c
Nx*(2N+1)+(Ny*Nx)															Ny					1		c	-2*(1 + c)

Figure 4.1: *Matrix A sparse structure simulating with Nx and Ny in n-th degree*

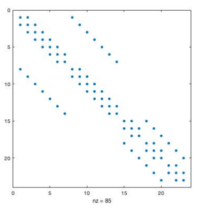


***Figure 4.2**: Matrix A sparse structure in Matlab (using `spy` command)*

(b) For the question in geometry setting: Nx=3; Ny=4
With the `spy` command in the function file: `diffusionMatrix2D.m`

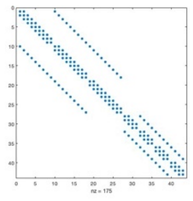


***Figure 4.3**: Matrix A sparse structure in Matlab (using `'spy'` command)*

(c) Write your answer to question 4(c) here.

$$\boldsymbol{T^{k+1}} = D^{-1}\boldsymbol{b} - \left(D^{-1}(L+U)\right)\boldsymbol{T^k}$$

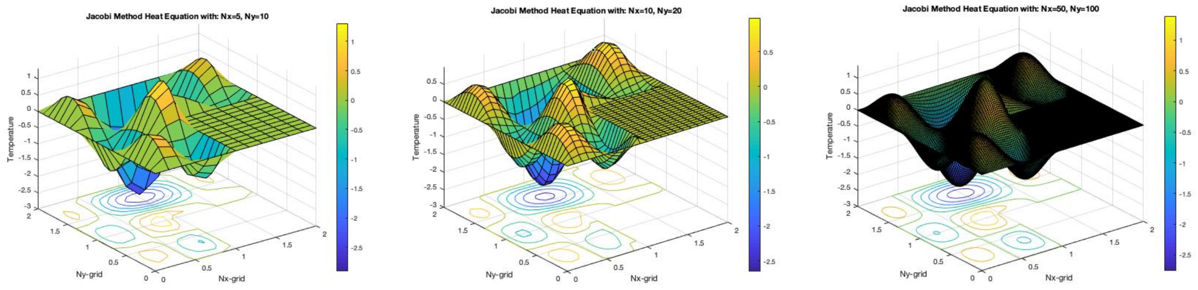


***Figure 4.4-4.6**: Jacobi method simulation figures-plotting in different mesh gird*

(d) Write your answer to question 4(d) here.

* Pivoting Parameter: $\omega = 1.7$;

$$\left(L + \frac{1}{\omega}\right)\boldsymbol{T^{k+1}} = \boldsymbol{b} - \left[U + \left(1 - \frac{1}{\omega}\right)\right]\boldsymbol{T^k}$$

* Minimum Tolerance: 0.0001

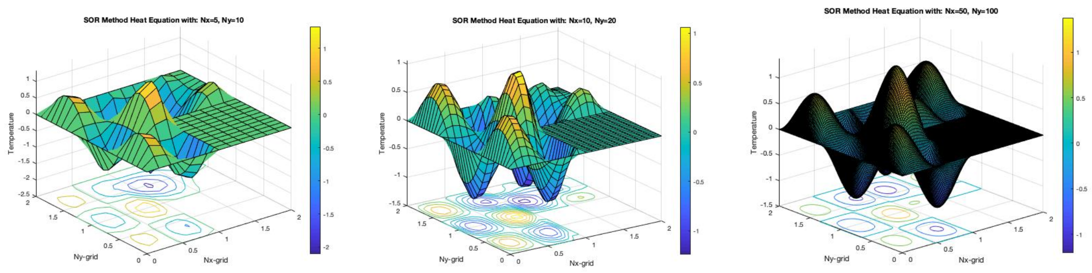


***Figure 4.4-4.6**: Jacobi method simulation figures-plotting in different mesh gird*

* The SOR method could run a high dimension matrix in a very short time;
* With a smoother result generate and larger magnitude in specific point output;
* In weak edge magnitude approach output compare to Jacobian.