

**AERO97052**  
**Computational Linear Algebra Part I**  
**Coursework Answers**  
**2019–2020**

Author: Zijun Fang  
Username: zf719  
CID: 01811420

You **must** use this Word template or its LaTeX version to write your answers. Your BlackBoard submission should include:

1. **One** professionally-typed **PDF** document with your solutions (no hand-writing, no photos/scans of hand-written solutions, no .docx files, no LaTeX source files). You should answer the questions in the same order as they appear on the question sheet. Start the answer to each question on a new page. You **must** write your report using the LaTeX or MS Word templates provided. The LaTeX template has been tested with the online Tex editor [Overleaf](#) (sign in if you have your own account, or sign up using your Imperial email address). Page margins must be **at least 2.5cm on all sides**. The document should be written using **11pt Arial or Helvetica** fonts. The font size can be reduced to 10pt for figure and table captions. **Do not modify the layout of the template!** It is not worth the time, and the marker will see it.

**The answer to each question must not exceed 2 pages (this includes solutions to all sub-questions, all tables, all figures, etc.). Anything beyond this page limit will not be marked.**

2. Your MATLAB and FORTRAN programs. Please submit what is essential, and clearly mark any programs or functions that do not work. Your code should be properly formatted, easy to read, and include good comments to facilitate understanding. You do not need to include your code in the PDF document with your solutions. Instead, you should organize your programs in subfolder called Q1scripts, Q2scripts, ... and include them in your submission.

Your answers and scripts should be collected and submitted in one single zipped folder named **username.zip** (substitute “username” with your username).

**You should write your own individual code and answers to questions. Your programs and answers will be checked against those of other students for similarities.**

## Solution to Question 1

(a) For the function derivative:

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

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

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

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

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

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  $R' * x = b$ ; while for the 'Rsolve\_tridiag' function for the output results based on the upper triangular system  $R * x = b$ .

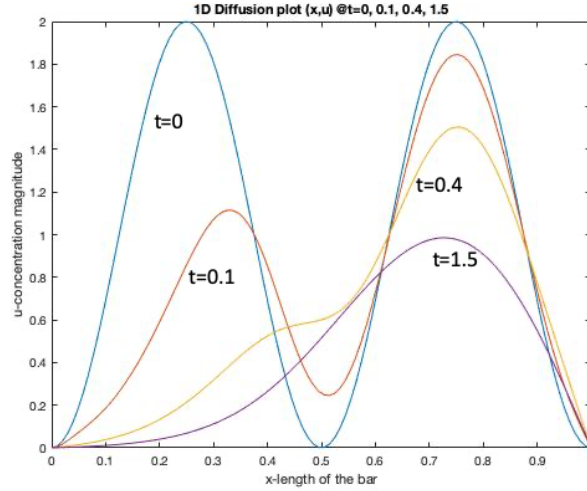
(c) With setting the plotting figure situation below:

$$N = 100;$$

$$\Delta t = 10^{-2};$$

$$T = 0, 0.1, 0.4, 1.5;$$

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):

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

## Solution to Question 2

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

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

For the output matrix in  $\Delta A \in \mathbb{R}^{n \times n}$  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  $\rho$  be its growth factor. Algorithm 4.9 completes successfully in floating point arithmetic on a computer satisfying Assumption 3.1 with sufficiently small  $\varepsilon$ . Moreover, for  $p = 1$  or  $\infty$ , 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

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

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

ii) Sometimes  $f \gg 1 \Rightarrow \|\Delta A\|_p$  not so small  $\Rightarrow$  loss of accuracy  $\times$   
 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	-0.50000000	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  $A \in \mathbb{R}^{n \times n}$

Gauss elimination with partial pivoting:  $A^k \mathbf{x} = (A * A * \dots * A) \mathbf{x} = \mathbf{b}$

$$A^0 = A \text{ and } A^k = J_k P_k A^{k-1}$$

$$(J_k P_k \dots J_1 P_1) * A = U$$

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  $A^k$ :  $m_u + ml + k + 1 - i$ ;  $i = k, \dots, k + ml$   
(i: row number)

With all processes after the partial pivoting result:

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

$$A = L * U; A^k = (L * U)^k$$

$$A^k \mathbf{x} = (L * U)^k \mathbf{x} = L^{k-1} * \mathbf{c} = \mathbf{b} \quad \text{flops: } o(k-1)$$

While replacing the matrix into the specific calculations:

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

(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:

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

(c) i) Row diagonally dominant:

$$0 < \alpha < 1$$

ii) Strictly row diagonally dominant:

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

$$\alpha \leq 1$$

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:

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

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:  $AT = b$

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

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

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

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

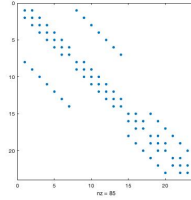
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	$-2^*(1+c)$	c						1														
2	c	$-2^*(1+c)$	c						1													
3		c	$-2^*(1+c)$	c						1												
4			c	$-2^*(1+c)$	c						1											
5				c	$-2^*(1+c)$	c						1										
6					c	$-2^*(1+c)$	c						1									
7						c	$-2^*(1+c)$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

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

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						1							
9		1						c	$-2^*(1+c)$	c						1						
10			1						c	$-2^*(1+c)$	c						1					
11				1						c	$-2^*(1+c)$	c						1				
12					1						c	$-2^*(1+c)$	c						1			
13						1						c	$-2^*(1+c)$	c						1		
14							1						c	$-2^*(1+c)$	c						1	
15								1						c	$-2^*(1+c)$	c						1
16	$Nx*(2*Ny+1)$														$-2^*(1+c)$	c						
17															c	$-2^*(1+c)$	c					
18																c	$-2^*(1+c)$	c				
19																	c	$-2^*(1+c)$	c			
20																		c	$-2^*(1+c)$	c		
21																			c	$-2^*(1+c)$	c	
22																				c	$-2^*(1+c)$	c
23																					c	$-2^*(1+c)$
															$Ny$							

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

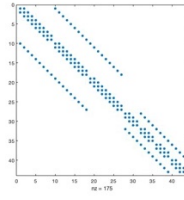




**Figure 4.2:** Matrix A sparse structure in Matlab (using `spy` command)

(b) For the question in geometry setting:  $N_x=3$ ;  $N_y=4$

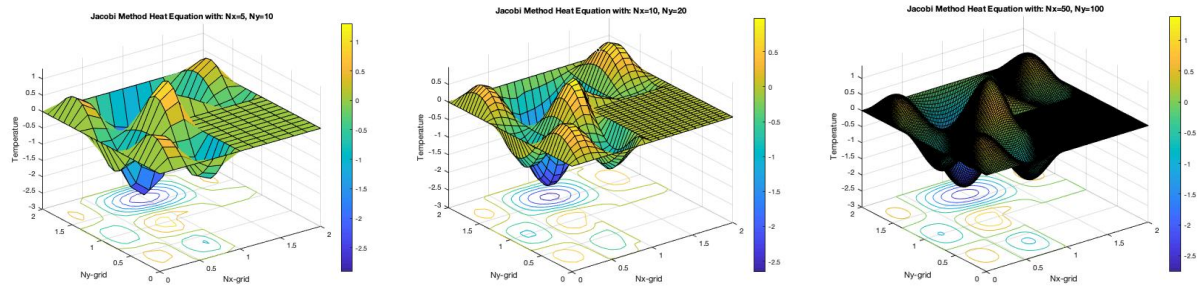
With the `spy` command in the function file: `diffusionMatrix2D.m`



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

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

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



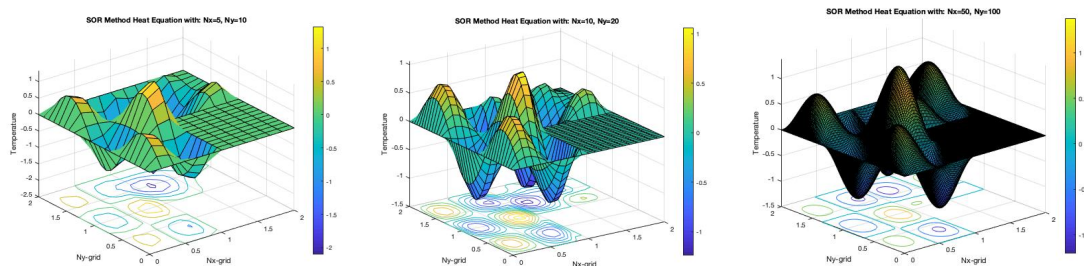
**Figure 4.4-4.6:** Jacobi method simulation figures-plotting in different mesh grid

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

\* Pivoting Parameter:  $\omega = 1.7$ ;

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

\* Minimum Tolerance: 0.0001



**Figure 4.4-4.6:** SOR method simulation figures-plotting in different mesh grid

- \* 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.