Numerical Analysis Assignment #3

Xinglu Wang Student Number: 3140102282
College of Information Science & Electronic Engineering

Problem 1

The following linear system $A\mathbf{x} = \mathbf{b}$ have $\tilde{\mathbf{x}}$ as the actual solution and $\mathbf{x}$ as an approximate solution. Compute $\|\mathbf{x} - \tilde{\mathbf{x}}\|_\infty$ and $\|A\tilde{\mathbf{x}} - \mathbf{b}\|_\infty$.

a). Solution: According to the definition of norm:

$$\begin{aligned} \mathbf{x} &= (0, -7, 5)^t, \\ \tilde{\mathbf{x}} &= (-0.2, -7.5, 5.4)^t \end{aligned}$$

$$\Rightarrow \|\mathbf{x} - \tilde{\mathbf{x}}\|_\infty = \|(0.2, 0.5, -0.4)\|_\infty = max(0.2, 0.5, -0.4) = 0.5 \tag{1}$$

First, applied equation with specific number,

$$\begin{aligned} x_1 + 2x_2 + 3x_3 &= 1 \\ 2x_1 + 3x_2 + 4x_3 &= -1 \\ 3x_1 + 4x_2 + 6x_3 &= 2 \end{aligned}$$

We get

$$A\tilde{\mathbf{x}} - b = \begin{bmatrix} 0 \\ -0.3 \\ -0.2 \end{bmatrix}$$

Then, the norm of residual vector for $\tilde{\mathbf{x}}$ is

$$\|A\tilde{\mathbf{x}} - \mathbf{b}\|_\infty = 0.3 \tag{2}$$

b). Solution: Similarly,

$$\|\mathbf{x} - \tilde{\mathbf{x}}\|_\infty = \|(0.33, 0.9, -0.8)\|_\infty = max(0.33, 0.9, -0.8) = 0.9 \tag{3}$$

and the norm of residual vector for $\tilde{\mathbf{x}}$ is

$$\|A\tilde{\mathbf{x}} - \mathbf{b}\|_\infty = \|(0.27, 0.16, 0.21)\|_\infty = 0.27 \tag{4}$$

Problem 2

Show that if A is symmetric, then $\|A\|_2 = \rho(A)$

Prove: First, we know, if A is a $n \times n$ matrix, then

$$\|A\|_2 = \left[\rho(A^T A)\right]^{\frac{1}{2}} \tag{5}$$

where ρ is the spectral radius defined by $\rho(A) = max\,|\lambda|$.
Therefore, we just need to prove $\rho(A^T A) = \rho(A)^2$
Assume λ is the eigenvalue of A^T, then

$$(AA^T - \lambda^2 I)x = (A^T Ax - A^2 x) = 0 \quad \text{,where } A = A^T$$

thus

$$\rho(A^T A) = max(\lambda^2) = \rho(A)^2 \tag{6}$$

$$\Rightarrow \|A\|_2 = \left[\rho(A^T A)\right]^{\frac{1}{2}} = \rho(A)$$

Problem 3

Implement the algorithm of Gaussian elimination with scaled partial pivoting, and solve the following linear systems.

a. $0.03x_1 + 58.9x_2 = 59.2,$
$5.31x_1 - 6.10x_2 = 47.0.$
Actual solution [10, 1].

b. $3.03x_1 - 12.1x_2 + 14x_3 = -119,$
$-3.03x_1 + 12.1x_2 - 7x_3 = 120,$
$6.11x_1 - 14.2x_2 + 21x_3 = -139.$
Actual solution $[0, 10, \frac{1}{7}]$.

Solution: My solution is shown below, and we can find that there still some roundoff error, although applying scaled partial strategies: My implement of algorithm of Gaussian elimination is shown below. In the process of make lots of mistakes, and find many skills in matlab, for example `[¬,p]` to discard an output argument.

problem a		problem b	
my answer	actual solution	my answer	actual solution
10.000000000000142	10	-0.000000000000004	0
1.000000000000000	1	9.999999999999998	10
		0.142857142857141	1/7

```
format long;
clear
A=[0.03 58.9 ;...
    5.31 -6.10];
b=[59.2;...
    47];
res3_1=GauEli([A,b])
A=[3.03 -12.1 14 -119;-3.03 12.1 -7 120; 6.11 -14.2 21 -139];
res3_2=GauEli(A)
%tt=linsolve(A(:,1:end-1),A(:,end))
```

```
function res = GauEli(A)
%The implement of Guass Elimination
%INPUT: augment matrix A
%OUTPUT: solution x or message about
n=size(A,1);
%ID=fopen('a.txt','w');
for i = 1:n-1
    cmpCol=max(A(:,1:end-1),[],2)./A(:,i);
    cmpCol=abs(cmpCol); %only use i:end of compare column!!
    [¬,p]=max(cmpCol(i:end));
   % fprintf(ID,'%d ',p);
    if p == n+1
        disp('No unique solution'); return;
    else
        tmp = A(i,:);
        A(i,:) = A(p,:);
        A(p,:) = tmp;
    end
    for j = i+1:n
        magn = A(j,i)/A(i,i);
        for k = i+1:n+1
            A(j,k) = A(j,k) - magn*A(i,k);
        end
    end
end
if A(n,n) == 0
    disp('No unique solution'); return;
end
```

```
res=zeros(n,1);
res(n) = A(n,n+1)/A(n,n);
for i = n - 1:-1:1
    sumax = 0;
    for j = i+1:n
        sumax = sumax + A(i,j)*res(j);
    end
    res(i) = (A(i,n+1) - sumax)/A(i,i);
end
```

Problem 4

Implement the Jacobi iterative method and list the first three iteration results when solving the following linear systems, using x(0) = 0.

a.
$$\begin{aligned} 4x_1 + x_2 - x_3 &= 5, \\ -x_1 + 3x_2 + x_3 &= -4, \\ 2x_1 + 2x_2 + 5x_3 &= 1. \end{aligned}$$

b.
$$\begin{aligned} -2x_1 + x_2 + \tfrac{1}{2}x_3 &= 4, \\ x_1 - 2x_2 - \tfrac{1}{2}x_3 &= -4, \\ x_2 + 2x_3 &= 0. \end{aligned}$$

As what is shown below, I make an comparison with the fractional solution, and find the TOL is satisfied, which means iterative method is just an method to approach true value, but can never be accurate. But it is easier and more relaxing to implement compared with Guass method.

My answer with $TOL = 10^-5$	Accurate answer	Numerical true value
1.447762108787455	97/67	1.44776119402985074626865671641791
−0.835818363310185	−56/67	-0.83582089552238805970149253731343
−0.044779294108796	−3/67	-0.04477611940298507462686567164179

Thus, for problem a). my answer is

$$\begin{bmatrix} -1.454543254562683 \\ 1.454543254562683 \\ -0.727274337771892 \end{bmatrix} \tag{7}$$

And for problem b). my answer is

$$\begin{bmatrix} 1.454543254562683 \\ -1.454543254562683 \\ -0.727274337771892 \end{bmatrix} \tag{8}$$

```
%% %% problem 4
clear;
A1 = [4,1,-1;-1,3,1;2,2,5];
b1 = [5;-4;1];
res=JacSol(A1, b1)

%y=sym('y',[3,1]);
%[y1,y2,y3]=solve(A1*y==b1) %symbolic computation. Get fractional solution
%vpa([y1;y2;y3])

A2 = [-2,1,0.5;1,-2,-0.5;0,1,2];
b2 = [4;-4;0];
res=JacSol(A2, b2)
```

```
function [res,time]=JacSol(A,b)
%seperate A into A=D-L-U;
D = diag(diag(A));
U = triu(A,1)*(-1);
```

```
L = tril(A,-1)*(-1);
assert(all(all(A==D-U-L)),'seperate A wrongly!')
T = inv(D)*(L+U);
c = inv(D)*b;
x(:,1)=zeros(size(b));
TOL=10^-5;
tic
for iter=1:10000
    x(:,end+1) = T*x(:,end)+c;
    normInf = norm(A*x(:,end)-b,inf);
    if normInf  < TOL
        break;
    end
end
if(nargout==2)
    time=toc;
end
res=x(:,end);
```

Problem 5

Use the Jacobi method and Gauss-Seidel method to solve the following linear systems, with TOL = 0.001 in the L ∞ norm.

a.
$$\begin{aligned} 3x_1 - x_2 + x_3 &= 1, \\ 3x_1 + 6x_2 + 2x_3 &= 0, \\ 3x_1 + 3x_2 + 7x_3 &= 4. \end{aligned}$$

b.
$$\begin{aligned} 10x_1 - x_2 &= 9, \\ -x_1 + 10x_2 - 2x_3 &= 7, \\ -2x_2 + 10x_3 &= 6. \end{aligned}$$

First, my answer for applying different method and its difference is shown below.

a).

result using JacSol	result using GauSei	difference between two results
0.035087676431390	0.035088637820252	0.000000961388862
-0.236842566509242	-0.236841914529931	0.000000651979311
0.657894221182574	0.657894261447005	0.000000040264431

b).

result using JacSol	result using GauSei	difference between two results
0.995789312500000	0.995789368750000	0.000000056250000
0.957894437500000	0.957894684375000	0.000000246875000
0.791578625000000	0.791578936875000	0.000000311875000

And we can make an comparison between two method in terms of efficiency of algorithm. From what is shown below, we know that for this two specific (and also special, which will be demonstrated soon) matrixes Gauss_Seidel method is more efficient.

Using Jacobi method	it costs 0.481078 ms
Using Gauss_Seidel method	it costs 0.065854 ms

Using Jacobi method	it costs 0.152235 ms
Using Gauss_Seidel method	it costs 0.043190 ms

Then, let us find the unusual of matrix A here. First, matrix A of two problem is both strictly diagonally dominant. Therefore, for any choice of $x_{(0)}$, both the Jacobi and Gauss-Seidel methods can converge to the unique solution of

$Ax = b$.
Second, for the matrix A in problem b)., which is

$$\begin{bmatrix} 10 & -1 & 0 \\ -1 & 10 & -2 \\ 0 & -2 & 10 \end{bmatrix}$$

Gauss-Seidel is no doubt superior to Jacobi. According to Stein-Rosenberg theorem, If $a_{ij}0$, for each $i \neq j$ and $a_{ii} > 0$, for each $i = 1, 2, , n$, and if two method are both converge, we have

$$0 \leq \rho(T_g) < \rho(T_j) < 1$$

which means Gauss-Seidel methods can converge more fast than Jacobi.

```
%% %% problem 5
clear;clc
A1=[3,-1,1;3,6,2;3,3,7];
b1=[1;0;4];
[res1,time1]=JacSol(A1,b1);
[res2,time2]=GauSei(A1,b1);
disp(res1);disp('');disp(res2);
% y=sym('y',[3,1]);
% [y1,y2,y3]=solve(A1*y==b1)
% vpa([y1;y2;y3])

fprintf('Using Jacobi method, it costs %f ms;\nUsing Gauss-Seidel method it costs %f ms ...
    \n\n',time1*1000,time2*1000)

A2=[10,-1,0;-1,10,-2;0,-2,10];
b2=[9;7;6];
[res1,time1]=JacSol(A2,b2);
[res2,time2]=GauSei(A2,b2);
disp(res1);disp('');disp(res2);
fprintf('Using Jacobi method, it costs %f ms;\nUsing Gauss-Seidel method it costs %f ms ...
    \n\n',time1*1000,time2*1000)
```

```
function[res,time]=GauSei(A,b)
D=diag(diag(A));
U = triu(A,1)*(-1);
L = tril(A,-1)*(-1);
assert(all(all(A==D-U-L)),'seperate A wrongly!');
T=inv(D-L)*U;
c=inv(D-L)*b;
x(:,1)=zeros(size(b));
TOL=10^-5;
tic
for iter=1:10000
    x(:,end+1)=T*x(:,end)+c;
    if(norm(A*x(:,end)-b,inf)<TOL)
        break;
    end
end
if(nargout==2)
    time=toc;
end
res=x(:,end);
```