2^ Esercitazione - Secondo Semestre
Analisi Numerica - Calcolo Numerico
A.A. 2014-2015

1. Costruire la fattorizzazione LU delle seguenti matrici:

$$A_1 = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 2 & 3 & 4 & 5 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 4 & 0 & 2 & 4 \\ 0 & 1 & 1 & 1 \\ 2 & 2 & 11 & 9 \\ 4 & 1 & 9 & 25 \end{pmatrix}$$

   Confrontare i risultati ottenuti con l'output della function Matlab `lu`.

2. Studiare la convergenza dei metodi di Jacobi e Gauss-Seidel applicati ai sistemi lineari $A_i x = b_i$, $i = 1, 2$ dove

$$A_1 = \begin{pmatrix} -3 & 3 & -6 \\ -4 & 7 & -8 \\ 5 & 7 & -9 \end{pmatrix}, \quad b_1 = \begin{pmatrix} -6 \\ -5 \\ 3 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 4 & 1 & 1 \\ 2 & -9 & 0 \\ 0 & -8 & -6 \end{pmatrix}, \quad b_2 = \begin{pmatrix} 6 \\ -7 \\ -14 \end{pmatrix},$$

   utilizzando eventualmente il comando `eig` per il calcolo degli autovalori.
   Seguendo come traccia il seguente codice per il metodo di Jacobi,

```
function [xnew,err]=Jacobi(A,b,x0,Nmax,toll)

        D=diag(diag(A));
        B=D-tril(A);
        C=D-triu(A);

        invD=diag(1./diag(A));
        J=invD*(B+C);

        xold=x0;
        i=1;
        test=1;
        err=zeros(1,Nmax);
        while i<Nmax && test>toll
                xnew=J*xold+invD*b;
                err(i)=norm(xnew-xold);
                test=err(i);
                xold=xnew;
                i=i+1;
        end
        err=err(1:i-1);
end
```

   scrivere una function Matlab che implementa il metodo iterativo di Gauss-Seidel (utilizzando un algoritmo opportuno per il calcolo della matrice inversa). Utilizzare i programmi per risolvere i sistemi lineari proposti e visualizzare, in tabella (si utilizzi il comando `fprintf`) e grafico in scala semi-logaritmica, l'andamento dell'errore. Confrontare i risultati con le considerazioni sulla convergenza dei metodi.

```
% Esercizio 1 - Esercitazione 8
diary esercizio_1.txt
clear all
close all
clc
A1 = [1  0  0  1;
     1  0  0  1;
     0  0  1  1;
     2  3  4  5];
A2 = [4  0  2  4;
     0  1  1  1;
     2  2 11  9;
     4  1  9 25];
% Fattorizzazione LU della matrice A1 con nostra funzione
[l1,u1] = Fatt_LU(A1)
% Fattorizzazione LU della matrice A1 con matlab
[L1, U1] = lu(A1)
% Fattorizzazione LU della matrice A2 con nostra funzione
[l2,u2] = Fatt_LU(A2)
% Fattorizzazione LU della matrice A2 con matlab
[L2, U2] = lu(A2)
diary off
```

COMMAND WINDOW

l1 =

0.5000	1.0000	0	1.0000
0.5000	1.0000	0	0
0	0	1.0000	0
1.0000	0	0	0

u1 =

2.0000	3.0000	4.0000	5.0000
0	-1.5000	-2.0000	-1.5000
0	0	1.0000	1.0000
0	0	0	0

l2 =

1.0000	0	0	0
0	0.5000	1.0000	0
0.5000	1.0000	0	0
1.0000	0.5000	-0.5000	1.0000

u2 =

4.0000	0	2.0000	4.0000
0	2.0000	10.0000	7.0000
0	0	-4.0000	-2.5000
0	0	0	16.2500

L1 =

0.5000	1.0000	0	1.0000
0.5000	1.0000	0	0
0	0	1.0000	0
1.0000	0	0	0

U1 =

2.0000	3.0000	4.0000	5.0000
0	-1.5000	-2.0000	-1.5000
0	0	1.0000	1.0000
0	0	0	0

L2 =

1.0000	0	0	0
0	0.5000	1.0000	0
0.5000	1.0000	0	0
1.0000	0.5000	-0.5000	1.0000

U2 =

4.0000	0	2.0000	4.0000
0	2.0000	10.0000	7.0000
0	0	-4.0000	-2.5000
0	0	0	16.2500

FUNZIONE FATTORIZZAZIONE LU

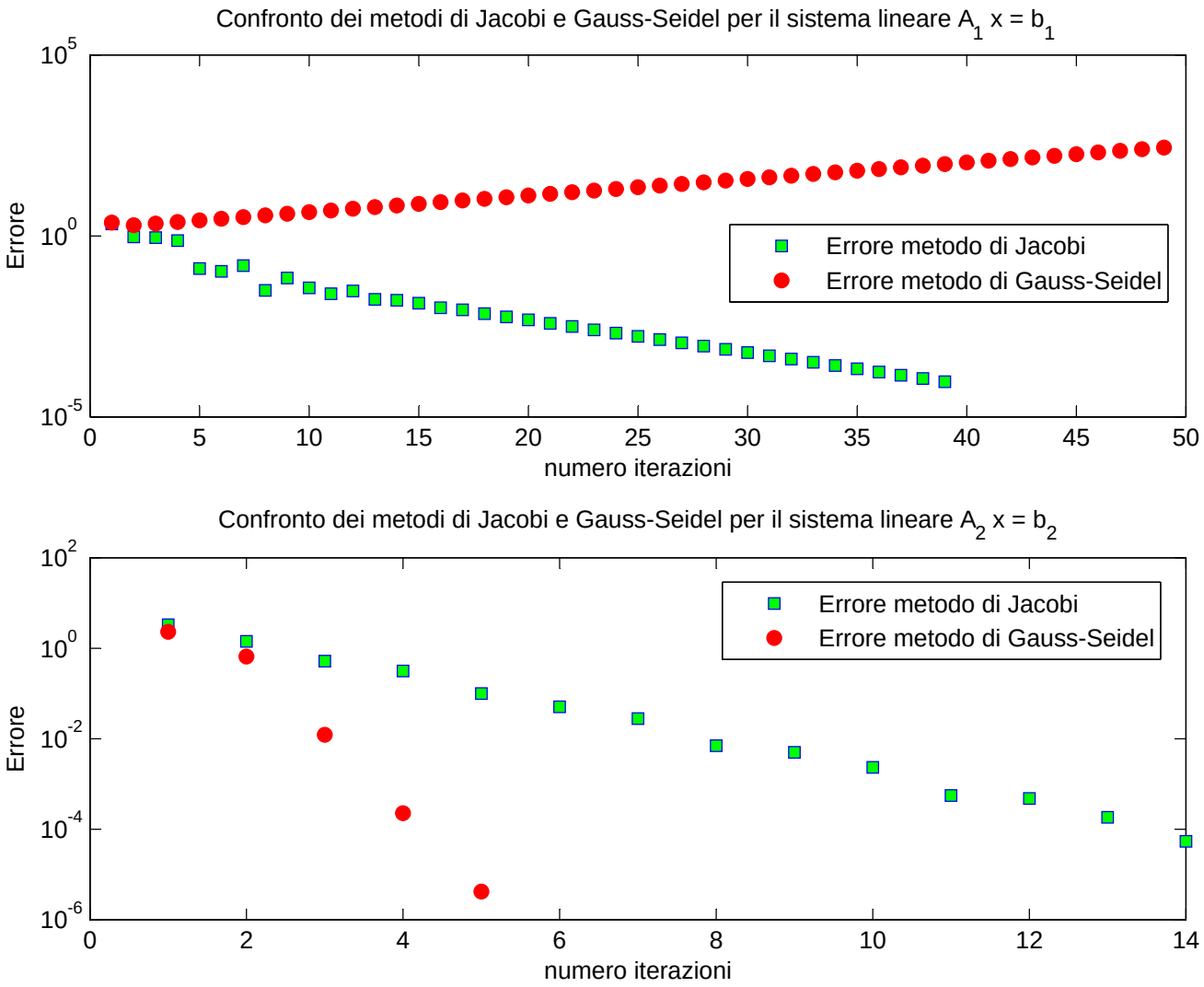
```
function [ L, U ] = Fatt_LU( A )
% Fattorizzazione LU
% ------------INPUT----------------
% A: matrice da fattorizzare
% --------------------------------
   % dimensioni matrice n righe, m colonne
   [n,m] = size(A);
   % iterazioni sulle n righe
   L=zeros(n,m);
   U=A;
   for i = 1:n
      % PIVOTING
      % inizializziamo max con la prima riga i
      max = i;
      % impostiamo max uguale alla riga con il massimo valore , cioe
      % scegliamo la riga del pivot
      for j = i+1 : n
         if (abs(U(j,i)) > abs(U(max,i)))
            max = j;
         end
      end
      % scambio della riga i con la riga max(del pivot)
      L([i max],:) = L([max i],:);
      % memorizziamo il pivot per la permutazione
      pivot(i) = max;
      U([i max],:) = U([max i],:);
      % SCALING
      for j=i+1 :n
         if U(i,i) ~=0
         % Costruice matrice valori coefficienti di gauss
         L(j,i)=-U(j,i)/U(i,i);
         % costruisce matrice riduzione di gauss
         U(j,:)=U(i,:).*L(j,i) + U(j,:);
         end
      end
   end
   % Permutazione per ordinare la matrice L, cioè la matrice P è la
   % permutazione che applico su L
   P = diag(ones(size(U,1),1));
   for i = 1:n-1
      P([i pivot(i)],:) = P([pivot(i) i],:);
   end
   L = L + diag(ones(size(U,1),1));
   L = P\L;
end
```

```
% Esercizio 2 - Esercitazione 8
diary esercizio_2.txt
clear all
close all
clc
% Matrice dei coefficienti A1
A1 = [-3  3  -6; -4  7  -8;5  7  -9];
% Vettore dei termini noti b1
b1 = [-6 -5   3]';
% Matrice dei coefficienti A2
A2 = [ 4  1   1; 2 -9   0;0 -8  -6];
% Vettore dei termini noti b2
b2 = [ 6 -7 -17]';
% vettori soluzione
x=zeros(3,1);
% numero massimo di iterazioni
iter_max=50;
% tolleranza  1*10^-4
toll = 0.0001;

% errore metodo Jacobi per il sistema lineare A_1*x=b_1
disp('il metodo di Jacobi per il sistema lineare A_1*x=b_1 è ');
errJ1 = Jacobi(A1,b1,x,iter_max,toll);
% errore metodo Gauss-Seidel per il sistema lineare A_1*x=b_1
disp('il metodo di Gauss-Seidel per il sistema lineare A_1*x=b_1 è ');
errGS1 = GaussSeidel(A1,b1,x,iter_max,toll);
% errore metodo Jacobi per il sistema lineare A_2*x=b_2
disp('il metodo di Jacobi per il sistema lineare A_2*x=b_2 è ');
errJ2 = Jacobi(A2,b2,x,iter_max,toll);
% errore metodo Gauss-Seidel per il sistema lineare A_2*x=b_2
disp('il metodo di Gauss-Seidel per il sistema lineare A_2*x=b_2 è ');
errGS2 = GaussSeidel(A2,b2,x,iter_max,toll);

% GRAFICI
hold all
subplot(2,1,1);
plot(1:length(errJ1),errJ1 ,1:length(errGS1),errGS1)
subplot(2,1,2);
plot(1:length(errJ2),errJ2 ,1:length(errGS2),errGS2)

% TABELLE ERRORI (con comando matlab fprintf
file_id = fopen('err_Jac_A1.txt','w');
fprintf(file_id,'%3i %12.8f\n',[1:length(errJ1); errJ1]);
fclose(file_id);
file_id = fopen('err_GS_A1.txt','w');
fprintf(file_id,'%3i %12.8f\n',[1:length(errGS1); errGS1]);
fclose(file_id);
file_id = fopen('err_Jac_A2.txt','w');
fprintf(file_id,'%3i %12.8f\n',[1:length(errJ2); errJ2]);
fclose(file_id);
file_id = fopen('err_GS_A2.txt','w');
fprintf(file_id,'%3i %12.8f\n',[1:length(errGS2); errGS2]);
fclose(file_id);
diary off
```

COMMAND WINDOW Esercizio 2 - Esercitazione 8

il metodo di Jacobi per il sistema lineare A_1*x=b_1 è
Convergente
il metodo di Gauss-Seidel per il sistema lineare A_1*x=b_1 è
Non convergente
il metodo di Jacobi per il sistema lineare A_2*x=b_2 è
Convergente
il metodo di Gauss-Seidel per il sistema lineare A_2*x=b_2 è
Convergente

Grafico Esercizio 2 - Esercitazione 8



FUNZIONE GAUSS-SEIDEL

```
function [err, xnew] = GaussSeidel(A, b, x0, iter_max, toll)
% valutiamo l'errore del metodo di Jacobi iterando fino a raggiungere una
% determinata tolleranza(toll).
% ------------INPUT----------------
% iter_max : numero massimo iterazioni
% toll : tolleranza
% --------------------------------
   D = diag(diag(A));
   B = D-tril(A);
   C = D-triu(A);

   Qgs = zeros(size(A,1));
   for i = 1:size(b)
      Qgs(i,i) = 1/A(i,i);
      for j = i+1:size(b)
         Qgs(j,i) = -A(j,:)*Qgs(:,i)/A(j,j);
      end
   end

   GS = Qgs*C;
   % esaminiamo la convergenza del metodo valutando la norma infinita
   % degli autovalori di J
   rag_spet = norm(eig(GS),inf);
   if ( rag_spet < 1 )
      disp('Convergente');
   else
      disp('Non convergente');
   end

   xold = x0;
   i = 1;
   test = 1;
   err = zeros(1, iter_max);
   while i < iter_max && test > toll
      % ricaviamo il vettore soluzione di quella spefica iterazione i
      xnew = GS*xold+Qgs*b;
      err(i) = norm(xnew-xold);
      test = err(i);
      xold = xnew;
      i = i+1;
   end
   err = err(1:i-1);
end
```

FUNZIONE JACOBI

```
function [err, xnew] = Jacobi(A, b, x0, iter_max, toll)
% valutiamo l'errore del metodo di Jacobi iterando fino a raggiungere una
% determinata tolleranza(toll).
% ------------INPUT----------------
% iter_max : numero massimo iterazioni
% toll : tolleranza
% --------------------------------

   D = diag(diag(A));
   B = D-tril(A);
   C = D-triu(A);

   invD = diag(1./diag(A));
   % matrice di iterazione
   J = invD*(B+C);
   % esaminiamo la convergenza del metodo valutando la norma infinita
   % degli autovalori di J
   rag_spet = norm(eig(J),inf);
   if ( rag_spet < 1 )
      disp('Convergente');
   else
      disp('Non convergente');
   end

   xold = x0;
   i = 1;
   test = 1;
   err = zeros(1, iter_max);
   while i < iter_max && test > toll
      % ricaviamo il vettore soluzione di quella spefica iterazione i
      xnew = J*xold+invD*b;
      err(i) = norm(xnew-xold);
      test = err(i);
      xold = xnew;
      i = i+1;
   end
   err = err(1:i-1);
end
```

TABELLE Esercizio 2 - Esercitazione 8

errore metodo Jacobi per il sistema lineare A_1*x=b_1

1	2.14972445
2	0.94414428
3	0.90098166
4	0.74367165
5	0.12498717
6	0.10532058
7	0.15189491
8	0.03129344
9	0.06881143
10	0.03659135
11	0.02514329
12	0.03010791
13	0.01773172
14	0.01667270
15	0.01379642
16	0.01032093
17	0.00904704
18	0.00710360
19	0.00577917
20	0.00477818
21	0.00382212
22	0.00313620
23	0.00254818
24	0.00206588
25	0.00168634
26	0.00136857
27	0.00111355
28	0.00090619
29	0.00073643
30	0.00059925
31	0.00048731
32	0.00039629
33	0.00032236
34	0.00026215
35	0.00021322
36	0.00017341
37	0.00014103
38	0.00011471
39	0.00009329

errore metodo Gauss-Seidel per il sistema lineare A_1*x=b_1

1	2.32771162
2	1.98161260
3	2.18363293
4	2.42089422
5	2.68830372
6	2.98654000
7	3.31824137
8	3.68689477
9	4.09653796
10	4.55170538
11	5.05744941
12	5.61938793
13	6.24376428
14	6.93751584
15	7.70835093
16	8.56483436
17	9.51648263
18	10.57386959
19	11.74874398
20	13.05415998
21	14.50462220
22	16.11624689
23	17.90694099
24	19.89660110
25	22.10733456
26	24.56370506
27	27.29300562
28	30.32556181
29	33.69506867
30	37.43896519
31	41.59885021
32	46.22094468
33	51.35660520
34	57.06289467
35	63.40321630
36	70.44801811
37	78.27557568
38	86.97286186
39	96.63651318
40	107.37390353
41	119.30433726
42	132.56037473
43	147.28930526
44	163.65478362
45	181.83864847
46	202.04294274
47	224.49215860
48	249.43573178
49	277.15081309

errore metodo Jacobi per il sistema lineare A_2*x=b_2

1	3.29889619
2	1.41476664
3	0.51838990
4	0.31477878
5	0.09937825
6	0.05089049
7	0.02791274
8	0.00701968
9	0.00501298
10	0.00232662
11	0.00055683
12	0.00047938
13	0.00018319
14	0.00005378

errore metodo Gauss-Seidel per il sistema lineare A_1*x=b_1

1	2.30479312
2	0.65661589
3	0.01215955
4	0.00022518
5	0.00000417