Metodi Matematici per l'Ingegneria
(A.A. 2019-2020)

Sistemi lineari

Metodi iterativi

Docente: Domenico Vitulano

Email: domenico.vitulano@sbai.uniroma1.it

Ufficio: Via A. Scarpa,

Pal. B, I piano, Stanza n. 11

Tel. 06 49766555

Ricevimento: consultare la pagina web dedicata al corso

Testi consigliati:

Calcolo Numerico, L. Gori, Ed. Kappa, 2006

Esercizi di Calcolo Numerico, L. Gori-M.L. Lo Cascio, F. Pitolli, Ed. Kappa, 2007

Il materiale didattico è disponibile sui siti:

https://www.sbai.uniroma1.it/vitulano-domenico/analisi-numerica/2019-2020

https://elearning.uniroma1.it/enrol/index.php?id=7967

Esercizio

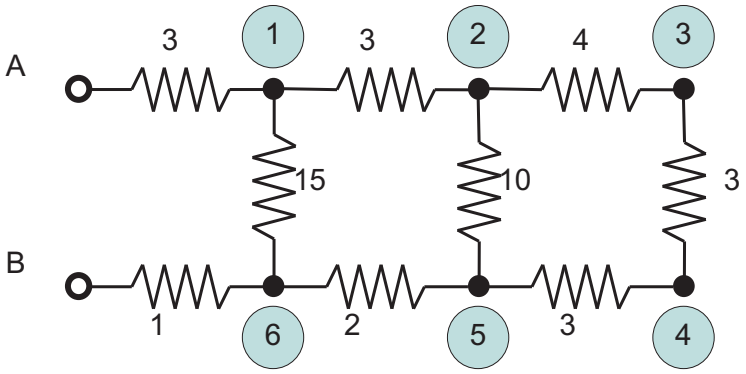
1. Scrivere una funzione Matlab **CS_iterativi** che riceva in input una matrice A e una variabile di tipo char che vale $'J'$ o $'G'$, rispettivamente Jacobi e Gauss-Seidel, e restituisca in output una variabile logica cs che vale 1 se il metodo iterativo scelto soddisfa la condizione sufficiente per la convergenza per ogni scelta della approssimazione iniziale e 0 altrimenti. La funzione deve anche stampare un messaggio di output relativo alla convergenza del metodo.

2. Scrivere una funzione Matlab **CNES_iterativi** che abbia le stesse variabili di input della funzione **CS_iterativi** ma che restituisca in output una variabile logica cnes che vale 1 se il metodo iterativo scelto soddisfa la condizione necessaria e sufficiente per la convergenza e 0 altrimenti.

3. Usare le funzioni precedenti per il sistema relativo al circuito elettrico

Esercizio: Circuito Elettrico

Determinare i potenziali nei nodi $1-6$ del circuito sapendo che tra A e B è applicata una differenza di potenziale pari a 100V (le resistenze sono misurate in Ohm)



Soluzione

Applicando la **legge di Ohm** $\Delta V = RI$ e la **legge di Kirchoff** $\sum_i I_i = 0$ in ogni nodo si ottiene il sistema lineare

$$\begin{cases} 11v_1 - 5v_2 - v_6 = 500 \\ -20v_1 + 41v_2 - 15v_3 - 6v_5 = 0 \\ -3v_2 + 7v_3 - 4v_4 = 0 \\ -v_3 + 2v_4 - v_5 = 0 \\ -3v_2 - 10v_4 + 28v_5 - 15v_6 = 0 \\ -2v_1 - 15v_5 + 47v_6 = 0 \end{cases}$$

Soluzione

```
function [cs] = CS_iterativi(A,tipo)
% verifica le condizioni sufficienti per la  convergenza dei metodi di
% Jacobi e Gauss-Seidel per la soluzione di sistemi aventi A come matrice
% dei coefficienti
%
% INPUT
% A = matrice dei coefficienti del sistema
% tipo = variabile char. Se tipo = 'J', si applica il metodo di
%        Jacobi. Se tipo = 'G', si applica il metodo di Gauss-Seidel
%
% OUTPUT
% cs = variabile logica. Se cs = 1, le condizioni sufficienti sono
%      verificate. Se cs  = 0, le condizioni sufficienti non
%      sono verificate
%
%
```

```
if tipo == 'J'
   Minv = inv(diag(diag(A)));  % Minv = diag(1./diag(A))
   L = tril(A,-1);
   U = triu(A,1);
   CJ = -Minv*(L+U);
   norma_1 = norm(CJ,1);  % norma_1 = max(sum(abs(CJ)));
   norma_inf = norm(CJ,inf); % norma_inf = max(sum(abs(CJ)'));
   test = 0;
   if norma_1 < 1
     fprintf('il metodo di Jacobi converge rispetto alla norma 1 \n')
     cs = 1;
     test = 1;
   end
   if norma_inf < 1
     fprintf('il metodo di Jacobi converge rispetto alla norma infinito')
     cs = 1;
     test = 1;
   end
```

```
if test  == 0
  cs = 0;
  fprintf('la matrice di iterazione del metodo di Jacobi non...
   soddisfa la condizione sufficiente')
end
```

```
elseif tipo == 'G'
   L = tril(A);
   U = triu(A,1);
   Minv = inv(L);
   CGS = -Minv*U;
   norma_1 = norm(CGS,1);
   norma_inf = norm(CGS,inf);
   test = 0;
   if norma_1 < 1
    fprintf('il metodo di Gauss-Seidel converge rispetto alla norma 1\n')
    cs = 1;
    test = 1;
   end
   if norma_inf < 1
    fprintf('il metodo di Gauss-Seidel converge rispetto alla...
     norma infinito')
    cs = 1;
    test = 1;
   end
```

```
   if test == 0
     cs = 0;
     fprintf('la matrice di iterazione del metodo di Gauss-Seidel...
      non soddisfa la condizione sufficiente')
   end
else
 fprintf('la variabile tipo non corrisponde ai metodi...
  di Jacobi o Gauss-Seidel')
    cs = [];
end
```

```
function [cnes] = CNES_iterativi(A,tipo)
% verifica la condizione necessaria e  sufficiente per
% la  convergenza dei metodi di Jacobi e Gauss-Seidel
% per la soluzione di sistemi aventi A come matrice
% dei coefficienti
%
% INPUT
% A = matrice dei coefficienti del sistema
% tipo = variabile char. Se tipo = 'J', si applica il metodo
         di Jacobi. Se tipo = 'G',
%        si applica il metodo di Gauss-Seidel
%
% OUTPUT
% cnes = variabile logica. Se cnes = 1, la condizione
%        necessaria e sufficiente e' verificata.
%        Se cnes  = 0, la condizione necessaria e
%        sufficiente non e' verificata
%
%
%
```

```
if tipo == 'J'
   Minv = inv(diag(diag(A)));
   L = tril(A,-1);
   U = triu(A,1);
   CJ = -Minv*(L+U);
   raggio_spettrale = max(abs(eig(CJ)));
   if raggio_spettrale < 1
       fprintf('il metodo di Jacobi converge')
       cnes = 1;
   else
       cnes = 0;
       fprintf('il metodo di Jacobi non converge')
   end
```

```
elseif tipo == 'G'
   L = tril(A);
   U = triu(A,1);
   Minv = inv(L);
   CGS = -Minv*U;
   raggio_spettrale = max(abs(eig(CGS)));
   if raggio_spettrale < 1
       fprintf('il metodo di Gauss-Seidel converge')
       cnes = 1;
   else
       cnes = 0;
       fprintf('il metodo di Gauss-Seidel non converge')
   end
else
    fprintf('la variabile tipo non corrisponde ai metodi...
     di Jacobi o Gauss-Seidel')
    cnes = [];
end
```

La matrice dei coefficienti del sistema relativo al circuito è

$$A = \begin{pmatrix} 11 & -5 & 0 & 0 & 0 & -1 \\ -20 & 41 & -15 & 0 & -6 & 0 \\ 0 & -3 & 7 & -4 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & -3 & 0 & -10 & 28 & -15 \\ -2 & 0 & 0 & 0 & -15 & 47 \end{pmatrix}$$

Si osserva che la matrice non è a diagonale dominante (controllare seconda e terza riga oppure terza e quarta colonna).

Inoltre,

$$C_J = \begin{pmatrix} 0 & \frac{5}{11} & 0 & 0 & 0 & \frac{1}{11} \\ \frac{20}{41} & 0 & \frac{15}{41} & 0 & \frac{6}{41} & 0 \\ 0 & \frac{3}{7} & 0 & \frac{4}{7} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{3}{28} & 0 & \frac{10}{28} & 0 & \frac{15}{28} \\ \frac{2}{47} & 0 & 0 & 0 & \frac{15}{47} & 0 \end{pmatrix}$$

e quindi

$$\|C_J\|_\infty = \max\{\frac{6}{11}, 1, 1, 1, 1, \frac{17}{47}\} = 1$$

$$\|C_J\|_1 = \max\{\frac{1021}{1927}, \frac{2135}{2156}, \frac{71}{82}, \frac{26}{28}, \frac{3721}{3854}, \frac{193}{308}\} < 1$$

Dal Command Window

```
>> A= [11 -5 0 0 0 -1; -20 41 -15 0 -6 0;...
 0 -3 7 -4 0 0; 0 0 -1 2 -1 0; 0 -3 0 -10 28 -15; -2 0 0 0 -15 47];
>>
>> [cs] = CS_iterativi(A,'J');
il metodo di Jacobi converge rispetto alla norma 1
>>
>> [cs] = CS_iterativi(A,'GS');
il metodo di Gauss-Seidel converge rispetto alla norma infinito
>>
>> [cnes] = CNES_iterativi(A,'J');
il metodo di Jacobi converge
>>
>> [cnes] = CNES_iterativi(A,'GS');
il metodo di Gauss-Seidel converge
>>
```

Esercizio

- Scrivere una funzione Matlab **jacobi.m** che implementi il metodo iterativo di Jacobi.

  La funzione, oltre alla matrice $\mathbf{A}$, al termine noto $\mathbf{b}$ del sistema e al vettore approssimazione iniziale $X0$, deve ricevere in input la precisione ε richiesta alla soluzione e il numero massimo di iterazioni consentite max_iter.

  Le variabili di output della funzione sono l'approssimazione della soluzione prodotta X, l'errore massimo ad ogni iterazione ERR e il numero di iterazioni effettuate $iter$.

  La funzione deve stampare l'errore ad ogni iterazione e un messaggio relativo alla convergenza del metodo.

- Scrivere una funzione Matlab **gauss_seidel.m** che implementi il metodo di Gauss-Seidel.

Le variabili di input e di output sono le stesse della funzione **jacobi**.

- Si consideri il sistema del circuito elettrico e lo si risolva con il metodo di Jacobi e quello di Gauss Seidel usando il vettore:

  $\mathbf{X}^{(0)} = [100 \quad 100 \quad 100 \quad 100 \quad 100 \quad 100]^T$

  e richiedendo una precisione non inferiore a 10^{-6} da raggiungere in meno di 200 iterazioni.

  Si confrontino i risultati con la soluzione data dal solutore di Matlab.

Soluzione

```
function [X, ERR, iter] = jacobi(A,b,X0,eps,max_iter)
% function [X] = jacobi(A,b,X0,eps,max_iter)
% Risolve un sistema lineare con il metodo di Jacobi
%
% Input:
% A = matrice dei coefficienti del sistema
% B = vettore dei termini noti
% X0 = vettore dell'approssimazione iniziale
% eps = accuratezza della soluzione
% max_iter = numero massimo di iterazioni consentite
%
% Output:
% X = vettore soluzione
% ERR = vettore dell'errore massimo commesso ad ogni iterazione
% iter = numero di iterazioni eseguite
```

```
% Calcolo delle dimensioni della matrice
dimA = size(A);
n = dimA(1);

% Costruzione matrice di iterazione
D = diag(diag(A));
L = tril(A,-1);
U = triu(A,1);
Minv = inv(D);
CJ = -Minv*(L+U);
QJ = Minv*b;
```

```
% Calcolo autovalori e verifica C.N.S. di convergenza
rhoCJ = max(abs(eig(CJ)))
if (rhoCJ >= 1)
    error(Attenzione: ==>> rho > 1, il metodo non converge)
    return
end
```

```
% Ciclo iterativo
err = eps + 1;
iter = 0;
ERR = [];
tic
while (err>eps & iter<= max_iter)
    X = CJ*X0+QJ;
    err = norm(X-X0,inf);
    ERR = [ERR err];
    X0 = X;
    iter = iter + 1;
end
toc
disp('ERRORE')
fprintf('%18.15f\n',ERR)
if (iter > max_iter) & (err>eps)
    fprintf('Il metodo non ha raggiunto l''accuratezza richiesta ...
    dopo %7d iterazioni',max_iter)
end
```

```
function [X, ERR, iter] = gauss_seidel(A,b,X0,eps,max_iter)
% function [X] = gauss_seidel(A,b,X0,eps,max_iter)
% Risolve un sistema lineare con il metodo di Gauss Seidel
%
% Input:
% A = matrice dei coefficienti del sistema
% B = vettore dei termini noti
% X0 = vettore dell'approssimazione iniziale
% eps = accuratezza della soluzione
% max_iter = numero massimo di iterazioni consentite
%
% Output:
% X = vettore soluzione
% ERR = vettore dell'errore massimo commesso ad ogni iterazione
% iter = numero di iterazioni eseguite
```

```
% Calcolo delle dimensioni della matrice
dimA = size(A);
n = dimA(1);
X0 = X0(:)';

% Costruzione matrice di iterazione
L = tril(A,);
Minv = inv(L);
U = triu(A,1);
C_GS = -Minv*(U);

% Calcolo autovalori e verifica C.N.S. di convergenza
rhoCGS = max(abs(eig(C_GS)))
if (rhoCGS >= 1)
    error('Attenzione: ==>> rho > 1, il metodo non converge')
    return
end
```

```
% Ciclo iterativo
err = eps + 1;, iter = 1;, ERR = [];
tic
while (err>eps & iter<= max_iter)
 for i = 1:n
  X(i)=(-sum(A(i,1:i-1).*X(1:i-1)-...
        A(i,i+1:end).*X0(i+1:end))+b(i))/A(i,i);
 end
 err = norm(X-X0,inf);
 ERR = [ERR err];
 X0 = X;
 iter = iter + 1;
end,
toc
X=X';
fprintf('%18.15f\n',ERR)
if (iter > max_iter) & (err>eps)
    fprintf('Il metodo non ha raggiunto l''accuratezza richiesta...
     dopo %7d iterazioni',max_iter)
end
```

```
%%%%%  S C R I P T

A = [ 11  -5   0   0   0    -1  ;
     -20  41 -15   0  -6    0  ;
       0  -3   7  -4   0    0  ;
       0   0  -1   2   -1   0  ;
       0  -3   0 -10  28  -15  ;
      -2   0   0   0  -15  47];

B = [500 0 0 0 0 0]';

format long

XM = A \ B    %Soluzione Matlab

XI = [100 100 100 100 100 100]; %Vettore iniziale

[XJ, ERR, iter] = jacobi(A,B,XI',1*10^-6,200);
```

```
ERR

XJ    %Soluzione Jacobi

iter


[XGS,ERR,iter]=gauss_seidel(A,B,[100 100 100 100 100 100]',1*10^-6,200);

ERR

XGS     %Soluzione Gauss-Seidel

iter

sum(abs(XM - XJ))  % confronto matlab J

sum(abs(XM - XGS'))  % confronto matlab GS
```

OUTPUT

```
XM =

   70.000000000000014
   52.000000000000007
   40.000000000000007
   31.000000000000007
   22.000000000000007
   10.000000000000004


% JACOBI

rhoCJ =   0.869179826523406

Elapsed time is 0.002804 seconds.
```

ERRORE

63.829787234042556

34.194528875379945

17.097264437689972

13.127579663275412

13.025891851211298

11.268881057074026

9.595679966128891

8.677345948335528

7.179435727613757

6.579306105739832

5.408413883468675

4.974463046448804

4.082743247229082

3.758795367595567

3.083772724397946

2.839806644645201

2.329584289740090

2.145425080027294

1.759914947567118
1.62081703757881O
1.329564451742257
1.224485427450048
1.00444986969338l
0.925066572037608
0.75883514714613 7
0.698863384278901
0.573279862594305
0.527972823840877
0.433097777905957
0.398869515975818
0.327193923432098
0.301335377813928
0.247186361424415
0.227650913951258
0.186742763075014
0.171984248877457
0.141079222037749

0.12992955462896 6
0.10658162364152 1
0.09815834459299 6
0.08051959979749 8
0.07415603509701 9
0.06083042958115 9
0.05602292463375 1
0.04595578185110 6
0.04232383892170 3
0.03471837861558 7
0.03197454172163 1
0.02622881746632 2
0.02415592121025 9
0.01981517839006 5
0.01824916005352 9
0.01496984357508 1
0.01378675811038 5
0.01130932118051 2
0.01041553137989 1

0.00854389325596 1
0.00786865868372 1
0.00645468554692 0
0.00594456367346 7
0.00487634433874 7
0.00449096074545 7
0.00368394927021 9
0.00339280215084 9
0.00278312630994 6
0.00256317235604 1
0.00210257837146 3
0.00193640897248 3
0.00158844239024 7
0.00146290580104 1
0.00120002624463 2
0.00110518666930 7
0.00090658811212 3
0.00083493931949 6
0.00068490335834 1

0.00063077458913 3
0.00051742638580 5
0.00047653353124 8
0.00039090195932 8
0.00036000848847 1
0.00029531609906 3
0.00027197689831 8
0.00022310350788 1
0.00020547135856 2
0.00016854880376 7
0.00015522818095 1
0.00012733416664 2
0.00011727078816 6
0.00009619759756 4
0.00008859498107 9
0.00007267474253 3
0.00006693116671 4
0.00005490384725 0
0.00005056472751 6

0.000041478405535
0.000038200315259
0.000031335839140
0.000028859328580
0.000023673398296
0.000021802460012
0.000017884626743
0.000016471182320
0.000013511362830
0.000012443542914
0.000010207477519
0.000009400767809
0.000007711479494
0.000007102031624
0.000005825818953
0.000005365397193
0.000004401252244
0.000004053415772
0.000003325029748

```
 0.000003062248489
 0.000002511972099
 0.000002313447808
 0.000001897728538
 0.000001747748669
 0.000001433683749
 0.000001320377933
 0.000001083110181
 0.000000997510625

XJ =

  70.000001323153995
  52.000002729229102
  40.000003081846472
  31.000003346309501
  22.000001974250299
  10.000000908548408

iter = 122
```

```
%Gauss-Seidel

rhoCGS =

   0.75822641924589 2

Elapsed time is 0.086413 seconds.

ERRORE
63.829787234042556
34.714433020460163
18.956346793176806
13.225212848067727
10.564560564124704
 7.887712037302165
 5.903081423560934
 4.441540399950149
 3.359360318290427
 2.546149641813081
```

1.930695165757008
1.464036314384160
1.110117869323339
0.841731151616521
0.638223914832800
0.483917965106976
0.366919191506817
0.278207761060585
0.210944461212655
0.159943662374246
0.121273510875852
0.091952780181479
0.069721027342588
0.052864324924698
0.040083127794446
0.030392086458974
0.023044082888802
0.017472632453462
0.013248211539967

0.010045143997367
0.00761649356393 0
0.00577502664219 1
0.00437877771961
0.00332010499069 9
0.00251739131862 6
0.00190875260536 0
0.00144726665318 7
0.00109735581214 2
0.00083204416807 8
0.00063087787021 5
0.00047834826851 6
0.00036269629480 0
0.00027500591286 2
0.00020851674859 3
0.00015810290763 3
0.00011987780152 4
0.00009089451619 8
0.00006891862355 5

0.00005255921155
0.00003962181997 3
0.00003042310684
0.00002278873657
0.00001727154381 1
0.00001309574081 9
0.00000992953666 7
0.00000752883703 1
0.00000570856315 1
0.00000432838338 3
0.00000328189464 2
0.00000248841922 7
0.00000188678519 2
0.00000143061038 2
0.00000108472658 4
0.00000082246835 1

```
XGS =

   70.000001360646763
   52.000002171174238
   40.000002579344013
   31.000002187840600
   22.000001362030311
   10.000000492590386

iter =

     64
```

```
ans =

     1.336333773238607e-05


ans =

     1.015362626510807e-05

>>
```

FINE

Se v_i sono i potenziali nei nodi e

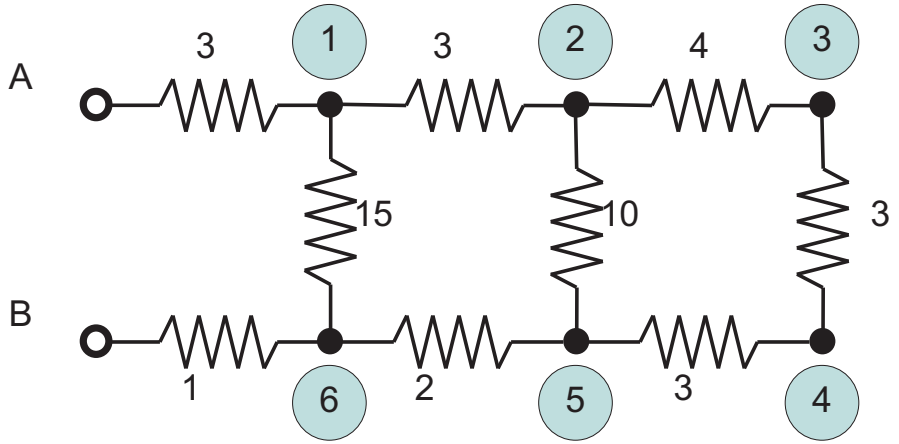
I_{pq} la corrente tra il nodo p e il nodo q (ramo pq)

si ha:

$$I_{pq} = \frac{v_p - v_q}{R_{pq}} \quad \textbf{legge di Ohm}$$

Ogni equazione esprime la **legge delle correnti di Kirchoff**:

La somma delle correnti in ciascun nodo deve essere nulla.



Es. per il primo nodo:

$$I_{A1} + I_{21} + I_{61} = \frac{100 - v_1}{3} + \frac{v_2 - v_1}{3} + \frac{v_6 - v_1}{15} = 0$$

che porta a

$$11v_1 - 5v_2 - v_6 = 500$$