



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Laboratorio di Calcolo Numerico
Laboratorio 11:
Metodi iterativi per la soluzione di sistemi lineari

20 Maggio 2019

Risolvere sistemi di equazioni lineari

Metodi Iterativi

Data una matrice quadrata A di ordine n non singolare e il vettore \mathbf{b} ($\mathbf{b} \in \mathbb{R}^n$), si vuole trovare un'approssimazione del vettore \mathbf{x} ($\mathbf{x} \in \mathbb{R}^n$) tale che

$$A\mathbf{x} = \mathbf{b}$$

con un metodo iterativo.

- Si parte da un'approssimazione iniziale della soluzione del sistema, \mathbf{x}_0 , e, attraverso un procedimento iterativo si ottiene un'approssimazione della soluzione, \mathbf{x} , con un certo grado di accuratezza.

Risolvere sistemi di equazioni lineari

Metodi Iterativi

Dato il sistema lineare

$$A\mathbf{x} = \mathbf{b}$$

e introducendo l'equazione vettoriale

$$\mathbf{f}(\mathbf{x}) = \mathbf{b} - A\mathbf{x}$$

la soluzione di $\mathbf{f}(\mathbf{x})$ viene cercata mediante un metodo di punto fisso. Lo schema di punto fisso che ne consegue è il seguente:

$$\mathbf{x}^{(k+1)} = \mathbf{g}(\mathbf{x}^{(k)})$$

con $\mathbf{x}^{(0)}$ vettore iniziale.

Risolvere sistemi di equazioni lineari

Metodi Iterativi stazionari

La funzione $g(\mathbf{x})$ può essere del tipo:

$$g(\mathbf{x}) = (I - CA)\mathbf{x} + C\mathbf{b}$$

- C è una matrice opportuna che caratterizza il metodo
- Nei metodi stazionari la matrice che moltiplica \mathbf{x} e \mathbf{b} non varia al variare delle iterazioni
- La matrice $E = (I - CA)$ è detta matrice di iterazione del metodo

Risolvere sistemi di equazioni lineari

Metodi Iterativi stazionari

La scelta della matrice C caratterizza la matrice di iterazione E e, quindi, il metodo iterativo. Metodi iterativi stazionari:

- Jacobi

$$E_j = F^{-1}(M + N)$$

- Gauss-Seidel

$$E_s = (F - M)^{-1}N$$

- Rilassamento

$$E_\omega = (F - \omega M)^{-1}((1 - \omega)F + \omega N)$$

Nota: la matrice A è scomposta nella somma di tre matrici $A = F - M - N$ con F matrice diagonale, $-M$ triangolare bassa con diagonale nulla e $-N$ triangolare bassa con diagonale nulla.

Metodo di Jacobi

Introducendo in

$$\mathbf{x}^{(k+1)} = \mathbf{g}(\mathbf{x}^{(k)})$$

la funzione vettoriale

$$\mathbf{g}(\mathbf{x}) = (I - CA)\mathbf{x} + C\mathbf{b}$$

e sapendo che

$$E_j = F^{-1}(M + N)$$

il metodo di Jacobi è:

$$\mathbf{x}^{(k+1)} = E_j \mathbf{x}^{(k)} + F^{-1}\mathbf{b}$$

È possibile implementare il metodo costruendo la matrice di iterazione (implementazione 1) oppure risolvendo componente per componente (implementazione 2).

Metodo di Jacobi

Implementazione 1

$$\mathbf{x}^{(k+1)} = (I - F^{-1}A)\mathbf{x}^{(k)} + F^{-1}\mathbf{b}$$

Implementazione 2

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^n a_{ij}x_j^{(k)} \right) \quad i = 1, \dots, n$$

Criterio di arresto: $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| < toll$ oppure $k > itmax$.

Metodo di Jacobi

SCRIPT - MATLAB

```
clear variables
close all
% Soluzione del sistema lineare Ax=b con metodo di Jacobi
% Input DATA
A = load('matrice.dat');
b = load('tnoto.dat');
x0 = load('xiniziale.dat');
toll = 10(-8);
itmax = 150;
% Apertura file di output
fid=fopen('risul_jacobi.dat','w');
% Chiamata alla function jacob
[xnew, iter, scarto, asint, vel_conv] = jacob2(A,b,x0,itmax,toll);
% Stampa della soluzione
ST = [2:iter; scarto(2:iter)'; asint'; vel_conv'];
fprintf(fid, '%5s %12s %12s %12s \n', 'iter', 'scarto', 'asint', 'R');
fmt=[ '%5d ' repmat('%12.7e ',1,3) '\n'];
fprintf(fid, fmt, ST); fclose(fid);
% Profilo di convergenza
semilogy(2:iter, abs(scarto(2:iter)), '-*b');
% Calcolo raggio spettrale matrice di iterazione Ej
[~,n]=size(A);
Finv=diag(1./diag(A));
EJ=(eye(n)-Finv*A);
ragg_spett=max(abs(eig(EJ)));
```


Metodo di Jacobi

FUNCTION JACOBI Implementazione 1 - MATLAB

```
function [xnew, iter, scarto, asint, vel_conv] = jacobi2(A,b,x0,itmax,toll)
%jacobi Metodo di Jacobi
% [xnew, iter, scarto, asint, vel_conv, n] = jacobi(A,b,x0,itmax,toll);
%
% Dati di Input:
%           A: matrice quadrata del sistema lineare
%           b: termine noto del sistema lineare
%           x0: approssimazione iniziale
%           toll: tolleranza richiesta
%           itmax: numero massimo di iterazioni
%
% Dati di Output:
%           xnew: soluzione all'ultima iterazione
%           iter: numero iterazioni effettuate
%           scarto: vettore degli scarti
%           asint: stima raggio spettrale
%           vel_conv: velocit convergenza metodo
%           n: ordine della matrice A
[m,n]=size(A);
if m~=n
    error('La matrice deve essere quadrata')
end
```

Metodo di Jacobi

FUNCTION JACOBI Implementazione 1 - MATLAB

```
% Inizializzazioni delle variabili
xold=x0; % Impongo valore iniziale di xold
scarto=ones(itmax,1); % Inizializzo il vettore dello scarto
iter=1; % Inizializzo contatore del numero di iterazioni
xnew=zeros(n,1); % Inizializzo il vettore della soluzione xnew
Finv=diag(1./diag(A));
EJ=eye(n)-Finv*A;
q=Finv*b;
% Ciclo while
while scarto(iter)>toll && iter<=itmax
    iter=iter+1;
    xnew=EJ*xold+q;
    sc=xnew-xold;
    scarto(iter)=norm(sc,2);
    xold=xnew;
end
asint=scarto(2:iter)./scarto(1:iter-1);
vel_conv=-log10(asint);
end
```

Metodo di Jacobi

FUNCTION JACOBI Implementazione 2 - MATLAB

```
function [xnew, iter, scarto, asint, vel_conv] = jacobi(A,b,x0,itmax,toll)
%jacobi Metodo di Jacobi
% [xnew, iter, scarto, , asint, vel_conv, n] = jacobi(A,b,x0,itmax,toll);
%
% Dati di Input:
%           A: matrice quadrata del sistema lineare
%           b: termine noto del sistema lineare
%           x0: approssimazione iniziale
%           toll: tolleranza richiesta
%           itmax: numero massimo di iterazioni
%
% Dati di Output:
%           xnew: soluzione all'ultima iterazione
%           iter: numero iterazioni effettuate
%           scarto: vettore degli scarti
%           asint: stima raggio spettrale
%           vel_conv: velocit convergenza metodo
%
[m,n]=size(A);
if m~=n
    error('La matrice deve essere quadrata')
end
```

Metodo di Jacobi

FUNCTION JACOBI Implementazione 2 - MATLAB

```
% Inizializzazioni delle variabili
xold=x0; % Impongo valore iniziale di xold
scarto=ones(itmax,1); % Inizializzo il vettore dello scarto
iter=1; % Inizializzo contatore del numero di iterazioni
xnew=zeros(n,1); % Inizializzo il vettore della soluzione xnew
% Ciclo while
while scarto(iter)>toll && iter<=itmax
    iter=iter+1;
    for i=1:n
        som1=A(i,1:i-1)*xold(1:i-1);
        som2=A(i,i+1:n)*xold(i+1:n);
        xnew(i)=(b(i)-som1-som2)/A(i,i);
    end
    sc=xnew-xold;
    scarto(iter)=norm(sc,2);
    xold=xnew;
end
asint=scarto(2:iter)./scarto(1:iter-1);
vel_conv=-log10(asint);
end
```

Metodo di Seidel

Implementazione 1

$$\mathbf{x}^{(k+1)} = (F - M)^{-1} N \mathbf{x}^{(k)} + (F - M)^{-1} \mathbf{b}$$

Implementazione 2

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij} x_j^{(k)} \right)$$

ESERCIZIO - Metodo di Seidel

Implementare in MATLAB il metodo di Seidel per risolvere il sistema $A\mathbf{x} = \mathbf{b}$ con gli algoritmi 1 e 2. Verificare che i risultati siano equivalenti utilizzando la seguente matrice di prova A e il termine noto \mathbf{b} :

$$A = \begin{pmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 100 \\ 100 \\ 200 \\ 0 \\ 0 \\ 100 \\ 0 \\ 0 \\ 100 \end{pmatrix}$$