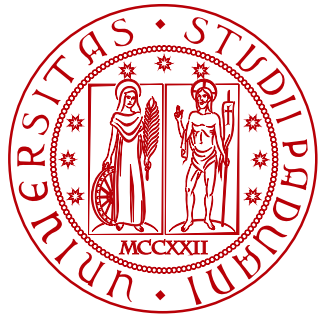

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Laboratorio di Calcolo Numerico
Laboratorio 12:
Metodi iterativi per la soluzione di sistemi lineari

Claudia Zoccarato

E-mail: claudia.zoccarato@unipd.it

Dispense: Moodle Dipartimento ICEA

24 Maggio 2017

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:

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

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

Risolvere sistemi di equazioni lineari

Metodi Iterativi stazionari

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

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

- 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

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

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] = jacobi2(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
F=diag(diag(A));
EJ=-inv(F)*(A-F);
q=inv(F)*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

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

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 $\mathbf{b}$:

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

ESERCIZIO - SOR

Implementare il metodo di sovrarilassamento (SOR) per risolvere il sistema lineare $A\mathbf{x} = \mathbf{b}$ dove la matrice A e il vettore $\mathbf{b}$ possono essere scaricati nella pagina moodle del corso nella cartella `Lab_12`. Si fissi la tolleranza per il criterio di uscita dal ciclo pari a con $toll = 10^{-8}$.

1. Si risolva il sistema dapprima con il metodo di Seidel ($\omega = 1$)
2. Si stimi la costante asintotica di convergenza $\lambda_{1,(S)}$ utilizzando il rapporto tra la norma euclidea degli scarti a due iterazioni successive.
3. Nel caso di Seidel, si riporti in un grafico semilogaritmico la norma euclidea $\|\mathbf{d}^{(\mathbf{k})}\|$ in funzione di k e si calcoli $\lambda_{1,S}$ utilizzando il valore della pendenza del segmento rettilineo del grafico. Si calcoli il valore di ω_{opt} dato da:

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \lambda_{1,S}}}$$

ESERCIZIO - SOR

3. Partendo da $\omega = 1.5$ e finendo con $\omega = 1.9$ con passo $\Delta\omega = 0.01$ si risolva il sistema per i diversi valori di ω. Il modo migliore di procedere è quello di costruire una FUNCTION che implementa il metodo SOR per un dato valore di ω. Una volta che questo è stato verificato, si procede più facilmente a implementare un ciclo per i diversi valori di ω richiesti.
4. Si riporti in un grafico il numero di iterazioni al variare di ω e si stimi il valore approssimato di ω_{opt}.

Note sull'implementazione di SOR

Implementazione 1 - componente per componente

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

Note sull'implementazione di SOR - Pseudocodice

1: dati di input: $\mathbf{x_0}$, $itmax$,$toll$
2: **while** $scarto > toll$ & $iter \leq itmax$
3: $\quad iter = iter + 1$
4: $\quad$ **per** $i = 1, .., n$
5: $\quad\quad \sigma_1 = \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)}$
6: $\quad\quad \sigma_2 = \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}$
7: $\quad\quad \delta = \frac{1}{a_{i,i}}(b_i - \sigma_1 - \sigma_2) - x_i^{(k)}$
8: $\quad\quad x_i^{(k+1)} = x_i^{(k)} + \omega\delta$
9: $\quad$ **fine per**
10: $\quad scarto = ||\mathbf{x^{(k+1)}} - \mathbf{x^{(k)}}||$
11: $\quad \mathbf{x^{(k)}} = \mathbf{x^{(k+1)}}$
12: **fine**