# Esercitazione 6:
# Metodi iterativi per sistemi lineari.

## Richiami di Teoria

*Iterazione di Jacobi e Gauss–Seidel.*

I metodi iterativi sono basati sul calcolo della soluzione $\mathbf{x}$ del sistema lineare $\mathbf{Ax} = \mathbf{b}$ come limite di una successione convergente $\{\mathbf{x}^{(k)}\}$, i.e.

$$\lim_{k\to\infty} \mathbf{x}^{(k)} = \mathbf{x}\,.$$

Una classe particolare di iterazioni sono della forma

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

dove $\mathbf{B} \in \mathbb{R}^{n\times n}$ è detta *matrice di iterazione* e $\mathbf{q} \in \mathbb{R}^n$ è un vettore.
Il seguente teorema caratterizza la convergenza o meno dei metodi stazionari.
*Teorema di convergenza:* Condizione necessaria e sufficiente affinchè la successione $\mathbf{x}^{(k)}$ converga a $\mathbf{x}$ è che

$$\lim_{k\to\infty} \mathbf{B}^k = \mathbf{O}$$

con $\mathbf{O}$ matrice nulla, ossia che il *raggio spettrale* di $\mathbf{B}$, $\rho(\mathbf{B})$, sia minore di $1$.
I metodi iterativi di Jacobi e Gauss–Seidel sono ottenuti dalla decomposizione di $\mathbf{A}$ nella forma $\mathbf{A} = \mathbf{M} - \mathbf{N}$ con $\mathbf{M}$ e $\mathbf{N}$ opportune. Il sistema lineare $\mathbf{Ax} = \mathbf{b}$ può essere riscritto come $\mathbf{Mx} = \mathbf{Nx} + \mathbf{b}$. Pertanto, a partire da un vettore $\mathbf{x}^{(0)}$ arbitrario, si costruisce il seguente schema iterativo

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

ovvero

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

in cui la matrice di iterazione $\mathbf{B}$ assume la forma

$$\mathbf{B} = \mathbf{M}^{-1}\mathbf{N}$$

e il vettore $\mathbf{q}$ è della forma

$$\mathbf{q} = \mathbf{M}^{-1}\mathbf{b}\,.$$

La costruzione di $\mathbf{M}$ e $\mathbf{N}$ nei metodi di Jacobi e Gauss–Seidel avviene a partire dalle seguenti tre matrici elementari:

- Matrice diagonale

$$D = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{pmatrix}$$

- matrice triangole inferiore nella sotto-diagonale

$$L = \begin{pmatrix} 0 & & & & \\ a_{21} & 0 & & & \\ \vdots & \ddots & \ddots & & \\ \vdots & & \ddots & & \\ a_{n1} & \cdots & \cdots & a_{n,n-1} & 0 \end{pmatrix}$$

- matrice triangole superiore nella sopra-diagonale

$$U = \begin{pmatrix} 0 & a_{12} & \cdots & \cdots & a_{1n} \\ & \ddots & \ddots & & \vdots \\ & & \ddots & \ddots & \vdots \\ & & & 0 & a_{n-1,n} \\ & & & & 0 \end{pmatrix}$$

  dove

$$\mathbf{A} = \mathbf{D} + \mathbf{L} + \mathbf{U}\,.$$

In MatLab si utilizzano i comando DIAG, TRIL e TRIU per costruire le precedenti tre matrici.

## Iterazione di Jacobi

Nell'iterazione di Jacobi si sceglie

$$\mathbf{M} = \mathbf{D} \quad \text{e} \quad \mathbf{N} = -(\mathbf{L} + \mathbf{U})\,.$$

dove, la matrice di iterazione $\mathbf{B}_J$ diventa

$$\mathbf{B}_J = -\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})\,.$$

Pertanto, lo schema iterativo diventa

$$\mathbf{x}^{(k+1)} = -\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})\mathbf{x}^{(k)} + \mathbf{D}^{-1}\mathbf{b}$$

che corrisponde, per le componenti dei vettori, a

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

dove ogni componente del nuovo iterato viene calcolata indipendentemente dalle altre componenti, ed utilizzando le componenti del vecchio iterato (eccettuata quella ad essa corispondente).

## Iterazione di Gauss–Seidel

Nell'iterazione di Gauss–Seidel si sceglie

$$\mathbf{M} = \mathbf{D} + \mathbf{L} \quad \text{e} \quad \mathbf{N} = -\mathbf{U} .$$

dove, la matrice di iterazione $\mathbf{B}_{\mathrm{GS}}$ diventa

$$\mathbf{B}_{\mathrm{GS}} = -(\mathbf{D} + \mathbf{L})^{-1}\mathbf{U} .$$

Pertanto, lo schema iterativo diventa

$$\mathbf{x}^{(k+1)} = -(\mathbf{D} + \mathbf{L})^{-1}\mathbf{U}\mathbf{x}^{(k)} + (\mathbf{D} + \mathbf{L})^{-1}\mathbf{b}$$

che corrisponde, per le componenti dei vettori, a

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

dove ogni componente del nuovo iterato viene calcolata utilizzando le nuove componenti già calcolate e le componenti del vecchio iterato non ancora ricalcolate (eccettuata quella ad essa corispondente).

## Esempio

Con riferimento al seguente sistema lineare (con elementi diagonali non nulli)

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3 \end{cases} \Rightarrow \begin{cases} x_1 &= (1/a_{11})(b_1 - a_{12}x_2 - a_{13}x_3) \\ x_2 &= (1/a_{22})(b_2 - a_{21}x_1 - a_{23}x_3) \\ x_3 &= (1/a_{33})(b_3 - a_{31}x_1 - a_{32}x_2) \end{cases}$$

l'iterazione di Jacobi si scrive:

$$\begin{cases} x_1^{(k+1)} &= (1/a_{11}) \left( b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)} \right) \\ x_2^{(k+1)} &= (1/a_{22}) \left( b_2 - a_{21}x_1^{(k)} - a_{23}x_3^{(k)} \right) \\ x_3^{(k+1)} &= (1/a_{33}) \left( b_3 - a_{31}x_1^{(k)} - a_{32}x_2^{(k)} \right) \end{cases}$$

mentre l'iterazione di Gauss–Seidel diventa:

$$\begin{cases} x_1^{(k+1)} &= (1/a_{11})\left(b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)}\right) \\ x_2^{(k+1)} &= (1/a_{22})\left(b_2 - a_{21}x_1^{(k+1)} - a_{23}x_3^{(k)}\right) \\ x_3^{(k+1)} &= (1/a_{33})\left(b_3 - a_{31}x_1^{(k+1)} - a_{32}x_2^{(k+1)}\right) \end{cases}$$

## Test di arresto

Vi sono due test principali che possono essere utilizzati:

- *Test sullo scarto tra due iterate:* Si potrà fermare un metodo iterativo all'iterazione $k+1$ quando
$$||\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}||_\infty < \epsilon\,,$$
- *Test sul residuo:* Si potrà fermare un metodo iterativo all'iterazione $k$ quando
$$\frac{||\mathbf{r}^{(k)}||_2}{||\mathbf{b}||_2} < \epsilon\,,$$
dove $\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}$.

## Nota:

In MatLab per estrarre una matrice triangolare inferiore si utilizza il comando `tril(A)` mentre una matrice triangolare inferiore con diagonale esclusa si scrive `tril(A,-1)` cioè si estree la triangolare inferiore a partire dalla diagonale $-1$ che corrisponde alla sottodiagonale della diagonale principale; la diagonale principale è individuata con $0$, la sopradiagonale a quella principale con $+1$.

# Esercizio 1

Per ognuna delle seguenti matrici

$$\mathbf{A}_1 = \begin{pmatrix} 3 & 0 & 4 \\ 7 & 4 & 2 \\ -1 & 1 & 2 \end{pmatrix} \qquad \mathbf{A}_2 = \begin{pmatrix} -3 & 3 & -6 \\ -4 & 7 & -8 \\ 5 & 7 & -9 \end{pmatrix}$$

$$\mathbf{A}_3 = \begin{pmatrix} 4 & 1 & 1 \\ 2 & -9 & 0 \\ 0 & -8 & 6 \end{pmatrix} \qquad \mathbf{A}_4 = \begin{pmatrix} 7 & 6 & 9 \\ 4 & 5 & -4 \\ -7 & -3 & 8 \end{pmatrix}$$

disengnare:

- lo spettro della matrice $\mathbf{A}_i$;

- lo spettro della matrice di iterazione di Jacobi $\mathbf{B}_J$;
- lo spettro della matrice di iterazione di Gauss–Seidel $\mathbf{B}_{\mathrm{GS}}$;
- approssimare la soluzione madiante il metodo di Jacobi;
- approssimare la soluzione madiante il metodo di Gauss–Seidel.

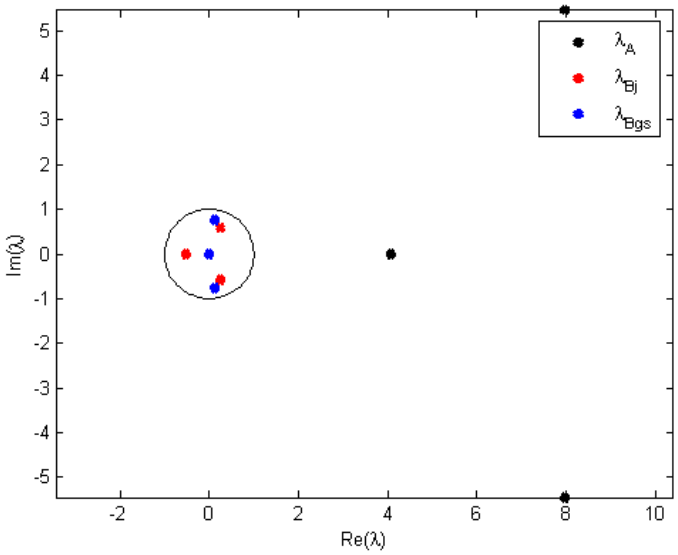
Dire inoltre analizzando gli spettri di tali matrici di iterazione la convergenza dei metodi.

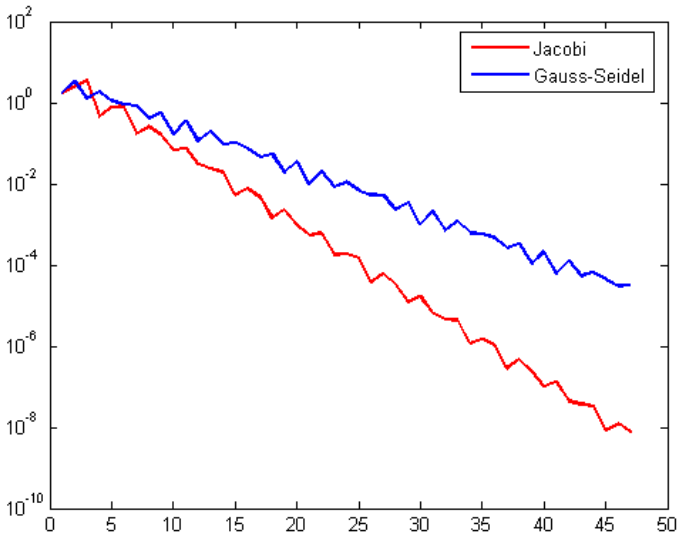
L'algoritmo di Jacobi riceve in ingresso la matrice $\mathbf{A}$, il vettore $\mathbf{b}$, il punto iniziale $\mathbf{x}_0$, la tolleranza $\mathrm{tol}$ e il numero massimo di iterazioni da eseguire $\mathrm{maxiter}$. In uscita dovrà restituire il vettore soluzione $\mathbf{x}$ ed un flag $\mathrm{ok}$ se l'algoritmo è arrivato o no a convergenza. Utilizzare un testo di arresto basato sullo scarto con in aggiunta un test di arresto su un numero massimo di iterazioni.

```
[x,ok] ← jacobi ( A, b, x_0, tol, maxiter )
 1  // copio punto iniziale nella soluzione
 2  x ← x_0
 3  // imposto il flag ad un quantita' falsa
 4  ok ← false
 5  // contatore iniziale delle iterazioni
 6  iter ← 0
 7  // scarto iniziale maggiore della tolleranza in modo da entrare nel ciclo iterativo
 8  scarto ← tol +1
 9  while  scarto ≥ tol and iter < maxiter do
10      // incremento iterazione
11      iter ← iter +1
12      // inizializzo soluzione al nuovo passo
13      xnew ← 0
14      // calcolo aggiornamento di Jacobi
15      for  i ← 1 to n
16          // prima sommatoria
17          for  j ← 1 to i − 1
18              xnew_i ← xnew_i +A_ij x_j
19          end
20          // seconda sommatoria
21          for  j ← i + 1 to n
22              xnew_i ← xnew_i +A_ij x_j
23          end
24          xnew_i ← (b_i − xnew_i)/a_ii
25      end
26      // aggiorno variabili
27      scarto ← ||xnew − x||_∞
28      x ← xnew
29  end
30  if scarto ≤ tol then
31      ok ← true
32  end
```

# Risultati

Ad esempio, per la matrice $\mathbf{A}_4$ abbiamo i seguenti risultati.



## Nota:

Sfruttando le potenzialità di MatLab/Octave i metodi di Jacobi e Gauss–Seidel sarebbero scritti nel seguenti modo:

```
function [x,flag,res] = mJacobi(A,b,x0,tol,maxiter)

% Vettore
x = x0;
flag = false;
iter = 0;

% Matrici di Jabobi
Minv_jac = 1./diag(A);              % vettore inversi diagonale
N_jac = -(tril(A,-1)+triu(A,+1));

normb   = norm(b);      % norma del vettore b
residuo = norm(b-A*x);  % residuo iniziale

% salvo tutti i residuo
res = residuo;

% Ciclo iterativo principale
while residuo >= tol && iter<maxiter,
    iter = iter+1;

    x = Minv_jac.*(N_jac*x+b);
    residuo = norm(b-A*x);
    res(iter+1) = residuo;
end

if( residuo<tol )
    flag = true;
end


function [x,flag,res] = mGaussSeidel(A,b,x0,tol,maxiter)

% Vettore
x = x0;
flag = false;
iter = 0;

% Matrici di Gauss-Seidel
M_gs = tril(A);
N_gs = -triu(A,+1);

normb   = norm(b);      % norma del vettore b
residuo = norm(b-A*x);  % residuo iniziale
```

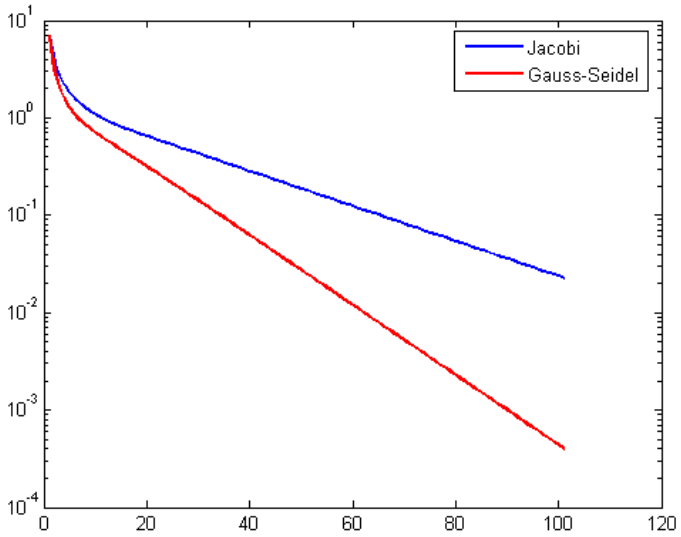
```
% salvo tutti i residuo
res = residuo;

% Ciclo iterativo principale
while residuo ≥ tol && iter<maxiter,
    iter = iter+1;

    x = M_gs\(N_gs*x+b);
    residuo = norm(b-A*x);
    res(iter+1) = residuo;
end

if( residuo<tol )
    flag = true;
end
```

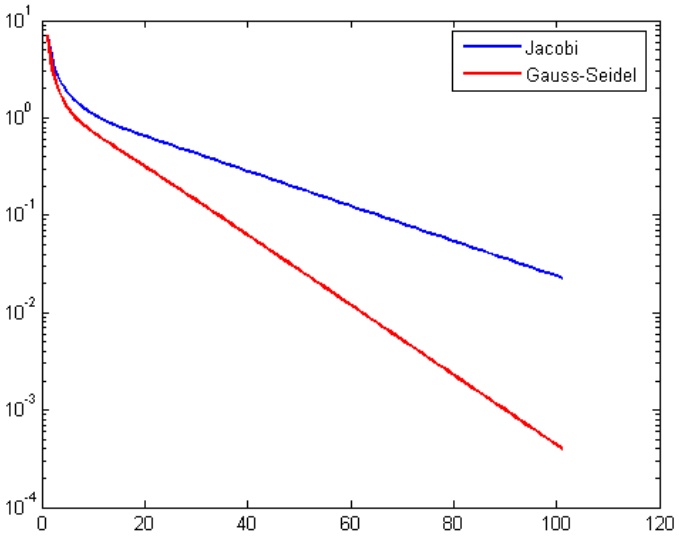
Questa implementazione funziona anche con il caso di matrici sparse.

# Esercizio 2

Provare Jacobi e Gauss-Seidel nella matrice simmetrica e definita positiva generata dal comando MatLab `A = gallery('poisson',10)`.
Riportare l'andamento del residuo.

# Risultati




Jacobi
Gauss-Seidel
$10^{1}$
$10^{0}$
$10^{-1}$
$10^{-2}$
$10^{-3}$
$10^{-4}$
0
20
40
60
80
100
120