Metodi iterativi per sistemi lineari

Lucia Gastaldi

Dipartimento di Matematica,
http://dm.ing.unibs.it/gastaldi/

8 febbraio 2008

Outline

Costruzione di un metodo iterativo

Consideriamo il sistema lineare $Ax = b$ con $\det(A) \neq 0$.
Introduciamo la seguente decomposizione della matrice $A = P - N$. Allora si ha

$$Ax = b \;\Rightarrow\; (P - N)x = b \;\Rightarrow\; Px = Nx + b \;\Rightarrow\; x = P^{-1}Nx + P^{-1}b$$

Il sistema è ricondotto ad un problema di punto fisso. Applichiamo il procedimento delle approssimazioni successive

$$x^{(k+1)} = P^{-1}Nx^{(k)} + P^{-1}b$$
$$x^{(k+1)} = Bx^{(k)} + g$$

B **matrice di iterazione**

Convergenza dei metodi iterativi

Sia $e^{(k)} = x - x^{(k)}$ l'errore al passo k.
Si ottiene che

$$e^{(k+1)} = Be^{(k)} = B^{k+1}e^{(0)}.$$

Teorema

Se $\|B\| < 1$ allora il metodo iterativo con matrice di iterazione B è convergente.

Teorema

Condizione necessaria e sufficiente per la convergenza del metodo iterativo con matrice di iterazione B è

$$\rho(B) < 1,$$

essendo $\rho(B)$ il **raggio spettrale** della matrice B ossia

$$\rho(B) = \max_{1 \le i \le n} |\lambda_i| \quad \text{con } \lambda_i \text{ autovalori di } B.$$

Matrici sparse

Il costo di un metodo iterativo è dato dal

numero delle iterazioni x costo del prodotto di B per un vettore

Quindi l'uso dei metodi iterativi è consigliato nel caso di matrici sparse e di grandi dimensioni.

Il formato sparse è utilizzato in Matlab per ridurre i costi di memorizzazione della matrice.

`S=sparse(A)` converte la matrice in formato full in una matrice in formato sparse tenendo in memoria solo gli elementi diversi da zero.

`S = sparse(i,j,s,m,n)` usa i vettori `i`, `j` e `s` per costruire la matrice di dimensione $m \times n$ tale che `S(i(k),j(k)) = s(k)`.

Il comando `spy(A)` mostra in un grafico quali sono gli elementi di A non nulli ed il loro numero `nnz`.

Il comando `spdiags`

Il comando `spdiags` generalizza il comando `diag`.

Sono disponibili quattro operazioni differenti.

- `B = spdiags(A)` estrae tutte le diagonali non nulle dalla matrice A. Le p colonne di B sono le diagonali di A.
  `[B,d] = spdiags(A)` fornisce anche il vettore `d` di lunghezza p, i cui valori specificano le diagonali di A.
- `B = spdiags(A,d)` estrae le diagonali specificate da d.
- `A = spdiags(B,d,A)` sostituisce le diagonali specificate da d con le colonne di B.
- `A = spdiags(B,d,m,n)` crea una matrice sparsa $m \times n$ prendendo le colonne di B e mettendole al posto delle diagonali specificate da d.

Decomposizione della matrice

$$A = D - E - F$$

essendo

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

$$E = -\begin{pmatrix} 0 & 0 & \cdots & 0 \\ a_{21} & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & \cdots & a_{nn-1} & 0 \end{pmatrix}, \quad F = -\begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & a_{n-1n} \\ 0 & \cdots & 0 & 0 \end{pmatrix}.$$

Metodo di Jacobi

La scelta $P = D$ dà luogo al **metodo di Jacobi**, che si può scrivere componente per componente nella forma:

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

La matrice di iterazione è:

$$B_J = D^{-1}(E + F) = D^{-1}(D - A) = I - D^{-1}A.$$

Metodo di Gauss-Seidel

Ponendo $P = D - E$ si ricava il **metodo di Gauss-Seidel** che componente per componente si scrive:

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

La matrice di iterazione è:

$$B_{GS} = (D - E)^{-1} F = (D - E)^{-1}(D - E - A) = I - (D - E)^{-1} A.$$

Esercizio

Si considerino le seguenti matrici e vettori:

$$A_1 = \begin{pmatrix} 1 & -2 & 2 \\ -1 & 1 & -1 \\ -2 & -2 & 1 \end{pmatrix} \qquad A_2 = \begin{pmatrix} 2 & -1 & 1 \\ 2 & 2 & 2 \\ -1 & -1 & 2 \end{pmatrix} \qquad b = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}$$

- Calcolare la soluzione dei sistemi lineari $A_1 x = b$ e $A_2 y = b$ mediante il metodo diretto.
- Calcolare poi la soluzione di entrambi i sistemi lineari mediante i metodi iterativi di Jacobi e Gauss-Seidel usando le function `jacobi` e `gseidel` come segue:

```
[x,iter]=jacobi(A,b,x0,toll,nmax)
[x,iter]=gseidel(A,b,x0,toll,nmax)
```

Function `jacobi` e `gseidel`

I comandi per usare le function `jacobi` e `gseidel` sono rispettivamente:

```
[x,iter]=jacobi(A,b,x0,toll,nmax)
[x,iter]=gseidel(A,b,x0,toll,nmax)
```

essendo

`A,b`	matrice e termine noto del sistema;
`x0`	vettore iniziale;
`toll`	precisione desiderata (ad es. `1.e-6`);
`nmax`	numero massimo di iterazioni (ad es. 300);
`x`	soluzione calcolata;
`iter`	numero di iterazioni eseguite.

Le function forniscono una tabella che contiene:

1 col	numero di iterazione k
2 col	residuo $\|r^{(k)}\| = \|b - Ax^{(k)}\|$
3 col	differenza fra iterate successive $\delta^{(k)} = \|x^{(k)} - x^{(k-1)}\|$
4 col	stima del raggio spettrale mediante il rapporto $\delta^{(k)}/\delta^{(k-1)}$

Esercizio (segue)

Confrontare il numero di iterazioni impiegate con quello previsto, calcolando il raggio spettrale della matrice di iterazione.
Si ricorda che le matrici di iterazione dei metodi di Jacobi e di Gauss-Seidel sono date rispettivamente da:

$$B_J = I - D^{-1}A \ (D \text{ diagonale di } A)$$

$$B_{GS} = I - (D - E)^{-1}A \ (D - E \text{ triangolare inferiore di } A).$$

Per calcolare gli autovalori usare il comando `eig(B)`.

Costruzione della matrice B

`eye(n)` fornisce la matrice identità di dimensione $n \times n$.

Per costruire la matrice B_J usare il comando `diag`.
Se v è un vettore, `diag(v,k)` fornisce la matrice che ha come k-esima diagonale v; se A è una matrice, `diag(A,k)` estrae il vettore che contiene la k-esima diagonale di A.

Per costruire la matrice B_{GS} usare il comando `tril`.
Il comando `tril(A)` estrae la parte triangolare inferiore della matrice A.
Il comando `triu` fornisce la parte triangolare superiore.

Esercizio

Si consideri l'equazione differenziale con valori ai limiti:

$$-u''(x) = 1 + x \quad x \in [0,1], \qquad u(0) = u(1) = 0.$$

La soluzione esatta è: $u(x) = \frac{2}{3}x - \frac{x^2}{2} - \frac{x^3}{6}$.

- Calcolare con un metodo diretto la soluzione del sistema lineare $Ax = b$.
- Calcolare poi la soluzione del sistema lineare mediante i metodi iterativi di Jacobi e Gauss-Seidel usando le function `jacobi` e `gseidel`.
- Confrontare la stima del raggio spettrale della matrice di iterazione con i seguenti valori esatti:

$$\rho(B_J) = \cos(\pi h) \qquad \rho(B_{GS}) = \rho(B_J)^2.$$

- Calcolare l'errore relativo rispetto alla soluzione esatta del problema e verificare che è paragonabile a quello ottenuto risolvendo il sistema lineare con il metodo diretto.

Il metodo di Richardson stazionario

Posto $A = P - N$ e quindi $N = P - A$, un metodo iterativo nella forma

$$x^{(k+1)} = P^{-1}Nx^{(k)} + P^{-1}b$$

si può riscrivere come segue

$$\begin{aligned} x^{(k+1)} &= P^{-1}(P - A)x^{(k)} + P^{-1}b = (I - P^{-1}A)x^{(k)} + P^{-1}b \\ &= x^{(k)} + P^{-1}(b - Ax^{(k)}) = x^{(k)} + P^{-1}r^{(k)}. \end{aligned}$$

La convergenza del metodo si ha se $\rho(I - P^{-1}A) < 1$.
Se questo non succede si può modificare il metodo introducendo un opportuno parametro α come segue

$$x^{(k+1)} = x^{(k)} + \alpha P^{-1}r^{(k)} \quad \text{in modo tale che } \rho(I - \alpha P^{-1}A) < 1.$$

Il metodo così costruito si chiama **metodo di Richardson stazionario**.

Convergenza del metodo di Richardson stazionario

$B_\alpha = I - \alpha P^{-1}A$ è la matrice di iterazione del metodo di Richardson stazionario.
Supponiamo che la matrice $P^{-1}A$ sia simmetrica e definita positiva e che λ_i, u_i siano i suoi autovalori ed autovettori cioè: $P^{-1}Au_i = \lambda_i u_i$.
Ordiniamo gli autovalori in modo che valga la seguente relazione:

$$0 < \lambda_{min} = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n = \lambda_{max}.$$

Teorema

Il metodo di Richardson stazionario converge **se e solo se**

$$0 < \alpha < \frac{2}{\lambda_{max}}.$$

Il minimo di $\rho(B_\alpha)$ al variare di α si ottiene per

$$\alpha_{ott} = \frac{2}{\lambda_{min} + \lambda_{max}} \quad \text{e si ha } \rho(B_{\alpha_{ott}}) = \frac{\lambda_{max} - \lambda_{min}}{\lambda_{max} + \lambda_{min}}.$$

Precondizionatori

La matrice P che compare nell'espressione del metodo di Richardson si chiama **matrice di precondizionamento**.
La scelta di P deve essere guidata dalla seguente condizione:

$$K(P^{-1}A) << K(A).$$

L'ideale sarebbe avere $K(P^{-1}A) \approx 1$ e quindi P^{-1} rappresenta un'approssimazione dell'inversa di A.

Precondizionatori di uso comune

- Precondizionatori diagonali: si sceglie P uguale alla diagonale di A se A è simmetrica e definita positiva. Se A non è simmetrica si può porre:

$$p_{ii} = \left(\sum_{j=1}^{n} a_{ij}^2 \right)^{1/2} .$$

- Fattorizzazione incompleta
- Precondizionatori polinomiali: Posto $A = D - C$ si ha $A = (I - CD^{-1})D$. Quindi l'inversa è data da:

$$A^{-1} = D^{-1}(I - CD^{-1})^{-1} = D^{-1}(I + CD^{-1} + (CD^{-1})^2 + \dots)$$

Troncando lo sviluppo in serie ad una certa potenza si ottiene un valore approssimato di A^{-1}.
Per esempio si può scegliere $P^{-1} = D^{-1}(I + CD^{-1})$.

Fattorizzazione incompleta

La fattorizzazione incompleta si ottiene con i comandi:

- `luinc` per matrici generali;
- `cholinc` per matrici simmetriche e definite positive.

`[L,U]=luinc(A,'0')` fornisce i fattori L e U con lo stesso schema di sparsità della matrice A.

`[L,U]=luinc(A,droptol)` calcola la fattorizzazione incompleta eliminando i termini della fattorizzazione che sono inferiori a `droptol`.

Function `richstaz`

```
[x,iter]=richstaz(A,b,x0,toll,nmax,alfa,prec,P1,P2)
```

INPUT

`A, b`	matrice e termine noto
`x0`	vettore iniziale
`toll`	tolleranza
`nmax`	numero massimo di iterazioni
`prec`	tipo di precondizionatore:
`prec=1`	precondizionatore diag: P1=P, P2 non assegnata
`prec=2`	P=P1*P2 con P1 e P2 matrici triangolari inf e sup risp.
`prec=3`	approssimazione dell'inversa $P^{-1} = D^{-1} * (I + CD^{-1})$ assegnare P1=D e $P2 = I + CD^{-1}$

OUTPUT

`x`	soluzione
`iter`	numero di iterazioni eseguite

Problema di Neumann

Si consideri il seguente problema differenziale:

$$-u''(x) = f(x) \text{ per } x \in [-1,1], \quad u'(-1) = u'(1) = 0 \quad \int_{-1}^{1} u(x)dx = 0.$$

Posto $f(x) = x$, la soluzione esatta è: $u(x) = \frac{x}{2} - \frac{x^3}{6}$.
Posto $h = 2/(n-1)$ e $x = -1 + ih$ per $i = 0, \ldots, n-1$, mediante il metodo delle differenze finite il problema si riconduce alla soluzione del sistema lineare $A^h\mathbf{u} = b$ essendo $A^h \in \mathbb{R}^{(n+1)\times(n+1)}$, $b \in \mathbb{R}^{n+1}$, $\mathbf{u} \in \mathbb{R}^{n+1}$ con

$$A^h = \begin{pmatrix} A_1 & v \\ -v^T & 1 \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ 0 \end{pmatrix}, \quad \mathbf{u} = \begin{pmatrix} u^h \\ m \end{pmatrix},$$

$u^h \in \mathbb{R}^n$ soluzione discreta, m valore della media e

$$A_1 = \frac{1}{h^2}\begin{pmatrix} 1 & -1 & 0 & \ldots & 0 \\ -1 & 2 & -1 & \ldots & 0 \\ \ldots & -1 & 2 & -1 & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & -1 & 2 & -1 \\ 0 & \ldots & 0 & -1 & 1 \end{pmatrix} \quad v = \begin{pmatrix} 1/2 \\ 1 \\ \ldots \\ 1 \\ 1/2 \end{pmatrix} \quad b_1 = \begin{pmatrix} f(x_1)/2 \\ f(x_2) \\ \ldots \\ f(x_{n-1}) \\ f(x_n)/2 \end{pmatrix}$$

Esercizio (segue)

Costruire la matrice A^h in formato sparse.

- Costruire il vettore `e=ones(n,1)`;
- `[-e 2*e -e]` dà una matrice che ha su ciascuna colonna le diagonali di A_1;
- usare il comando `A=spdiags([-e 2*e -e],-1:1,n,n)` per costruire la matrice A_1;
- correggere gli elementi `A(1,1)` e `A(n,n)`;
- dividere per h^2 la matrice costruita;
- costruire il vettore v;
- mettere insieme tutti i blocchi `A=[A v;v' 1]`.
- visualizzare la struttura della matrice con il comando `spy`

Costruire il vettore b.
Risolvere il sistema lineare.
La soluzione u^h è data dalle prime n componenti del vettore soluzione.
Plottare la soluzione ottenuta insieme a quella esatta.

Esercizio sul problema di Neumann

Si consideri il sistema lineare ottenuto dall'applicazione delle differenze finite al problema di Neumann.

- Posto `toll=1.e-5` e `nmax=100`, risolvere il sistema lineare con i metodi di Jacobi e Gauss-Seidel.
- Osservare che non convergono entrambi i metodi.
- Usare il metodo di Richardson con matrice di precondizionamento P data dalla diagonale di A. Trovare, andando per tentativi, un valore di α per il quale il metodo di Richardson è convergente.
- Usare il precondizionamento dato da

$$p_{ii} = \left(\sum_{j=1}^{n} a_{ij}^2 \right)^{1/2} .$$

Metodi iterativi e risoluzione del Laplaciano con condizioni di Dirichlet

Si consideri il seguente problema:

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega = [-1,1] \times [-1,1] \\ u &= g \qquad \text{su } \partial\Omega. \end{aligned} \tag{1}$$

La function `dirichlets` costruisce la matrice in formato sparse e il termine noto mediante il seguente comando:

```
[A,f]=dirichlets(effe,a1,a2,b1,b2,g1,g2,g3,g4,Nx,Ny)
```

con le stesse notazioni della function `dirichlet`.

Esercizio

Risoluzione del problema del Laplaciano con i metodi iterativi

Costruire la matrice dei coefficienti e il termine noto corrispondente alla soluzione del problema(1) con differenze finite. Per risolvere il sitema lineare ottenuto applicare i seguenti metodi (usare `toll=1.e-5` e `nmax=300`):

- metodo di Jacobi;
- metodo di Gauss-Seidel;
- metodo di Richardson con precondizionatore diagonale ($\alpha = .8$);
- metodo di Richardson precondizionato con la fattorizzazione incompleta ($\alpha = 1$);
- metodo di Richardson con precondizionatore polinomiale ($\alpha = 1$);
- metodo di Richardson con precondizionatore tridiagonale (estrarre le diagonali principali di A e usare la loro fattorizzazione di Choleski;
- confrontare il numero di iterazioni richieste per $n = 10$.

Rappresentazione della soluzione

- suddivisione dell'intervallo $[-1, 1]$ con $n + 2$ punti equispaziati: `X=linspace(-1,1,n+2)`;
- costruzione della griglia: `[x,y]=meshgrid(X)`;
- ricostruzione della soluzione in un array `(n+2)x(n+2)` in modo di tenere in considerazione anche le condizioni al bordo:

```
k=1;
for i=2:n+1
   for j=2:n+1
      U(i,j)=u(k); k=k+1;
   end
end
U(1,1)=(g1(0)+g4(0))/2; U(1,Nx+2)=(g1(a)+g2(0))/2;
U(Ny+2,Nx+2)=(g2(b)+g3(a))/2; U(Ny+2,1)=(g3(0)+g4(b))/2;
U(1,2:Nx+1)=g1((1:Nx)*hx);
U(Ny+2,2:Nx+1)=g3((1:Nx)*hx);
U(2:Ny+1,Nx+2)=g2((1:Ny)*hy);
U(2:Ny+1,1)=g4((1:Ny)*hy);
```

- `surf(x,y,U)`

Il metodo di Richardson non stazionario

Il metodo di Richardson non stazionario può essere applicato al caso di matrici A simmetriche e definite positive.

Algoritmo

1. Assegnazione dei dati: A, b, $x^{(0)}$;
2. Calcolo del residuo iniziale: $r^{(0)} = b - Ax^{(0)}$;
3. Calcolo della direzione iniziale: $Pz^{(0)} = r^{(0)}$;
4. Fino a convergenza ripetere:
   a. $\alpha_k = \dfrac{{z^{(k)}}^T r^{(k)}}{{z^{(k)}}^T A z^{(k)}}$
   b. $x^{(k+1)} = x^{(k)} + \alpha_k z^{(k)}$
   c. $r^{(k+1)} = r^{(k)} - \alpha_k A z^{(k)}$
   d. $Pz^{(k+1)} = r^{(k+1)}$

Il metodo del gradiente o di massima discesa

Sia A simmetrica e definita positiva, se $P = I$ (matrice identità) allora il metodo di Richardson non stazionario prende il nome di Metodo del gradiente.

Algoritmo

1. Assegnazione dei dati: A, b, $x^{(0)}$;
2. Calcolo del residuo iniziale: $r^{(0)} = b - Ax^{(0)}$;
3. Fino a convergenza ripetere:
   a. $\alpha_k = \dfrac{{r^{(k)}}^T r^{(k)}}{{r^{(k)}}^T A r^{(k)}}$
   b. $x^{(k+1)} = x^{(k)} + \alpha_k r^{(k)}$
   c. $r^{(k+1)} = r^{(k)} - \alpha_k A r^{(k)}$

La function `gradiente` risolve il sistema lineare $Ax = b$ con il metodo del gradiente:

```
[x,iter]=gradiente(A,b,x0,toll,nmax)
```

Il metodo del gradiente coniugato

Il **metodo del gradiente coniugato** si applica nel caso di matrici simmetriche e definite positive ed è caratterizzato dalla scelta di direzioni $s^{(k)}$ tali che:

$$s^{(k)\,T} A s^{(j)} = 0 \quad \text{per } k \neq j.$$

Siccome le direzioni $s^{(k)}$ sono linearmente indipendenti il metodo del gradiente coniugato è a terminazione finita.

Algoritmo

1. Assegnazione dei dati: A, b, $x^{(0)}$;
2. Residuo e direzione iniziale: $r^{(0)} = b - Ax^{(0)}$, $s^{(0)} = r^{(0)}$;
3. Fino a convergenza ripetere:
   a. $\alpha_k = \dfrac{s^{(k)\,T} r^{(k)}}{s^{(k)\,T} A s^{(k)}}$
   b. $x^{(k+1)} = x^{(k)} + \alpha_k s^{(k)}$
   c. $r^{(k+1)} = r^{(k)} - \alpha_k A s^{(k)}$
   d. $\beta_k = -\dfrac{s^{(k)\,T} A r^{(k+1)}}{s^{(k)\,T} A s^{(k)}}$
   e. $s^{(k+1)} = r^{(k+1)} + \beta_k s^{(k)}$

Il metodo del gradiente coniugato precondizionato

`x = pcg(A,b)`

fornisce la soluzione del sistema lineare $Ax = b$ mediante il metodo del gradiente coniugato.

`x = pcg(A,b,toll,nmax)`

Viene specificata la tolleranza (default `tol=1.e-6`) e il numero massimo di iterazioni (default `nmax=min(N,20)`).

`x = pcg(A,b,toll,nmax,P)`

fornisce la soluzione del sistema lineare precondizionato $P^{-1}Ax = P^{-1}b$.

`x = pcg(A,b,toll,nmax,M1,M2)`

fornisce la soluzione di $P^{-1}Ax = P^{-1}b$, essendo $P = M1 * M2$.

`x = pcg(A,b,toll,nmax,M1,M2,x0)`

vettore iniziale `x0` assegnato dall'utente (default `x0=zeros(size(b))`).

Il metodo del gradiente coniugato precondizionato (segue)

`[x,flag,relres,iter,resvec] = pcg(A,b,...)`

fornisce ulteriori informazioni:

- `flag`
  - `flag=0` il metodo ha ottenuto la soluzione;
  - `flag=1` dopo `nmax` iterate non è stata raggiunta la soluzione;
  - `flag=2` la matrice P è mal condizionata;
  - `flag=3` due successive iterate erano uguali;
  - `flag=4` una delle quantità scalari è diventata troppo piccola o troppo grande.
- `relres` residuo relativo `norm(b-A*x)/norm(b)`
- `iter` numero di iterazioni effettuate
- `resvec` vettore dei residui in norma (iterata per iterata)

Esercizio

Risoluzione del problema del Laplaciano con i metodi iterativi

Costruire la matrice dei coefficienti e il termine noto corrispondente alla soluzione del problema(1) con differenze finite (usare la function `dirichlets`). Per risolvere il sitema lineare ottenuto applicare i seguenti metodi (usare `toll=1.e-5` e `nmax=300`):

- metodo di Richardson precondizionato con la fattorizzazione incompleta ($\alpha = 1$);
- metodo del gradiente o di massima discesa;
- metodo del gradiente coniugato;
- metodo del gradiente coniugato precondizionato con la fattorizzazione incompleta;
- confrontare il numero di iterazioni richieste per $n = 10$.

Metodi a terminazione finita per matrici qualunque

Se A è una matrice non simmetrica, A è **definita positiva** se la matrice simmetrica $(A + A^T)/2$ è definita positiva.

Nel caso di matrici non simmetriche o non definite positive si possono usare i seguenti metodi iterativi, che hanno la proprietà di essere a terminazione finita, ossia in matematica esatta forniscono la soluzione esatta dopo n iterazioni.

Funzione	**Metodo**
gmres	Generalized Minimum Residual Method
cgs	Conjugate Gradients Squared Method
bicg	BiConjugate Gradients Method
bicgstab	BiConjugate Gradients Stabilized Method

gmres

```
x = gmres(A,b)
```

fornisce la soluzione del sistema lineare $Ax = b$ mediante il metodo **gmres**.

```
x = gmres(A,b,restart)
```

il metodo **gmres** viene riinizializzato ogni `restart` iterazioni. Se `restart=N` o `restart=[]` il metodo non viene mai riinizializzato.

```
x = gmres(A,b,restart,toll,nmax)
```

Viene specificata la tolleranza (default `tol=1.e-6`) e il numero massimo di iterazioni esterne (default `nmax=min(N/restart,10)`).

```
x = gmres(A,b,restart,toll,nmax,P)
x = gmres(A,b,restart,toll,nmax,M1,M2)
```

fornisce la soluzione del sistema lineare precondizionato $P^{-1}Ax = P^{-1}b$ o $P = M1 * M2$.

gmres (segue)

`[x,flag,relres,iter,resvec] = gmres(A,b,...)`

fornisce ulteriori informazioni:

- `flag`
  - `flag=0` il metodo ha ottenuto la soluzione;
  - `flag=1` dopo `nmax` iterate non è stata raggiunta la soluzione;
  - `flag=2` la matrice P è mal condizionata;
  - `flag=3` due successive iterate erano uguali;
- `relres` residuo relativo `norm(b-A*x)/norm(b)`
- `iter` numero di iterazioni effettuate: `iter(1)<=nmax` è il numero delle iterazioni del ciclo esterno; `iter(1)<=restart` numero delle iterate interne.
- `resvec` vettore dei residui in norma (iterata per iterata)

Esercizio

- Usare la function `gmres` per risolvere il problema di Neumann con 10, 20 e 50 punti.
- Usare la fattorizzazione incompleta provvista da `luinc` per ottenere la soluzione nel caso di 50 punti.

bicg, cgs, bicgstab

`x = solver(A,b)`

fornisce la soluzione del sistema lineare $Ax = b$ mediante il metodo prescelto: **bicg**, **cgs**, **bicgstab**.

`x = solver(A,b,toll,nmax)`

Viene specificata la tolleranza (default `tol=1.e-6`) e il numero massimo di iterazioni (default `nmax=min(N,20)`).

`x = solver(A,b,toll,nmax,P)`

fornisce la soluzione del sistema lineare precondizionato $P^{-1}Ax = P^{-1}b$.

`x = solver(A,b,toll,nmax,M1,M2)`

fornisce la soluzione di $P^{-1}Ax = P^{-1}b$, essendo $P = M1 * M2$.

`x = solver(A,b,toll,nmax,M1,M2,x0)`

vettore iniziale `x0` assegnato dall'utente (default `x0=zeros(size(b))`).

`bicg, cgs, bicgstab` (segue)

`[x,flag,relres,iter,resvec] = solver(A,b,...)`

fornisce ulteriori informazioni:

- `flag`
  - `flag=0` il metodo ha ottenuto la soluzione;
  - `flag=1` dopo `nmax` iterate non è stata raggiunta la soluzione;
  - `flag=2` la matrice P è mal condizionata;
  - `flag=3` due successive iterate erano uguali;
  - `flag=4` una delle quantità scalari è diventata troppo piccola o troppo grande.
- `relres` residuo relativo `norm(b-A*x)/norm(b)`
- `iter` numero di iterazioni effettuate
- `resvec` vettore dei residui in norma (iterata per iterata)

Esercizio

- Testare una delle function `bicg`, `cgs` o `bicgstab` per risolvere il problema di Neumann con 10, 20 e 50 punti.
- Usare la fattorizzazione incompleta provvista da `luinc` per ottenere la soluzione nel caso di 50 punti.