Metodi iterativi per sistemi lineari

Lucia Gastaldi

DICATAM - Sezione di Matematica,
http://lucia-gastaldi.unibs.it

UNIVERSITÀ
DEGLI STUDI
DI BRESCIA

Indice

Costruzione di un metodo iterativo

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

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

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

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

$B = P^{-1}(P - A)$ **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 A in formato full nella matrice S 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` generalizza il comando `diag` alle matrici sparse.

Metodo di Jacobi

La scelta P uguale alla matrice diagonale contenente la diagonale di A `P=diag(diag(A))` 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:

$$B_J = P^{-1}(P - A) = I - P^{-1}A.$$

Metodo di Gauss-Seidel

Ponendo P uguale alla matrice triangolare ingeriore che si estrae da A `P=tril(A)` si ricava il **metodo di Gauss-Seidel** che componente per componente si scrive:

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

per $i = 1, \ldots, n$.
La matrice di iterazione è:

$$B_{GS} = P^{-1}(P - A) = I - P^{-1}A.$$

Esercizio 1

Si considerino le seguenti matrici e vettori:

$$A_1 = \begin{bmatrix} 3 & 0 & 4 \\ 7 & 4 & 2 \\ -1 & 1 & 2 \end{bmatrix} \quad b_1 = \begin{bmatrix} 7 \\ 13 \\ 2 \end{bmatrix} \qquad A_2 = \begin{bmatrix} -3 & 3 & -6 \\ -4 & 7 & -8 \\ 5 & 7 & -9 \end{bmatrix} \quad b_2 = \begin{bmatrix} -6 \\ -5 \\ 3 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 4 & 1 & 1 \\ 2 & -9 & 0 \\ 0 & -8 & -6 \end{bmatrix} \quad b_3 = \begin{bmatrix} 6 \\ -7 \\ -14 \end{bmatrix} \qquad A_4 = \begin{bmatrix} 7 & 6 & 9 \\ 4 & 5 & -4 \\ -7 & -3 & 8 \end{bmatrix} \quad b_4 = \begin{bmatrix} 22 \\ 5 \\ -2 \end{bmatrix}$$

- La soluzione è $x = [1, 1, 1]^\top$.
- Applicare i metodi iterativi di Jacobi e Gauss-Seidel usando la function `met_iter`.
- Riportare in una tabella per ciascuna matrice il numero di iterazioni, il raggio spettrale e il residuo relativo finale.

Function `met_iter`

Per applicare i metodi di Jacobi e Gauss-Seidel si usa il seguente comando

```
[x,iter,vectr]=met_iter(A,b,x0,toll,nmax,metodo,alfa,P)
```

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);
`metodo`	se metodo='J' Jacobi;
	se metodo='GS' Gauss-Seidel;
	se metodo='R' Richardson stazionario;
`alfa`	parametro del metodo di Richardson;
`P`	matrice di precondizionamento;
`x`	soluzione calcolata;
`iter`	numero di iterazioni eseguite;
`vectr`	vettore dei residui relativi.

Calcolo del raggio spettrale

Si ricorda che le matrici di iterazione dei metodi di Jacobi e di Gauss-Seidel sono date rispettivamente da:

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

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

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`.
`diag(A)` estrae il vettore `v` che contiene la diagonale principale.
`diag(v)` fornisce la matrice diagonale.

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

Per il calcolo del modulo degli autovalori usare il comando `abs`.

Esercizio 2

Si consideri il sistema lineare $Ax = b$ dove A è la matrice tridiagonale di dimensione $n = 10$ con elementi

$$a_{ii} = 3, \ a_{ii+1} = -2, \ a_{ii-1} = -1 \quad \text{per } i = 1, \ldots, n.$$

Il termine noto è scelto in modo che la soluzione sia il vettore $x \in \mathbb{R}^n$ che ha tutte le componenti pari a 1.
Porre `toll=1e-12, nmax=400` e risolvere il sistema con i metodi di Jacobi e Gauss-Seidel.
Confrontare il numero di iterazioni e l'accuratezza della soluzione.
Plottare in scala semilogaritmica il vettore dei residui relativi `vectr`.

Il metodo di Richardson stazionario

Posto $A = P - (P - A)$, un metodo iterativo nella forma

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

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

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

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

Convergenza del metodo di Richardson stazionario

$B_\alpha = I - \alpha P^{-1}A$ matrice di iterazione del metodo di Richardson stazionario.
P ed A matrici simmetriche e definite positive.
λ_i, u_i autovalori ed autovettori di $P^{-1}A$: $P^{-1}Au_i = \lambda_i u_i$.
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

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

Esercizio 3

Si consideri il sistema lineare $A_1 x = b_1$ dell'esercizio 1.
Verificare che il metodo di Jacobi o Gauss-Seidel non convergono.
Usare il metodo di Richardson con la matrice `P=tril(A)` e $\alpha = 0.2$.
Quante iterazioni sono richieste per la convergenza?

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

Fattorizzazione incompleta

La fattorizzazione incompleta si ottiene con i comandi:

- `ilu` per matrici generali;
- `ichol` per matrici simmetriche e definite positive.

`[L,U,P]=ilu(A)` fornisce i fattori L, U e P tali che $L * U$ fornisce la fattorizzazione incompleta di PA con lo stesso schema di sparsità della matrice A.
`L=ichol(A)` fornisce la matrice triangolare inferiore della fattorizzazione incompleta $L\,L^\top$ di A con lo stesso schema di sparsità della matrice A.

Per calcolare la fattorizzazione incompleta eliminando gli elementi che sono inferiori a una certa tolleranza `toll` si deve introdurre una struttura come segue:

- **ilu**

```
setup.droptol=toll;
[L,U]=ilu(A,setup);
```

- **ichol**

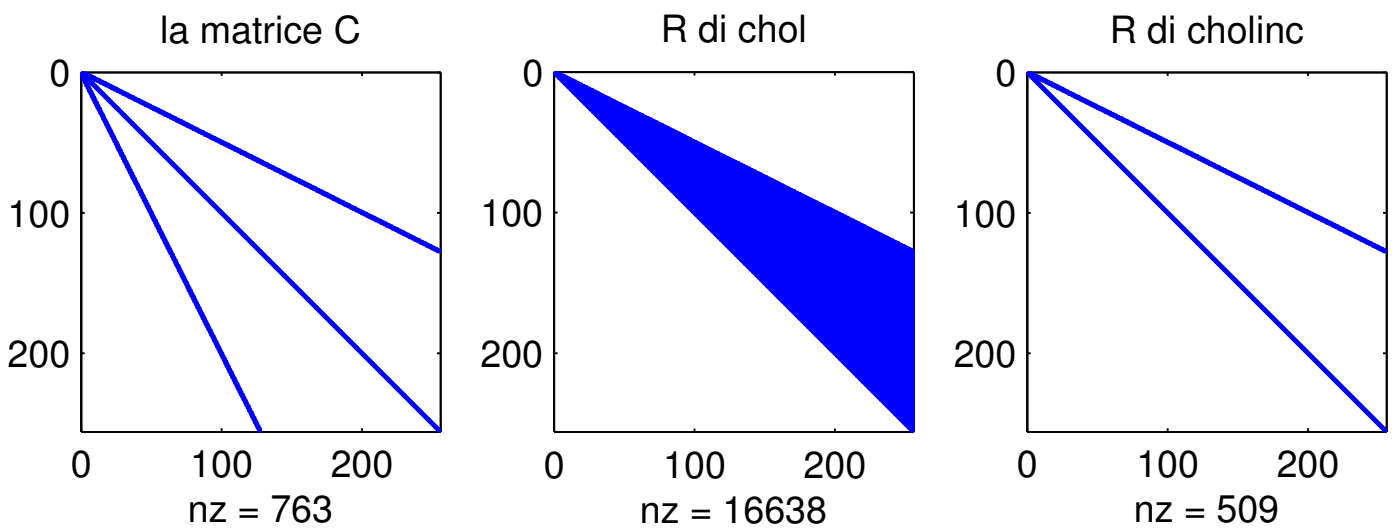
```
opt.droptol=toll;
L=ichol(A,opt);
```

Confronto risultati di `lu` e `ilu`

Il file di tipo script `cap.m` fornisce la costruzione della matrice simmetrica e definita positiva associata a una particolare rete idrica (porre `n=8`).

- Usare il comando `spy` per visualizzare la distribuzione degli elementi diversi da zero della matrice.
- Calcolare la fattorizzazione di Choleski di C usando il comando `chol` e visualizzare la matrice R con il comando `spy`.
- Calcolare la fattorizzazione incompleta di Choleski di C usando il comando `ichol` e visualizzare la matrice R con il comando `spy`.

Risultato


la matrice C
0
100
200
0
100
200
nz = 763
R di chol
0
100
200
0
100
200
nz = 16638
R di cholinc
0
100
200
0
100
200
nz = 509

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

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

Function `richardson`

La function `richardson` implementa il metodo di Richardson dinamico.

```
[x,iter,vectr]=richardson(A,b,x0,tol,nmax,P,P2)
```

INPUT

`A, b`	matrice e termine noto
`x0`	vettore iniziale
`tol`	tolleranza
`nmax`	numero massimo di iterazioni
`P`	matrice di precondizionamento
`P2`	viene definita se P e P2 danno la fattorizzazione LU di P

OUTPUT

`x`	soluzione
`iter`	numero di iterazioni eseguite
`vectr`	vettore dei residui.

Esercizio 4

Si consideri il sistema $Ax = b$ essendo $A \in \mathbb{R}^{n \times n}$ la matrice pentadiagonale che ha valore 4 sulla diagonale principale e valore -1 sulla prima e terza diagonale sopra e sotto quella principale. b viene scelto in modo che la soluzione sia $x = (1, \ldots, 1)^\top$.

- Posto P uguale alla matrice identità, si determinino le iterazioni necessarie per ottenere un errore con tolleranza pari a 10^{-5} utilizzando il metodo di Jacobi, il metodo di Richardson stazionario con `alfa=0.25` e il metodo di Richardson dinamico. Porre `tol=1e-5`, `nmax=5000`, `x0=zeros(n,1)`.
- Porre P uguale alla matrice tridiagonale che ha valori 2 sulla diagonale principale e -1 nelle diagonali sopra e sotto quella principale. Con gli stessi parametri di prima, si determinino le iterazioni necessarie per ottenere un errore con tolleranza pari a 10^{-5} utilizzando il metodo di Gauss-Seidel, il metodo di Richardson stazionario con `alfa=0.1821` e il metodo di Richardson dinamico.

Utilizzare $n <= 100$.

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

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

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 $z^{(k)}$ tali che:

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

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

Gradiente coniugato - Algoritmo

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

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))`).

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 5

Si consideri la matrice pentadiagonale $A \in \mathbb{R}n \times n$ che ha valore 4 sulla diagonale principale e valore -1 sulla prima e terza diagonale sopra e sotto la diagonale principale e il termine noto $b \in \mathbb{R}^n$ tale che la soluzione del sistema abbia tutte componenti pari a 1.

Risolvere il sistema lineare $Ax = b$ applicando i seguenti metodi (usare `toll=1.e-5` e `nmax=300`):

- metodo di Richardson stazionario precondizionato con la fattorizzazione incompleta ($\alpha = 1$);
- metodo di Richardson dinamico precondizionato con la fattorizzazione incompleta;
- metodo del gradiente;
- metodo del gradiente coniugato;
- metodo del gradiente coniugato precondizionato con la fattorizzazione incompleta;

Calcolare l'errore relativo e confrontare il numero di iterazioni richieste per $N_x = N_y = n = 10$.

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(2)<=restart` numero delle iterate interne.
- `resvec` vettore dei residui in norma (iterata per iterata)

Esempio

- Costruire la matrice A, il termine noto e risolvere con `gmres` con i seguenti comandi

```
load west0479
A = west0479
b = sum(A,2)
[x,flag] = gmres(A,b,5)
```

Si ottiene `flag=1` perchè `gmres` non converge nel numero di iterazioni prescritto per default.

- Usando la fattorizzazione incompleta

```
setup.droptol=1e-5;
[L1,U1] = ilu(A,setup);
[x1,flag1] = gmres(A,b,5,1e-6,5,L1,U1);
```

`flag1=2` perchè la matrice `U1` ha elementi nulli sulla diagonale principale.

Esempio (segue)

- Occorre usare `ilu` con $droptol = 1e-6$

```
setup.droptol=1e-6;
[L2,U2] = ilu(A,setup);
[x1,flag2] = gmres(A,b,5,1e-6,5,L2,U2);
```

- Testare il comportamento di `gmres` con diversi valori di restart: con `restart=2` si ha `flag=1`.
  Scegliere quindi le seguenti copie di valori per `restart` e `nmax`:

restart	nmax
3	5
4	5
6	3
8	3

- Plottare il residuo relativo in scala semilogaritmica per ciascun caso.

```
semilogy(resvec/norm(b),'o-')
```

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

- Costruire la matrice: `A=gallery('wilk',21);`
- Costruire il termine noto in modo che la soluzione sia il vettore di componenti tutte uguali a 1;
- Risolvere con `bicgstab` usando `toll=1e-12` e `nmax=15`.

Esercizio

- Costruire la matrice A con i seguenti comandi
  `load west0479;  A = west0479;`
- Costruire il termine noto in modo che la soluzione sia il vettore di componenti tutte uguali a 1;
- Usare `bicgstab` come segue (`tol=1e-12; nmax=20`):
  `[x0,fl0,rr0,it0,rv0] = bicgstab(A,b,tol,nmax);`
- Controllare i valori di `fl0` e `it0`;
- Plottare il residuo:
  `semilogy(0:0.5:nmax,rv0/norm(b),'-o');`
- Usare `ilu(A)` per costruire il precondizionatore;
- Usare `bicgstab` come segue:
  `[x1,fl1,rr1,it1,rv1] = bicgstab(A,b,tol,nmax,L,U);`
- Poiché U è singolare il metodo non converge. Riprovare usando `ilu(A,setup)` con `setup.droptol=1e-5` e `setup.droptol=1e-6`.
- Se `fl1=0` plottare il residuo:
  `semilogy(0:0.5:it1,rv1/norm(b),'-o');`