Metodi iterativi per la soluzione di sistemi lineari

Alvise Sommariva

Università degli Studi di Padova
Dipartimento di Matematica

8 aprile 2014

Metodi iterativi e diretti

Supponiamo che siano $A \in \mathbb{R}^{n \times n}$ (matrice non singolare), $b \in \mathbb{R}^n$ (vettore colonna) e di dover risolvere il problema $Ax = b$ avente sol. unica x^*.

A tal proposito si può utilizzare la **fatt. LU con pivoting**. Il costo computazionale è in generale di $O(n^3/3)$ operazioni moltiplicative. Questo diventa proibitivo se n è particolarmente elevato.

L'idea dei **metodi iterativi** è quello di ottenere una **successione di vettori** $x^{(k)} \to x^*$ **cosicchè per** $\bar{k} \ll n$ **sia** $x^{(\bar{k})} \approx x^*$.

In generale, la soluzione non è ottenuta esattamente come nei metodi diretti in un numero finito di operazioni (in aritmetica esatta), ma quale limite, accontentandosi di poche iterazioni ognuna dal costo quadratico. Quindi il costo totale sarà di ordine $O(\bar{k} \cdot n^2)$.

Metodi iterativi stazionari

Supponiamo che la matrice non singolare $A \in \mathbb{R}^{n \times n}$ sia tale che

$$A = M - N, \text{ con M non singolare.}$$

Allora $Mx - Nx = Ax = b$ e quindi $Mx = Nx + b$. Moltiplicando ambo i membri per M^{-1} e posto $\phi(x) = M^{-1}Nx + b$ abbiamo $x = M^{-1}Nx + M^{-1}b = \phi(x)$. Viene quindi naturale utilizzare la succ. del metodo di punto fisso

$$x^{(k+1)} = \phi(x^{(k)}) = M^{-1}Nx^{(k)} + M^{-1}b.$$

La matrice $P = M^{-1}N$ si dice di *iterazione* e non dipende, come pure b dall'indice di iterazione k. Per questo motivo tali metodi si chiamano **iterativi stazionari**.

Quale utile notazione, sia inoltre $A = D - E - F$ con D la matrice diagonale estratta da A, E, F rispettivamente triangolari inferiore e superiore.

Esempio A, E, F

```
>> A=[1 2 3 4; 5 6 7 2; 8 9 1 2; 3 4 5 1]
A =
      1      2      3      4
      5      6      7      2
      8      9      1      2
      3      4      5      1
>> E=-(tril(A)-diag(diag(A)))
E =
      0      0      0      0
     -5      0      0      0
     -8     -9      0      0
     -3     -4     -5      0
>> F=-(triu(A)-diag(diag(A)))
F =
      0     -2     -3     -4
      0      0     -7     -2
      0      0      0     -2
      0      0      0      0
>> % A=diag(diag(A))-E-F.
```

Metodo di Jacobi

Nel caso del **metodo di Jacobi** si ha

$$M = D, \;\; N = E + F \tag{1}$$

e quindi

$$\begin{aligned} P &= M^{-1}N = D^{-1}(E+F) = D^{-1}(D-D+E+F) \\ &= D^{-1}(D-A) = I - D^{-1}A \end{aligned} \tag{2}$$

Si osservi che se D è singolare allora il metodo di Jacobi, almeno in questa versione di base, non può essere utilizzato visto che in (5) non ha senso la scrittura D^{-1}.

Qualora sia $a_{ii} \neq 0$ per ogni $i = 1, \ldots, n$, il metodo di Jacobi può essere descritto come metodo delle sostituzioni simultanee

$$x_i^{(k+1)} = (b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)})/a_{ii}, \;\; i = 1, \ldots, n. \tag{3}$$

Metodo di Jacobi

Vediamo perchè questa derivazione. Da $Ax = b$, con $A \in \mathbb{R}^{n \times n}$ abbiamo

$$\sum_{j<i} a_{i,j} x_j + a_{i,i} x_i + \sum_{j>i} a_{i,j} x_j = \sum_j a_{i,j} x_j = b_i, \; i = 1, \ldots, n$$

e quindi evidenziando x_i, supposto $a_{i,i} \neq 0$ deduciamo

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

Quindi note delle approssimazioni della soluzione $x^{(k)} = \{x_j^{(k)}\}$ per $j = 1, \ldots, n$ è naturale introdurre il metodo iterativo

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

Metodo di Gauss-Seidel

Il **metodo di Gauss-Seidel** è definito quale metodo staz. in cui

$$M = D - E, N = F \tag{4}$$

e quindi

$$P = M^{-1}N = (D - E)^{-1}F \tag{5}$$

Similmente al metodo di Jacobi, possiamo riscrivere più semplicemente anche Gauss-Seidel come

$$x_i^{(k+1)} = \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) / a_{ii}. \tag{6}$$

Da (6) si capisce perchè tale metodo è noto anche come metodo delle sostituzioni successive.

Metodo di Gauss-Seidel

Vediamo perchè questa derivazione. Da $Ax = b$, con $A \in \mathbb{R}^{n \times n}$ abbiamo

$$\sum_{j<i} a_{i,j} x_j + a_{i,i} x_i + \sum_{j>i} a_{i,j} x_j = \sum_j a_{i,j} x_j = b_i, \ \ i = 1, \ldots, n$$

e quindi evidenziando x_i, supposto $a_{i,i} \neq 0$ deduciamo

$$x_i = \frac{1}{a_{i,i}} \left(b_i - \left(\sum_{j<i} a_{i,j} x_j + \sum_{j>i} a_{i,j} x_j \right) \right), \ \ i = 1, \ldots, n$$

Quindi note delle approssimazioni della soluzione $x^{(k)} = \{x_j^{(k)}\}$ per $j > i$, e già calcolate $x^{(k+1)} = \{x_j^{(k+1)}\}$ per $j < i$ è naturale introdurre il metodo iterativo

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

SOR

La versione di Gauss-Seidel con la scelta del parametro ω è nota in letteratura come **successive over relaxation** (**SOR**)

$$x^{(k+1)} = \left(\frac{D}{\omega} - E\right)^{-1} \left(\left(\frac{1}{\omega} - 1\right) D + F\right) x^{(k)} + \left(\frac{D}{\omega} - E\right)^{-1} b. \qquad (7)$$

dove D, E, F sono al solito tre matrici tali cha $A = D - E - F$ con D la matrice diagonale estratta da A, E, F rispettivamente triangolari inferiore e superiore.

SOR

Il metodo SOR deriva dallo descrivere esplicitamente una iter. di Gauss-Seidel

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

e sostituirla con la combinazione convessa

$$x_i^{(k+1)} = \omega \overline{x}_i^{(k+1)} + (1-\omega) x^{(k)}.$$

cioè

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

Gauss-Seidel è SOR per $\omega = 1$.

SOR

Notiamo che le iterazioni di SOR verificano l'uguaglianza

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

con

$$M = \frac{D}{\omega} - E, \;\; N = \left(\frac{1}{\omega} - 1\right) D + F$$

ed è

$$M - N = \left(\frac{D}{\omega} - E\right) - \left(\left(\frac{1}{\omega} - 1\right) D + F\right) = -E + D - F = A,$$

per cui SOR è un metodo iterativo stazionario.

I metodi di Richardson stazionari

Definizione

Fissato α, un metodo di **Richardson stazionario**, *con matrice di precondizionamento P, verifica*

$$P(x^{(k+1)} - x^{(k)}) = \alpha r^{(k)}. \tag{8}$$

dove

$$r^{(k)} = b - Ax^{(k)} \tag{9}$$

è il residuo alla k-sima iterazione.

I metodi di Richardson non stazionari

Definizione

Fissati α_k *dipendenti dalla iterazione* k*, un metodo di* **Richardson non stazionario***, con matrice di precondizionamento* P*, verifica*

$$P(x^{(k+1)} - x^{(k)}) = \alpha_k r^{(k)}. \qquad (10)$$

Si osservi che se $\alpha_k = \alpha$ per ogni k, allora il metodo di Richardson non stazionario diventa stazionario.

I metodi di Richardson

I metodi di Jacobi e di Gauss-Seidel, SOR, sono metodi iterativi del tipo

$$Mx^{(k+1)} = Nx^{(k)} + b, \tag{11}$$

per opportune scelte delle matrici M (che dev'essere invertibile), N tali che

$$A = M - N. \tag{12}$$

Essendo $r^{(k)} = b - Ax^{(k)}$,

$$M(x^{(k+1)} - x^{(k)}) = Nx^{(k)} + b - Mx^{(k)} = b - Ax^{(k)} = r^{(k)}. \tag{13}$$

I metodi di Richardson

Ne consegue che i metodi di Jacobi e di Gauss-Seidel, SOR, verificano

$$M(x^{(k+1)} - x^{(k)}) = r^{(k)} \tag{14}$$

In altri termini sono dei metodi di Richardson sono metodi di Richardson stazionari, con $\alpha = 1$ e matrice di precondizionamento $P = M$.

Per quanto riguarda i metodi di Richardson precondizionati e non stazionari, un classico esempio è il metodo del gradiente classico che vedremo in seguito.

Norma di matrici

Sia $\rho(A)$ il massimo degli autovalori in modulo della matrice A (il cosidetto **raggio spettrale**).

Sia $\|\cdot\| : \mathbb{R}^n \rightarrow \mathbb{R}_+$ una norma vettoriale. Definiamo **norma naturale** (in alcuni testi *norma indotta*) di una matrice $A \in R^{n \times n}$ la quantità

$$\|A\| := \sup_{x \in \mathbb{R}^n, x \neq 0} \frac{\|Ax\|}{\|x\|}.$$

Si nota subito che questa definizione coincide con quella di norma di un operatore lineare e continuo in spazi normati.

Norma di matrici

Vediamo alcuni esempi (cf. [4, p.24]). Sia x un arbitrario elemento di R^n, $A \in R^{n \times n}$.

- Si definisce $\|x\|_1 := \sum_{k=1}^{n} |x_k|$ e si dimostra che la norma naturale corrispondente è (cf. [4, p.26])

$$\|A\|_1 = \max_j \sum_{i=1}^{n} |a_{i,j}|.$$

- Si definisce $\|x\|_\infty := \max_k |x_k|$ e si dimostra che la norma naturale corrispondente è (cf. [4, p.26])

$$\|A\|_\infty = \max_i \sum_{j=1}^{n} |a_{i,j}|.$$

- Si definisce $\|x\|_2 := \left(\sum_{k=1}^{n} |x_k|^2\right)^{1/2}$ e si dimostra che la norma naturale corrispondente è (cf. [4, p.27])

$$\|A\|_2 = \rho^{1/2}(A^T A).$$

Per quanto riguarda un esempio chiarificatore in Matlab/Octave

```
>> A=[1 5; 7 13]
A =
      1     5
      7    13
>> norm(A,1)
ans =
     18
>> norm(A,inf)
ans =
     20
>> norm(A,2)
ans =
    15.5563
>> eig(A*A')
ans =
      2
    242
>> sqrt(242)
ans =
    15.5563
>> raggio_spettrale_A=max(abs(eig(A)))
raggio_spettrale_A =
    15.4261
>>
```

Norma di matrici

Si dimostra che (cf. [4, p.28])

Teorema

Per ogni norma naturale $\|\cdot\|$ e ogni matrice quadrata A si ha $\rho(A) \leq \|A\|$. Inoltre per ogni matrice A di ordine n e per ogni $\epsilon > 0$ esiste una norma naturale $\|\cdot\|$ tale che

$$\rho(A) \leq \|A\| \leq \rho(A) + \epsilon.$$

e inoltre (cf. [4, p.29], [3, p.232])

Teorema

Fissata una norma naturale $\|\cdot\|$, i seguenti asserti sono equivalenti

1. $A^m \to 0$;
2. $\|A^m\| \to 0$;
3. $\rho(A) < 1$.

Sul raggio spettrale

Ricordiamo che il raggio spettrale

$$\rho(A) = \max_k(|\lambda_k|)$$

(con $\{\lambda_k\}_{k=1,\ldots,n}$ autovalori della matrice $A \in \mathbb{R}^{n \times n}$) **non è una norma**.
Infatti la matrice

$$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

ha raggio spettrale nullo, ma non è la matrice nulla. Osserviamo che dagli esempi il raggio spettrale di una matrice A **non coincide in generale con la norma 1, 2, ∞**, ma che a volte $\rho(A) = \|A\|_2$ come nel caso di una matrice diagonale A (essendo gli autovalori di una matrice diagonale, proprio i suoi elementi diagonali).

Teorema di Hensel

Teorema

Se P è diagonalizzabile allora un metodo iterativo stazionario consistente $x^{(k+1)} = Px^{(k)} + c$ converge per ogni vettore iniziale x_0 se e solo se $\rho(P) < 1$.

Dimostrazione.

Ricordiamo che $A \in \mathbb{R}^{n \times n}$ è diagonalizzabile se e solo $\mathbb{R}^n$ possiede una base composta di autovettori di A (cf. [3, p.57]).
Consideriamo un metodo iterativo stazionario $x^{(k+1)} = Px^{(k)} + c$ in cui scelto $x^{(0)}$ si abbia

$$x^{(0)} - x^* = \sum_{s=1}^{n} c_s u_s$$

dove $\{u_k\}_k$ è una base di autovettori di P avente autovalori $\{\lambda_k\}_k$.

Teorema di Hensel

Dimostrazione.

Supponiamo $|\lambda_s| < 1$ per $s = 1, \ldots, n$.

Se il metodo è *consistente*, cioè

$$x^* = Px^* + c$$

abbiamo

$$\begin{aligned} x^{(k)} - x^* &= x^{(k-1)} + c - (Px^* + c) \\ &= P(x^{(k-1)} - x^*) = \ldots = P^k(x^0 - x^* \\ &= \sum_{s=1}^{n} c_s P^k u_s \\ &= \sum_{s=1}^{n} c_s \lambda_s^k u_s \end{aligned} \tag{15}$$

□

Teorema di Hensel

Dimostrazione.

Quindi se $|\lambda_s^k| < 1$ per ogni $s = 1, \ldots, n$ e $k = 1, 2, \ldots$, abbiamo

$$\|x^{(k)} - x^*\| = \|\sum_{s=1}^{n} c_s \lambda_s^k u_s\| \leq \sum_{s=1}^{n} |c_s||\lambda_s^k|\|u_s\| \to 0.$$

Se invece per qualche k si ha $|\lambda^k| \geq 1$ e $c_k \neq 0$ allora $\|x^{(k)} - x^*\|$ non converge a 0 al crescere di k.

Infatti, se $\lambda_l \geq 1$ è l'autovalore di massimo modulo, posto

$$x^{(0)} - x^* = u_l$$

abbiamo da (15)

$$\|x^{(k)} - x^*\| = \|\lambda_l^k u_l\| = |\lambda_l^k|\|u_l\|$$

convergente per qualsiasi scelta del vettore $x^{(0)}$. □

Teorema di Hensel

Dimostriamo ora una sua generalizzazione, scoperta da Hensel nel 1926.

Teorema

[Hensel] *Un metodo iterativo stazionario consistente* $x^{(k+1)} = Px^{(k)} + c$ *converge per ogni vettore iniziale* x_0 *se e solo se* $\rho(P) < 1$.

Si noti che il teorema riguarda la convergenza **per ogni vettore iniziale** x_0 ed è quindi di convergenza globale. Inoltre non si richiede che la matrice P sia diagonalizzabile.

Teorema di Hensel

Dimostrazione.

La dimostrazione è tratta da [3, p.236].
$\Leftarrow$ *Se* $\rho(P) < 1$, *allora* $x = Px + c$ *ha una e una sola sol.* x^*.
Infatti,

$$x = Px + c \Leftrightarrow (I - P)x = c$$

e la matrice $I - P$ *ha autovalori* $1 - \lambda_k$ *con* $k = 1, \ldots, n$ *tali che*

$$0 < |1 - |\lambda_k|_{\mathbb{C}}|_{\mathbb{R}} \leq |1 - \lambda_k|_{\mathbb{C}},$$

poichè $|\lambda_k|_{\mathbb{C}} \leq \rho(P) < 1$ *e quindi*

$$\det(I - P) = \prod_{k=1}^{n} (1 - \lambda_k) \neq 0,$$

per cui la matrice $I - P$ *è invertibile e il sistema* $(I - P)x = c$ *ha una e una sola soluzione* x^*.

Teorema di Hensel

Dimostrazione.

Sia $e(k) = x^{(k)} - x^*$. Come stabilito dal Teorema 0.1, sia inoltre una norma naturale $\|\cdot\|$ tale che

$$\rho(P) \leq \|P\| = \rho(P) + (1 - \rho(P))/2 < 1.$$

Essendo $x^{(k+1)} = Px^{(k)} + c$ e $x^* = Px^* + c$, sottraendo membro a membro le equazioni si ottiene

$$e^{(k+1)} = Pe^{(k)} = P^k e^{(0)}$$

da cui essendo $\|\cdot\|$ una norma naturale

$$\|e^{(k+1)}\| = \|Pe^{(k)}\| = \|P^k e^{(0)}\| \leq \|P^k\| \|e^{(0)}\|. \qquad (16)$$

Poichè il raggio spettrale è minore di 1 dal Teorema 0.2 abbiamo che $\|P^k\| \rightarrow 0$ da cui per (16) necessariamente $\|e^{(k+1)}\| \rightarrow 0$ e quindi per le proprietà delle norme $e^{(k+1)} \rightarrow 0$ cioè $x^{(k)} \rightarrow x^*$.

Teorema di Hensel

Dimostrazione.

$\Rightarrow$ Supponiamo che la successione $x^{(k+1)} = Px^{(k)} + c$ converga a x^* per qualsiasi $x^{(0)} \in \mathbb{R}^n$ ma che sia $\rho(P) \geq 1$. Sia $\lambda_{\max}$ il massimo autovalore in modulo di P e scegliamo $x^{(0)}$ tale che $e^{(0)} = x^{(0)} - x^*$ sia autovettore di P relativamente all'autovalore $\lambda_{\max}$. Essendo $Pe^{(0)} = \lambda_{\max} e^{(0)}$ e $e^{(k+1)} = P^k e^{(0)}$ abbiamo che

$$e^{(k+1)} = \lambda_{\max}^k e^{(0)}$$

da cui, qualsiasi sia la norma $\|\cdot\|$, per ogni $k = 1, 2, \ldots$ si ha

$$\|e^{(k+1)}\| = |\lambda_{\max}^k|_{\mathbb{C}} \|e^{(0)}\| \geq \|e^{(0)}\|$$

il che comporta che la successione non è convergente (altrimenti per qualche k sarebbe $e^{(k)} < e^{(0)}$). $\square$

Velocità di convergenza

L'analisi che faremo in questa sezione non è rigorosamente matematica, ciò nonostante permette di capire il legame tra il raggio spettrale della matrice di iterazione P e la riduzione dell'errore.

Si vede facilmente che se intendiamo calcolare x^* tale che $Ax^* = b$ con un metodo stazionario $x^{(k+1)} = Px^{(k)} + c$, posto $e^{(k)} = x^{(k)} - x^*$ si ha, supposto il metodo stazionario sia consistente, cioè $x^* = Px^* + c$

$$\begin{aligned} e^{(k)} &= x^{(k)} - x^* \\ &= (Px^{(k-1)} + c) - (Px^* + c) \\ &= Pe^{(k-1)} = \ldots = P^k \|e^{(0)}\| \end{aligned} \tag{17}$$

e quindi

$$\|e^{(k)}\| \leq \|P^k\| \|e^{(0)}\|, \tag{18}$$

Velocità di convergenza

Se $e^{(k-1)} \neq 0$, la quantità $\|e^{(k)}\|/\|e^{(k-1)}\|$ esprime la riduzione dell'errore al k-simo passo e

$$\sigma_k = \left(\frac{\|e^{(k)}\|}{\|e^{(k-1)}\|} \cdots \frac{\|e^{(1)}\|}{\|e^{(0)}\|} \right)^{\frac{1}{k}}$$

la riduzione media per passo dell'errore relativo ai primi k passi (cf. [3, p.239]).

Velocità di convergenza

Si dimostra che

Teorema

Sia $A \in \mathbb{C}^{n \times n}$ e $\|\cdot\|$ una norma naturale. Allora

$$\lim_k \|A^k\|^{\frac{1}{k}} = \rho(A)$$

Quindi per k sufficientemente grande si ha

$$\|P^k\|^{1/k} \approx \rho(P).$$

Sotto queste ipotesi, se $\|e^{(k+m)}\| \approx \|P^m\| \|e^{(k)}\|$ abbiamo

$$\|e^{(k+m)}\| \approx \|P^m\| \|e^{(k)}\| \approx \rho^m(P) \|e^{(k)}\| \qquad (19)$$

Velocità di convergenza

per cui affinchè

$$\|e^{(k+m)}\|/\|e^{(k)}\| \approx \rho^m(P) \approx \epsilon$$

applicando il logaritmo naturale ad ambo i membri, si vede serve sia,

$$m \log(\rho(P)) \approx \log \epsilon$$

e quindi

$$m \approx \frac{\log \epsilon}{\log(\rho(P))}.$$

Velocità di convergenza

Se

$$R(P) = -\log(\rho(P))$$

è la cosidetta **velocità di convergenza asintotica** del metodo iterativo relativo a P, si può così stimare che il numero di iterazioni m necessarie per ridurre l'errore di un fattore ϵ relativamente alla k-sima iterazione, cioè affinchè

$$\|e^{(k+m)}\|/\|e^{(k)}\| = \epsilon.$$

Si vede facilmente che è circa

$$m \approx \left\lceil \frac{-\log(\epsilon)}{R(P)} \right\rceil.$$

Velocità di convergenza

Conseguentemente, da

$$m \approx \left\lceil \frac{-\log(\epsilon)}{R(P)} \right\rceil$$

e

$$R(P) = -\log(\rho(P))$$

se P è la matrice d'iterazione di un metodo stazionario convergente (e consistente), essendo $\rho(P) < 1$, minore è $\rho(P)$ necessariamente è maggiore $R(P)$ e si può *stimare* il numero di iterazioni per ridurre l'errore di un fattore. ϵ.

Si desidera quindi cercare metodi con $\rho(P)$ più piccolo possibile.

Alcuni teoremi di convergenza

Definizione

Una matrice A si dice **tridiagonale** *se* $A_{i,j} = 0$ *qualora* $|i - j| > 1$.

Esempio:

$$\begin{pmatrix} 1 & 2 & 0 & 0 & 0 \\ 1 & 5 & 7 & 0 & 0 \\ 0 & 2 & 0 & 9 & 0 \\ 0 & 0 & 4 & 4 & 2 \\ 0 & 0 & 0 & 5 & 3 \end{pmatrix}$$

Alcuni teoremi di convergenza

Teorema

Per matrici tridiagonali $A = (a_{i,j})$ con componenti diagonali non nulle, i metodi di Jacobi e Gauss-Seidel sono o entrambi convergenti o divergenti e il tasso di convergenza del metodo di Gauss-Seidel è il doppio di quello del metodo di Jacobi.

Il che vuol dire che *asintoticamente* sono necessarie metà iterazioni del metodo di Gauss-Seidel per ottenere la stessa precisione del metodo di Jacobi.

Nota.

Il teorema si estende a matrici tridiagonali a blocchi, con blocchi diagonali non singolari.

Alcune definizioni

Definizione

La matrice A è a **predominanza diagonale** *(per righe) se per ogni $i = 1, \ldots, n$ risulta*

$$|a_{i,i}| \geq \sum_{j=1, j \neq i}^{n} |a_{i,j}|$$

e per almeno un indice s si abbia

$$|a_{s,s}| > \sum_{j=1, j \neq s}^{n} |a_{s,j}|.$$

La matrice A è a predominanza diagonale per colonne se A^T a predominanza diagonale per righe.

Alcune definizioni

Definizione

Se

$$|a_{s,s}| > \sum_{j=1, j \neq s}^{n} |a_{s,j}|, \; s = 1, \ldots, n$$

allora la matrice A si dice a **predominanza diagonale (per righe) in senso stretto**.

La matrice A è a predominanza diagonale per colonne in senso stretto se A^T a predominanza diagonale per righe in senso stretto.

Ad esempio la matrice

$$A = \begin{pmatrix} 4 & -4 & 0 \\ -1 & 4 & -1 \\ 0 & -4 & 4 \end{pmatrix}$$

è a predominanza diagonale per righe (non stretta).

Teoremi di convergenza

Teorema

Sia A una matrice quadrata a predominanza diagonale per righe in senso stretto. Allora il metodo di Jacobi converge alla soluzione di $Ax = b$, qualsiasi sia il punto $x^{(0)}$ iniziale.

Teorema

Sia A è a predominanza diagonale per righe in senso stretto. Allora il metodo di Gauss-Seidel converge, qualsiasi sia il punto $x^{(0)}$ iniziale.

Tali teoremi valgono anche se A è a predominanza diagonale per colonne in senso stretto.

Alcune definizioni

La matrice A è simmetrica se $A = A^T$. Una matrice A è definita positiva se ha tutti gli autovalori positivi. Ad esempio la matrice

$$A = \begin{pmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{pmatrix}$$

è simmetrica e definita positiva:

```
>> A=[4 -1 0; -1 4 -1; 0 -1 4];
>> eig(A) % AUTOVALORI DI A.
ans =
     2.5858
     4.0000
     5.4142
>>
```

Alcuni teoremi di convergenza

Ricordiamo che equivalentemente una matrice A è definita positiva se

$$x^T Ax > 0, \text{ per ogni } x \neq 0.$$

Teorema

Sia A simmetrica con elementi diagonali positivi. Allora il metodo di Gauss-Seidel converge se e solo A è definita positiva.

Più in generale,

Teorema

Sia A simmetrica con elementi diagonali positivi. Allora il metodo SOR converge se e solo se $0 < w < 2$ *e A è definita positiva.*

Per una dimostrazione si veda [6, p.215].

Test di arresto

Consideriamo il sistema lineare $Ax = b$ avente un'unica soluzione x^* e supponiamo di risolverlo numericamente con un metodo iterativo stazionario del tipo

$$x^{(k+1)} = Px^{(k)} + c,$$

che sia *consistente* cioè

$$x^* = Px^* + c.$$

Desideriamo introdurre un test di arresto che interrompa le iterazioni, qualora una certa quantità relativa al sistema lineare $Ax = b$ e alle iterazioni eseguite, sia al di sotto di una tolleranza $\epsilon > 0$ fissata dall'utente.

Test di arresto: criterio dello step

Posto $\Delta^{(k)} := x^{(k+1)} - x^{(k)}$ e $e^{(k)} = x^* - x^{(k)}$, essendo

$$\begin{aligned} e^{(k)} &= x^* - x^{(k)} = (Px^* + c) - (Px^{(k-1)} + c) \\ &= P(x^* - x^{(k-1)}) = Pe^{(k-1)} \end{aligned} \tag{20}$$

abbiamo

$$\begin{aligned} \|e^{(k)}\|_2 &= \|x^* - x^{(k)}\|_2 = \|(x^* - x^{(k+1)}) + (x^{(k+1)} - x^{(k)})\|_2 \\ &= \|e^{(k+1)} + \Delta^{(k)}\|_2 = \|Pe^{(k)} + \Delta^{(k)}\|_2 \\ &\leq \|P\|_2 \cdot \|e^{(k)}\|_2 + \|\Delta^{(k)}\|_2 \end{aligned} \tag{21}$$

Test di arresto: criterio dello step

Fissata dall'utente una tolleranza *tol*, si desidera interrompere il processo iterativo quando $\|x^* - x^{(k)}\| \leq \text{tol}$.

Non disponendo di x^*, il **criterio dello step**, consiste nell'interrompere il metodo iterativo alla $k+1$-sima iterazione qualora $\|x^{(k+1)} - x^{(k)}\| \leq \text{tol}$.

Di seguito desideriamo vedere quando tale criterio risulti attendibile cioè

$$|x^{(k+1)} - x^{(k)}| \approx |x^* - x^{(k)}|$$

Se P è simmetrica, allora esistono una matrice ortogonale U, cioè tale che $U^T = U^{-1}$, e una matrice diagonale a coefficienti reali Λ per cui

$$P = U \Lambda U^T$$

ed essendo P e Λ simili hanno gli stessi autovalori $\{\lambda_k\}_k$.

Test di arresto: criterio dello step

Di conseguenza, se P è simmetrica

$$\begin{aligned} \|P\|_2 &= \sqrt{\rho(PP^T)} = \sqrt{\rho(U\Lambda U^T(U\Lambda U^T)^T)} \\ &= \sqrt{\rho(U\Lambda^2 U^T)} \end{aligned} \tag{22}$$

Essendo $U\Lambda^2 U^T$ simile a Λ^2, $U\Lambda^2 U^T$ e Λ^2 hanno gli stessi autovalori uguali a $\{\lambda_k^2\}_k$ e di conseguenza lo stesso raggio spettrale, da cui

$$\rho(U\Lambda^2 U^T) = \rho(\Lambda^2)$$

e quindi ricaviamo

$$\begin{aligned} \|P\|_2 &= \sqrt{\rho(\Lambda^2)} = \sqrt{\max_k |\lambda_k^2|} \\ &= \sqrt{(\max_k |\lambda_k|^2)} = \sqrt{(\max_k |\lambda_k|)^2} \\ &= \max_k |\lambda_k| = \rho(P) \end{aligned} \tag{23}$$

Test di arresto: criterio dello step

Quindi da (21)

$$\begin{aligned} \|e^{(k)}\|_2 &\leq \|P\|_2 \cdot \|e^{(k)}\|_2 + \|\Delta^{(k)}\|_2 \\ &= \rho(P) \cdot \|e^{(k)}\|_2 + \|\Delta^{(k)}\|_2 \end{aligned} \qquad (24)$$

e se $\rho(P) < 1$, cioè il metodo iterativo stazionario converge per qualsiasi scelta del vettore iniziale, portando $\rho(P) \cdot \|e^{(k)}\|_2$ a primo membro e dividendo per $1 - \rho(P)$ deduciamo

$$\|x^* - x^{(k)}\|_2 = \|e^{(k)}\|_2 =\leq \frac{\|\Delta^{(k)}\|_2}{1-\rho(P)} = \frac{\|x^{(k+1)} - x^{(k)}\|_2}{1-\rho(P)}$$

da cui se P è simmetrica allora il criterio dello step è affidabile se $\rho(P)$ è piccolo.

Test di arresto: criterio del residuo

Si definisce **residuo** alla k-sima iterazione la quantità

$$r^{(k)} := b - Ax^{(k)}$$

ed essendo $b = Ax^*$ abbiamo per $e^{(k)} = x^* - x^{(k)}$

$$b - Ax^{(k)} = Ax^* - Ax^{(k)} = A(x^* - x^{(k)}) = Ae^{(k)}$$

da cui

$$r^{(k)} = Ae^{(k)}.$$

Interromperemo il processo iterativo quando

$$\|r^{(k)}\| \leq \text{tol},$$

desiderando pure

$$\frac{\|x^{(k)} - x^*\|}{\|x^*\|} \leq \text{tol}$$

Test di arresto: criterio del residuo

Osserviamo che

- essendo A invertibile e $r^{(k)} = Ae^{(k)}$ ricaviamo $e^{(k)} = A^{-1}r^{(k)}$ da cui
$$\|e^{(k)}\| = \|A^{-1}r^{(k)}\| \leq \|A^{-1}\| \|r^{(k)}\|;$$
- poichè $b = Ax^*$ abbiamo $\|b\| \leq \|A\| \|x^*\|$ e quindi
$$\frac{1}{\|x^*\|} \leq \frac{\|A\|}{\|b\|}.$$

Test di arresto: criterio del residuo

Di conseguenza, denotato con $\kappa(A) = \|A\| \|A^{-1}\|$ il numero di condizionamento (necessariamente maggiore o uguale a 1), se $x^* \neq 0$ abbiamo

$$\frac{\|e^{(k)}\|}{\|x^*\|} \leq \frac{\|A\|}{\|b\|} \|e^{(k)}\| \leq \frac{\|A\|}{\|b\|} \cdot \|A^{-1}\| \|r^{(k)}\| \leq \kappa(A) \frac{\|r^{(k)}\|}{\|b\|}$$

Quindi

$$\frac{\|x^{(k)} - x^*\|}{\|x^*\|} = \frac{\|e^{(k)}\|}{\|x^*\|} \leq \kappa(A) \frac{\|r^{(k)}\|}{\|b\|}.$$

Il criterio d'arresto $\frac{\|r^{(k)}\|}{\|b\|} \leq$ tol è quindi inappropriato quando $\kappa(A) \gg 1$ in quanto non implica necessariamente che

$$\frac{\|x^{(k)} - x^*\|}{\|x^*\|} \leq \text{tol}.$$

I metodi di discesa

Sia A **una matrice simmetrica definita positiva**.

Si osserva che se x^* è l'unica soluzione di $Ax = b$ allora è pure il minimo del funzionale dell'energia

$$\phi(x) = \frac{1}{2} x^T A x - b^T x, \quad x \in \mathbb{R}^n$$

in quanto

$$\mathrm{grad}(\phi(x)) = Ax - b = 0 \Leftrightarrow Ax = b.$$

Quindi invece di calcolare la soluzione del sistema lineare, intendiamo calcolare il minimo del funzionale ϕ.

I metodi di discesa

Un generico **metodo di discesa** consiste nel generare una successione

$$x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)}$$

dove $p^{(k)}$ è una direzione fissata secondo qualche criterio. Lo scopo ovviamente è che

$$\phi(x^{(k+1)}) < \phi(x^{(k)}),$$

e che il punto x^*, in cui si ha il minimo di ϕ, venga calcolato rapidamente.

I metodi di discesa

Il metodo del **gradiente classico** consiste nello scegliere α_k e $p^{(k)}$ così da ottenere la massima riduzione del funzionale dell'energia a partire dal punto $x^{(k)}$.

Differenziando

$$\phi(x) = \frac{1}{2}x^T Ax - b^T x, \;\; x \in \mathbb{R}^n$$

si vede che tale scelta coincide con lo scegliere

$$\begin{aligned} p^{(k)} &= r^{(k)} \\ \alpha_k &= \frac{\|r^{(k)}\|_2^2}{(r^{(k)})^T A r^{(k)}}. \end{aligned} \tag{25}$$

Essendo $x^{(k+1)} = x^{(k)} + \alpha_k r^{(k)}$ è un metodo di Richardson non stazionario con $P = I$ e parametro α_k definito da (25).

Il metodo del gradiente coniugato (1952).

Supponiamo di dover risolvere il sistema lineare $Ax = b$. Con il termine residuo in $x^{(k)}$ si intende la quantità $r^{(k)} = b - Ax^{(k)}$. La succ. delle iterazioni del gradiente coniugato è quella propria dei metodi di discesa,

$$x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)}, \quad \alpha_k = \frac{(r^{(k)})^T r^{(k)}}{(p^{(k)})^T A p^{(k)}}$$

dove $p^{(0)} = r^{(0)}$ e

$$p^{(k)} = r^{(k)} + \beta_k p^{(k-1)}, \quad \beta_k = \frac{(r^{(k)})^T r^{(k)}}{(r^{(k-1)})^T r^{(k-1)}}.$$

Con questa scelta si prova che $p^{(k)}$ e $p^{(k-1)}$ sono *A-coniugati*.

$$(p^{(k)})^T A p^{(k-1)} = 0.$$

Il metodo del gradiente coniugato: proprietà

Il metodo del gradiente coniugato ha molte proprietà particolari.
Ne citiamo alcune.

Teorema

Se A è una matrice simmetrica e definita positiva di ordine n, allora il metodo del gradiente coniugato è convergente e fornisce in aritmetica esatta la soluzione del sistema $Ax = b$ in al massimo n iterazioni.

Questo teorema tradisce un po' le attese, sia perchè in generale i calcoli non sono compiuti in aritmetica esatta, sia perchè in molti casi della modellistica matematica n risulta essere molto alto.

Il metodo del gradiente coniugato: proprietà

Definizione

Lo spazio

$$\mathcal{K}_k = span(r^{(0)}, Ar^{(0)}, \ldots, A^{k-1}r^{(0)})$$

per $k \geq 1$ *si dice* **spazio di Krylov**.

Teorema

Sia

$$\mathcal{K}_k = span(r^{(0)}, Ar^{(0)}, \ldots, A^{k-1}r^{(0)})$$

per $k \geq 1$. *Allora la k-sima iterata dal metodo del gradiente coniugato, minimizza il funzionale* ϕ *nell'insieme* $x^{(0)} + \mathcal{K}_k$.

Per una dimostrazione si veda [7, p.12]. Si osservi che essendo la k-sima iterazione del gradiente classico pure in $x^{(0)} + \mathcal{K}_k$, il gradiente classico non minimizza in generale il funzionale ϕ in $x^{(0)} + \mathcal{K}_k$.

Il metodo del gradiente coniugato: proprietà

Si può dimostrare che se A è simmetrica e definita positiva,

$$\|x\|_A = \sqrt{x^T A x}$$

e

$$e_k = x^* - x^{(k)}$$

allora

$$\|e_k\|_A \leq \left(\frac{\sqrt{K_2(A)} - 1}{\sqrt{K_2(A)} + 1} \right)^{2k} \|e_0\|_A.$$

Questo risultato stabilisce che la convergenza del gradiente coniugato è lenta qualora si abbiano alti numeri di condizionamento

$$K_2(A) := \|A\|_2 \|A^{-1}\|_2 = \frac{\max_i |\lambda_i|}{\min_j |\lambda_j|}$$

(ove al solito $\{\lambda_i\}$ sono gli autovalori di A).

Metodo di Jacobi in Matlab

Un codice gratuito del metodo di Jacobi, è `jacobi.m` tratto da Netlib:

```
function [x,error,iter,flag]=jacobi(A,x,b,max_it,tol)
% input
% A, REAL matrix
% x, REAL initial guess vector
% b, REAL right hand side vector
% max_it, INTEGER maximum number of iterations
% tol, REAL error tolerance
%
% output
% x, REAL solution vector
% error, REAL error norm
% iter, INTEGER number of iterations performed
% flag, INTEGER: 0 = solution found to tolerance
%   1 = no convergence given max_it
```

Nota.

Osserviamo che `jacobi` *inizialmente usa un test del residuo (relativo), per poi usarne uno basato sullo step (relativo).*

Metodo di Jacobi in Matlab

```
iter = 0; % initialization
flag = 0;
bnrm2 = norm( b );
if  ( bnrm2 == 0.0 ), bnrm2 = 1.0; end
r = b - A*x;
error = norm( r ) / bnrm2;
if ( error < tol ) return, end
[m,n]=size(A);
[ M, N ] = split( A , b, 1.0, 1 ); % matrix splitting
for iter = 1:max_it, % begin iteration
  x_1 = x;
  x   = M \ (N*x + b); % update approximation
  error = norm( x - x_1 ) / norm( x ); % compute error
  if ( error <= tol ), break, end % check convergence
end
if ( error > tol ) flag = 1; end % no convergence
```

Metodo di SOR in Matlab

```
function [x, error, iter, flag] = sor(A, x, b, w, max_it, tol)
% sor.m solves the linear system Ax=b using the
% Successive Over-Relaxation Method (Gauss-Seidel method when omega = 1 ).
% input
% A, REAL matrix
% x REAL initial guess vector
% b REAL right hand side vector
% w REAL relaxation scalar
% max_it  INTEGER maximum number of iterations
% tol REAL error tolerance
% output
% x REAL solution vector
% error REAL error norm
 % iter INTEGER number of iterations performed
%  flag INTEGER: 0 = solution found to tolerance
%                          1 = no convergence given max_it
```

Metodo di SOR in Matlab

```
flag = 0; % initialization
iter = 0;
bnrm2 = norm( b );
if  ( bnrm2 == 0.0 ), bnrm2 = 1.0; end
r = b - A*x;
error = norm( r ) / bnrm2;
if ( error < tol ) return, end
[ M, N, b ] = split( A, b, w, 2 );  % matrix splitting
for iter = 1:max_it % begin iteration
 x_1 = x;
 x   = M \ ( N*x + b ); % update approximation
 error = norm( x - x_1 ) / norm( x ); % compute error
 if ( error <= tol ), break, end % check convergence
end
b = b / w; % restore rhs
 if ( error > tol ) flag = 1; end; % no convergence
```

Routine split

La routine `split` chiamata da `jacobi` e `sor`, tratta da Netlib

```
function [ M, N, b ] = split( A, b, w, flag )
% split.m sets up the matrix splitting for the stat.
% iterative methods:jacobi and sor (gauss-seidel, w=1)
% input
% A DOUBLE PRECISION matrix
% b DOUBLE PRECISION right hand side vector (for SOR)
% w DOUBLE PRECISION relaxation scalar
% flag INTEGER flag for method: 1 = jacobi 2 = sor.
% output
% M DOUBLE PRECISION matrix
% N DOUBLE PRECISION matrix such that A = M - N
% b DOUBLE PRECISION rhs vector ( altered for SOR )
[m,n] = size( A );
if ( flag == 1 ), % jacobi splitting
  M = diag(diag(A)); N = diag(diag(A)) - A;
elseif ( flag == 2 ),  % sor/gauss-seidel splitting
  b = w * b;
  M =  w * tril( A, -1 ) + diag(diag( A ));
  N = -w * triu( A,  1 ) + ( 1.0 - w ) * diag(diag(A));
end;
```

Gradiente coniugato in Matlab

Un codice gratuito del Grad. Coniugato, è `cg.m` tratto da Netlib:

```
function [x,error,iter,flag]=cg(A,x,b,M,max_it,tol)
flag = 0;  iter = 0;  bnrm2 = norm( b );
if  ( bnrm2 == 0.0 ), bnrm2 = 1.0; end
r = b - A*x; error = norm( r ) / bnrm2;
if ( error < tol ) return, end
for iter = 1:max_it
  z  = M \ r; rho = (r'*z);
  if ( iter > 1 )
    beta = rho / rho_1; p = z + beta*p;
  else
    p = z;
  end
  q = A*p; alpha = rho / (p'*q ); x = x + alpha * p;
  r = r - alpha*q;  error = norm( r ) / bnrm2;
  if ( error <= tol ), break, end
  rho_1 = rho;
end
if ( error > tol ) flag = 1; end
```

Esercizio in Matlab: matrice di Poisson

Consideriamo il sistema lineare $Ax = b$ dove A è la matrice tridiagonale a blocchi (di Poisson)

$$A = \begin{pmatrix} B & -I & 0 & \dots & 0 \\ -I & B & -I & \dots & 0 \\ 0 & -I & B & \dots & \dots \\ 0 & \dots & \dots & \dots & -I \\ 0 & 0 & \dots & -I & B \end{pmatrix}$$

con

$$B = \begin{pmatrix} 4 & -1 & 0 & \dots & 0 \\ -1 & 4 & -1 & \dots & 0 \\ 0 & -1 & 4 & \dots & \dots \\ 0 & \dots & \dots & \dots & -1 \\ 0 & 0 & \dots & -1 & 4 \end{pmatrix}$$

Esercizio in Matlab: matrice di Poisson

La matrice A è facilmente disponibile, con il comando `gallery` di Matlab. Vediamo un esempio:

```
>> A=gallery('poisson',3); % A  sparse.
>> A=full(A); % A piena.
>> A
A =
     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
>>
```

Esercizio in Matlab: matrice di Poisson

Evidentemente A è una matrice di Poisson con B matrice quadrata di ordine 3, dove

$$A = \begin{pmatrix} B & -I & 0 & \dots & 0 \\ -I & B & -I & \dots & 0 \\ 0 & -I & B & \dots & \dots \\ 0 & \dots & \dots & \dots & -I \\ 0 & 0 & \dots & -I & B \end{pmatrix}$$

in cui

$$B = \begin{pmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{pmatrix}$$

Esercizio in Matlab: matrice di Poisson

Per ulteriori dettagli sulle origini della matrice di Poisson, si considerino ad esempio [1, p. 557], [3, p. 283], [4, p. 334].

Le matrici di Poisson sono

- simmetriche;
- tridiagonali a blocchi;
- diagonalmente dominanti;
- non singolari (deriva dal primo e dal secondo teorema di Gerschgorin [3, p. 76-80], [4, p. 955]);
- definite positive.

Esercizio in Matlab: matrice di Poisson

Per accertarsene, calcoliamo il minimo autovalore della matrice di Poisson con $B \in \mathcal{M}_5$, semplicemente digitando sulla shell di Matlab-Octave

```
>> A=makefish(5);
>> m=min(eig(A))
m =
    0.5359
>>
```

Tale matrice di Poisson non è malcondizionata essendo

```
>> A=makefish(5);
>> cond(A)
ans =
   13.9282
>>
```

Esercizio in Matlab: matrice di Poisson

Poniamo ora

```
b=ones(size(A,1),1);
```

e risolviamo il sistema $Ax = b$ digitando

```
x_sol=A\b;
```

Nota la soluzione esatta confrontiamo i vari metodi risolvendo il sistema lineare con un numero massimo di iterazioni `maxit` e una tolleranza `tol` come segue

```
maxit=200; tol=10^(-8);
```

Esercizio in Matlab: matrice di Poisson

A tal proposito consideriamo l'm-file `demo_algebra_lineare.m`,

```
maxit=200; tol=10^(-8);
siz=5;
A = makefish(siz);       % MATRICE DI POISSON.
b=ones(size(A,1),1);     % TERMINE NOTO.
x_sol=A\b;               % SOLUZIONE ESATTA. METODO LU.
norm_x_sol=norm(x_sol);
if norm(x_sol) == 0
    norm_x_sol=1;
end
x=zeros(size(b));        % VALORE INIZIALE.
```

Esercizio in Matlab: matrice di Poisson

```
% JACOBI.
[x_j,error_j,iter_j,flag_j]=jacobi(A,x,b,maxit,tol);
fprintf('\t \n [JACOBI ][STEP REL., NORMA 2]:%2.2e [REL.ERR.]:%2.2e',error_j,
     norm(x_j-x_sol)/norm_x_sol);
fprintf('\t \n [ITER.]:%3.0f [FLAG]:%1.0f \n',iter_j,flag_j);
% GAUSS-SEIDEL.
w=1;
[x_gs,error_gs,iter_gs,flag_gs]=sor(A,x,b,w,maxit,tol);
fprintf('\t \n [GS ][STEP REL., NORMA 2]:%2.2e [REL.ERR.]:%2.2e',error_gs,norm(
     x_gs-x_sol)/norm_x_sol);
fprintf('\t \n [ITER.]:%3.0f [FLAG]:%1.0f \n',iter_gs,flag_gs);
% SOR.
w_vett=0.8:0.025:2;
for index=1:length(w_vett)
    w=w_vett(index);
   [x_sor,error_sor(index),iter_sor(index),flag_sor(index)] = sor(A,x,b,w,maxit
        ,tol);
   relerr(index)=norm(x_sor-x_sol)/norm_x_sol;
end
```

Esercizio in Matlab: matrice di Poisson

```
[min_iter_sor, min_index]=min(iter_sor);
fprintf('\t \n [SOR OTT.] [STEP REL.,NORMA 2]:%2.2e [REL.ERR.]:%2.2e',error_sor(
      min_index),relerr(min_index));
fprintf('\t \n [ITER.]:%3.0f [FLAG]:%1.0f [w]:%2.3f \n',min_iter_sor,flag_sor(
      min_index),w_vett(min_index));
plot(w_vett,iter_sor,'r-');
% GRADIENTE CONIUGATO.
M=eye(size(A));
[x_gc,error_gc,iter_gc,flag_gc]=cg(A,x,b,M,maxit,tol);
fprintf('\t \n [GC][STEP REL., NORMA 2]:%2.2e [REL.ERR.]:%2.2e',error_gc,norm(
      x_gc-x_sol)/norm_x_sol);
fprintf('\t \n  [ITER.]:%3.0f [FLAG]:%1.0f \n',iter_gc,flag_gc);
```

Esercizio in Matlab: matrice di Poisson

Lanciamo la demo nella shell di Matlab-Octave e otteniamo

```
>> demo_algebra_lineare

 [JACOBI  ] [STEP REL., NORMA 2]: 8.73e-009 [REL.ERR.]: 5.65e-008
            [ITER.]: 116 [FLAG]: 0

 [GAU.SEI.] [STEP REL., NORMA 2]: 9.22e-009 [REL.ERR.]: 2.76e-008
            [ITER.]:  61 [FLAG]: 0

 [SOR OTT.] [STEP REL., NORMA 2]: 2.31e-009 [REL.ERR.]: 1.10e-009
            [ITER.]:  21 [FLAG]: 0 [w]: 1.350

 [GC] [STEP REL., NORMA 2]: 4.41e-017 [REL.ERR.]: 2.21e-016
            [ITER.]:   5 [FLAG]: 0
>>
```

Esercizio in Matlab: matrice di Poisson

Una breve analisi ci dice che

- Come previsto dalla teoria, il metodo di Gauss-Seidel converge in approssimativamente metà iterazioni di Jacobi;
- Il metodo SOR ha quale costante quasi ottimale $w = 1.350$;
- Il metodo del gradiente coniugato converge in meno iterazioni rispetto agli altri metodi (solo 5 iterazioni, ma si osservi il test d'arresto differente). Essendo la matrice di Poisson di ordine 25, in effetti ciò accade in meno di 25 iterazioni come previsto. Vediamo cosa succede dopo 25 iterazioni:

```
>> A=gallery('poisson',5);
>> A=full(A); b=ones(size(A,1),1);
>> maxit=25;tol=0;
>> x=zeros(size(b)); M=eye(size(A));
>> [x_gc,error_gc,iter_gc,flag_gc]=cg(A,x,b,M,maxit,tol);
>> error_gc
error_gc =
   8.3759e-39
```

Il residuo relativo, seppur non nullo è molto piccolo.

Esercizio in Matlab: matrice di Poisson

Un punto delicato riguarda la **scelta del parametro ω ottimale** (cioè minimizzante il raggio spettrale di SOR). Sia questo valore uguale a ω^*. Nel nostro codice abbiamo calcolato per forza bruta ω^+, tra i numeri reali $\omega^+ \leq 2$ del tipo $w_j = 0.8 + j \cdot 0.025$ quello per cui venivano compiute meno iterazioni.

E' possibile calcolare ω^* matematicamente? Nel caso della matrice di Poisson la risposta è affermativa. Da [4, Teor.5.10, p.333]

$$\omega^* = \frac{2}{1 + \sqrt{1 - \rho^2(B_J)}}$$

dove $\rho(B_J)$ è il massimo degli autovalori in modulo della matrice B_J la matrice di iterazione del metodo di Jacobi.

Il raggio spettrale della matrice di iterazione di SOR ott. vale $\omega^* - 1$.

Esercizio in Matlab: matrice di Poisson

Vediamo di calcolare questo valore nel caso della sopracitata matrice di Poisson. Dalla teoria, con ovvie notazioni,

$$B_J = I - D^{-1}A$$

e quindi

```
>> format long;
>> D=diag(diag(A));
>> BJ=eye(size(A))-inv(D)*A;
>> s=eig(BJ);
>> s_abs=abs(s);
>> rho=max(s_abs);
>> w=2/(1+sqrt(1-rho^2))
w =
    1.33333333333333
```

Esercizio in Matlab: matrice di Poisson

```
>> maxit=50; tol=10^(-8);
>> b=ones(size(A,1),1);
>> [x_sor, error_sor, iter_sor, flag_sor] = sor(A, x, b, w, maxit, tol);
>> iter_sor
iter_sor =
    22
>>
```

Esercizio in Matlab: matrice di Poisson

Si rimane un po' sorpresi dal fatto che per $w = 1.350$ il numero di iterazioni fosse inferiore di quello fornito dal valore ottimale teorico $w^* = 1.333\ldots$.

Il fatto è che questo è ottenuto cercando di massimizzare la velocità asintotica di convergenza. Purtroppo questo minimizza una stima del numero di iterazioni k minime da compiere e non quello effettivo.

Abbiamo detto che un punto chiave è la grandezza del raggio spettrale delle matrici di iterazione e che è desiderabile che questo numero oltre ad essere strettamente minore di uno sia il più piccolo possibile. Vediamo i raggi spettrali dei metodi esposti.

Esercizio in Matlab: matrice di Poisson

Salviamo in `raggispettrali.m` il seguente programma principale

```
maxit=50; tol=0;

siz=5;
A = makefish(siz);     % MATRICE DI POISSON.
b=ones(size(A,1),1);   % TERMINE NOTO.

[ M, N ] = split( A , b, 1.0, 1 ); % JACOBI.
P=inv(M)*N;
rho_J=max(abs(eig(P)));
fprintf('\n \t [RAGGIO SPETTRALE][JACOBI]: %2.15f',rho_J);

[ M, N, b ] = split( A, b, 1, 2 ); % GS.
P=inv(M)*N;
rho_gs=max(abs(eig(P)));
fprintf('\n \t [RAGGIO SPETTRALE][GAUSS-SEIDEL]: %2.15f',rho_gs);
```

Esercizio in Matlab: matrice di Poisson

```
D=diag(diag(A));E=-(tril(A)-D); F=-(triu(A)-D);
w=1.350;
M=D/w-E; N=(1/w-1)*D+F; P=inv(M)*N;
rho_sor=max(abs(eig(P)));
fprintf('\n \t [RAGGIO SPETTRALE][SOR BEST]:%2.15f',rho_sor);
w=1.33333333333333;
[ M, N, b ] = split( A, b, w, 2 ); % SOR OPT.
M=D/w-E; N=(1/w-1)*D+F; P=inv(M)*N;
rho_sor_opt=max(abs(eig(P)));
fprintf('\n \t [RAGGIO SPETTRALE][SOR OPT]: %2.15f',rho_sor_opt);
```

Di seguito:

```
>> raggispettrali
   [RAGGIO SPETTRALE][JACOBI]: 0.866025403784438
   [RAGGIO SPETTRALE][GAUSS-SEIDEL]: 0.750000000000000
   [RAGGIO SPETTRALE][SOR BEST]: 0.350000000000001
   [RAGGIO SPETTRALE][SOR OPT]: 0.333333380707781
>>
```

Esercizio in Matlab: matrice di Poisson

Il valore del raggio spettrale della matrice di iterazione del metodo SOR per parametro ottimale, per quanto visto anticipatamente vale $\omega^* - 1$, e l'esperimento numerico lo conferma.
Abbiamo poi osservato che in questo caso la velocità di convergenza del metodo di Gauss-Seidel è il doppio di quella di Jacobi. Poste B_{GS}, B_J le rispettive matrici di iterazione, e detta R la velocità di convergenza, osserviamo che da

$$\begin{aligned} R(B_J) &:= -\ln(\rho(B_J)) && (26) \\ R(B_{GS}) &:= -\ln(\rho(B_{GS})) && (27) \\ R(B_{GS}) &:= 2R(B_J) && (28) \end{aligned}$$

si ha

$$-\ln(\rho(B_{GS})) = R(B_{GS}) = 2R(B_J) = -2\ln(\rho(B_J)) = -\ln(\rho(B_J))^2$$

da cui essendo il logaritmo una funzione invertibile

$$\rho(B_{GS}) = (\rho(B_J))^2.$$

Esercizio in Matlab: matrice di Poisson

Il raggio spettrale della matrice di iterazione di Gauss-Seidel coincide quindi col quadrato di quella di Jacobi ed infatti come è facile verificare

```
>> 0.866025403784438^2
ans =
   0.75000000000000
>>
```

Esercizio in Matlab II

- Si calcoli in Matlab la matrice di Poisson P_{20} di ordine 20 aiutandosi con

```
>> help gallery
```

  In Octave si usi la routine `makefish` scaricabile da internet.
- Sia b il vettore composto di componenti uguali a 1, avente lo stesso numero di righe di P_{20}.
  1. Si risolva con Gauss-Seidel e Jacobi, il problema $P_{20}x = b$, con tolleranza di $10^{(-12)}$, partendo da $x^0 = [0 \ldots 0]$ e $b = [1, \ldots, 1]$. Quante iterazioni servono?
  2. Si risolva col gradiente coniugato il problema $P_{20}x = b$, con tolleranza di $10^{(-12)}$, partendo da $x^0 = [0 \ldots 0]$. Quante iterazioni servono?

Bibliografia

- K. Atkinson, *Introduction to Numerical Analysis*, Wiley, 1989.
- K. Atkinson e W. Han, *Theoretical Numerical Analysis*, Springer, 2001.
- D. Bini, M. Capovani e O. Menchi, *Metodi numerici per l'algebra lineare*, Zanichelli, 1988.
- V. Comincioli, *Analisi Numerica, metodi modelli applicazioni*, Mc Graw-Hill, 1990.
- S.D. Conte e C. de Boor, *Elementary Numerical Analysis, 3rd Edition*, Mc Graw-Hill, 1980.
- L.A. Hageman e D.M. Young *Applied Iterative Methods*, Dover, 2004.
- C.T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, SIAM, 1995.
- The MathWorks Inc., *Numerical Computing with Matlab*, http://www.mathworks.com/moler.
- Netlib, http://www.netlib.org/templates/matlab/.
- A. Quarteroni e F. Saleri, *Introduzione al calcolo scientifico*, Springer Verlag, 2006.
- A. Suli e D. Mayers, *An Introduction to Numerical Analysis*, Cambridge University Press, 2003.
- Wikipedia (Metodo di Gauss-Seidel), http://it.wikipedia.org/wiki/Metodo_di_Gauss-Seidel.
- Wikipedia (Metodo del Gradiente Coniugato), http://it.wikipedia.org/wiki/Metodo_del_gradiente_coniugato.
- Wikipedia (Metodo di Jacobi), http://it.wikipedia.org/wiki/Metodo_di_Jacobi.
- Wikipedia (Successive Over Relaxation), http://it.wikipedia.org/wiki/Successive_Over_Relaxation.