METODI ITERATIVI PER LA SOLUZIONE DI EQUAZIONI LINEARI *

A. SOMMARIVA†

Conoscenze richieste. Spazi vettoriali, operazioni elementari con le matrici, programmazione in Matlab/Octave. Fattorizzazione LU. Norma di matrici.

Conoscenze ottenute. Metodi iterativi stazionari. Metodo di Jacobi. Metodo di Gauss-Seidel. Velocità di convergenza. Raggio spettrale e convergenza di un metodo stazionario. Metodi di rilassamento. Metodo SOR. Velocità di convergenza asintotica. Convergenza dei metodi di Jacobi e Gauss-Seidel per particolari matrici. Metodo del gradiente coniugato.

1. Introduzione.

Sia A una matrice reale avente n righe ed n colonne, b un vettore colonna avente n righe e si supponga di voler risolvere il sistema lineare $Ax = b$. Come noto, se il determinante della matrice è diverso da 0 (cioè la matrice A è non singolare) allora il problema $Ax = b$ ha una ed una sola soluzione.

Ricordiamo che in Matlab/Octave la soluzione può essere calcolata con il metodo LU, utilizzando il comando \. Un esempio:

```
>> A=[1 2 4; 2 4 16; 3 9 81];
>> b=ones(3,1);
>> x=A\b
>> norm(A*x-b)
ans =  9.9301e-16
>> det(A)
ans = -24.000
```

Uno dei principali problemi del metodo LU è legato all'alto costo computazionale. Se A è una generica matrice quadrata di ordine n infatti necessitano circa

$$O\left(\frac{n^3}{3} + \frac{n^2}{2}\right)$$

operazioni moltiplicative, che possono risultare eccessive nel caso di matrici di grandi dimensioni. Per ovviare a questo problema si usano metodi iterativi (stazionari) del tipo

$$x^{(k+1)} = P\,x^{(k)} + c, \;\; k = 0, 1, \ldots$$

con P dipendente da A e c dipendente da A e b (ma non da k). A differenza dei metodi diretti (come ad esempio il metodo LU), in genere un metodo iterativo stazionario convergente calcola usualmente solo un approssimazione della soluzione x (a meno di una tolleranza prefissata). Se m è il numero di iterazioni necessarie, visto che ogni iterazione ha un costo $O(n^2)$ dovuto al prodotto matrice-vettore $P\,x^{(k)}$, ci si augura che il costo computazionale $O(m\,n^2)$ del metodo iterativo sia di gran lunga inferiore a $O(\frac{n^3}{3} + \frac{n^2}{2})$ di un metodo diretto quale LU.

Per una breve storia dell'algebra lineare si consulti [0].

1.1. I metodi di Jacobi, Gauss-Seidel e SOR.

Sia $A = M - N$ con M non singolare, un generico *metodo iterativo stazionario* e' del tipo

$$x^{(k+1)} = M^{-1}Nx^{(k)} + M^{-1}b. \tag{1.1}$$

*Ultima revisione: 17 febbraio 2010

†Dipartimento di Matematica Pura ed Applicata, Universitá degli Studi di Padova, stanza 419, via Trieste 63, 35121 Padova, Italia (alvise@euler.math.unipd.it). Telefono: +39-049-8271350.

La matrice $P = M^{-1}N$ è usualmente chiamata *matrice di iterazione* del metodo iterativo stazionario definito da M, N. Osserviamo che posto $c = M^{-1}b$, il metodo sopracitato è ovviamente tystazionario essendo

$$x^{(k+1)} = Px^{(k)} + c \tag{1.2}$$

con P e c indipendenti da k.

Questa definizione dei metodi stazionari, forse un po' astratta, ha il vantaggio di offrire una rappresentazione compatta degli stessi ed è comunemente utilizzata in letteratura.
Sia ora $A = D - E - F$ con D matrice diagonale, E, F rispettivamente triangolare inferiore e superiore con elementi diagonali nulli.

1.2. Il metodo di Jacobi. Il **metodo di Jacobi** fu scoperto nel 1845, nell'ambito di alcune ricerche su problemi di piccole oscillazioni che comportavano alla risoluzione di sistemi lineari con matrici diagonalmente dominanti [0, p.313].

Nel caso del metodo di Jacobi [0] si ha

$$M = D, \ N = E + F \tag{1.3}$$

e quindi

$$P = M^{-1}N = D^{-1}(E+F) = D^{-1}(D-D+E+F) = D^{-1}(D-A) = I - D^{-1}A \tag{1.4}$$

Si osservi che se D è non singolare allora il metodo di Jacobi, almeno in questa versione di base, non può essere utilizzato visto che in (7) 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

$$x_i^{(k+1)} = (b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)})/a_{ii}, \ i = 1, \ldots, n. \tag{1.5}$$

1.3. Il metodo di Gauss-Seidel. Il **metodo di Gauss-Seidel** fu scoperto nel 1874, da studi preliminari di Gauss (1823) completati dal suo allievo Seidel per lo studio di problemi ai minimi quadrati del tipo $Sx = f$ con S non quadrata, che venivano risolti quali soluzione del sistema di equazioni normali $S^T SxS^T f$. Mentre Gauss oltre a problemi di Astronomia era interessato a problemi di Geodesia (triangolazione di Hannover usando una *catena* di 26 triangoli), Seidel si interessava alla risoluzione di un sistema di equazioni con 72 incognite per uno studio di luminosità stellare.

Il metodo di Gauss-Seidel [0] è definito quale metodo stazionario in cui

$$M = D - E, N = F \tag{1.6}$$

e quindi

$$P = M^{-1}N = (D - E)^{-1}F \tag{1.7}$$

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

$$x_i^{(k+1)} = \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)} \right) / a_{ii}. \tag{1.8}$$

Da (8) si capisce perchè tale metodo è noto anche come *metodo delle sostituzioni successive*.

1.4. Generalizzazioni del metodo di Jacobi e Gauss-Seidel. Quali generalizzazioni del metodo di Jacobi e Gauss-Seidel si introducono, per un opportuno parametro ω, la versione **rilassata del metodo di Jacobi**

$$x^{(k+1)} = (I - \omega D^{-1} A) x^{(k)} + \omega D^{-1} b \tag{1.9}$$

la versione **rilassata del metodo di Gauss-Seidel**

$$x^{(k+1)} = \left(\frac{D}{\omega} - E \right)^{-1} \left(\left(\frac{1}{\omega} - 1 \right) D + F \right) x^{(k)} + \left(\frac{D}{\omega} - E \right)^{-1} b. \tag{1.10}$$

L'idea di fondo di questi metodi rilassati è la seguente [0, p. 261], [0]. Ogni metodo precedentemente esposto può essere scritto come

$$x^{(k+1)} = x^{(k)} + r^{(k)}$$

ove $r^{(k)}$ è la correzione da apportare per passare da $x^{(k)}$ a $x^{(k+1)}$. Nei metodi rilassati, se $r^{(k)}$ è la correzione di Jacobi o Gauss-Seidel, si considera quale correzione $w \cdot r^{(k)}$ e quindi

$$x^{(k+1)} = x^{(k)} + w \cdot r^{(k)}.$$

Si osservi che i metodi di Jacobi e Gauss-Seidel si ottengono rispettivamente da (9) e (10) per la scelta $\omega = 1$.

2. Convergenza dei metodi iterativi.

2.1. Norma di matrici. Sia $\rho(P)$ il massimo degli autovalori in modulo della matrice di iterazione $P = M^{-1}N$ (il cosidetto *raggio spettrale*).

Sia $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}_+$ 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.

Vediamo alcuni esempi. 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. [0, 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. [0, p.26])

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

- Si definisce $\|x\|_2 := \left(\sum_{k=1}^{n} |x_k|^2 \right)^2$ e si dimostra che la norma naturale corrispondente è (cf. [0, 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

>>
```

Si dimostra che (cf. [0, p.28])

TEOREMA 2.1. *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. [0, p.29], [0, p.232])

TEOREMA 2.2. *Fissata una norma naturale $\|\cdot\|$, i seguenti asserti sono equivalenti*

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

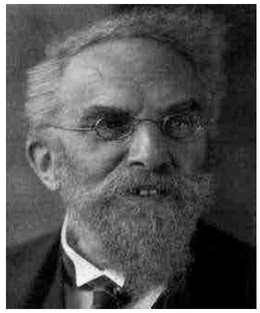

FIGURA 2.1. *Kurt Wilhelm Sebastian Hensel (1861-1941).*

Nota.

1. Ricordiamo che il raggio spettrale 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.
2. 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.

2.2. Il teorema di Hensel e la convergenza di un metodo iterativo stazionario. Consideriamo un metodo iterativo stazionario $x^{(k+1)} = Px^{(k)} + c$ in cui scelto $x^{(0)}$ si abbia

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

dove $\{u_k\}_k$ è una base di autovettori di P avente autovalori $\{\lambda_k\}_k$. Questo accade se e solo se A è diagonalizzabile, cioè simile a una matrice diagonale (cf. [0, p.57]). Se il metodo è *consistente*, cioè $x^* = Px^* + c$ abbiamo $x^{(k)} - x^* = P(x^{(k-1)} - x^*) = P^k(x^0 - x^*) = \sum_{s=1}^{n} c_s P^k u_s = \sum_{s=1}^{n} c_s \lambda_s^k u_s$ e quindi se $|\lambda^k| < 1$ per ogni k abbiamo

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

mentre se 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, abbiamo che la componente $c_l \lambda_s^l$ relativa all'autovettore u_s non tende a 0 e quindi $x^{(k)} - x^*$ non tende a 0. Di conseguenza non è vero che il metodo è convergente per qualsiasi scelta del vettore $x^{(0)}$.
Di conseguenza

TEOREMA 2.3. *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.*

Dimostriamo ora una sua generalizzazione, scoperta da Hensel nel 1926 [0, p.313].

TEOREMA 2.4. *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. (cf. [0, p.236])

- Se $\rho(P) < 1$, allora il problema $x = Px + c$ ha una e una sola soluzione 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^*.
Sia $e(k) = x^{(k)} - x^*$. Come stabilito dal Teorema , sia inoltre una norma naturale $\| \cdot \|$ tale che
$$\rho(A) \leq \|A\| = \rho(A) + (1 - \rho(A))/2 < 1.$$
Essendo $x^{(k+1)} = Px^{(k)} + c$ e $x = Px + c$, sottraendo membro a membro le equazioni si ottiene
$$e^{(k+1)} = Pe^{(k+1)} = P^k e^{(0)}$$
da cui
$$\|e^{(k+1)}\| = \|Pe^{(k)}\| = \|P^k e^{(0)}\| \leq \|P^k\| \|e^{(0)}\|.$$
Poichè il raggio spettrale è minore di 1 allora $\|P^k\| \to 0$ da cui $\|e^{(k+1)}\| \to 0$ e quindi per le proprietà delle norme $e^{(k+1)} \to 0$ cioè $x^{(k)} \to 0$.
Si noti che questa direzione della dimostrazione poteva essere vista come applicazione del teorema di punto fisso di Banach che stabilisce che se K è un insieme non vuoto e chiuso di uno spazio di Banach V e $T : K \to K$ è una mappa L contrattiva, cioè $\|T(x) - T(y)\| < L\|x - y\|$ con $0 \leq L < 1$, allora esiste ed è unico $x^* \in K$ tale che $x^* = T(x^*)$ e inoltre per ogni $x^{(0)} \in K$ la sequenza $\{x^{(k)}\}_k \subseteq K$ definita da $x^{(k+1)} = T(x^{(k)})$, $k = 0, 1, \ldots$ converge ad x^*. Per una dimostrazione si veda ad esempio [0, p.133], [0, p.133]. Il problema che stiamo analizzando corrisponde a porre $K = V = \mathbb{R}^n$ dotati di una norma $\| \cdot \|$ tale che
$$\rho(A) \leq \|A\| = (1 + \rho(A))/2 < 1,$$
e $T(x) = Px + c$. Certamente T è contrattiva in quanto
$$\|T(x) - T(y)\| = \|Px + c - Py - c\| \leq \|P(x - y)\| \leq \|P\| \|x - y\| = \frac{1 + \rho(A)}{2} \|x - y\|.$$
Di conseguenza per ogni $x^{(0)} \in \mathbb{R}^n$ la sequenza $x^{(k+1)} = Px^{(k)} + c$ converge a x^* soluzione di $x = Px + c$.
- 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 $e^{(0)} = x^{(0)} - x^*$ un suo autovettore. 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 $\| \cdot \|$,
$$\|e^{(k+1)}\| = |\lambda_{\max}^k|_{\mathbb{C}} \|e^{(0)}\| \geq \|e^{(0)}\|$$
il che comporta che la successione non è convergente. $\square$

2.3. Sulla velocità di convergenza. Abbiamo visto che

$$\|e^{(k)}\| \leq \|P^k\|\|e^{(0)}\|, \;\; e^{(k)} = x^{(k)} - x^* \tag{2.1}$$

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. [0, p.239]).
Si dimostra che

TEOREMA 2.5. *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

$$\|P^k\| \approx \rho^k(P).$$

Sotto queste ipotesi, se

$$\|e^{(k+m)}\| \approx \|P^m\|\|e^{(k)}\| \tag{2.2}$$

abbiamo

$$\|e^{(k+m)}\| \approx \|P^m\|\|e^{(k)}\| \approx \rho^{\frac{1}{m}}(P)\|e^{(k)}\| \tag{2.3}$$

per cui affinchè

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

applicando il logaritmo $\ln$ ad ambo i membri, si vede serve sia,

$$m \ln(\rho(P)) \approx \ln \epsilon \Rightarrow m \approx \frac{\ln \epsilon}{\ln(\rho(P))}$$

Se

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

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 è circa

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

Conseguentemente 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.

3. I metodi di Richardson. Fissato α, la versione di base del metodo di Richardson consiste in un metodo iterativo del tipo

$$x^{(k+1)} - x^{(k)} = \alpha r^{(k)}. \tag{3.1}$$

D'altra parte come visto precedentemente i metodi di Jacobi e di Gauss-Seidel e le loro versioni *rilassate* sono metodi iterativi del tipo

$$Mx^{(k+1)} = Nx^{(k)} + b, \tag{3.2}$$

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

$$A = M - N. \tag{3.3}$$

Se

$$r^{(k)} = b - Ax^{(k)} \tag{3.4}$$

è il *residuo* alla k-sima iterazione allora da (15) e (16)

$$M(x^{(k+1)} - x^{(k)}) = Nx^{(k)} + b - Mx^{(k)} = b - Ax^{(k)} = r^{(k)} \tag{3.5}$$

Ne consegue che i metodi di Jacobi e di Gauss-Seidel e le loro versioni *rilassate* sono generalizzazioni di un metodo di Richardson del tipo

$$M(x^{(k+1)} - x^{(k)}) = \alpha r^{(k)} \tag{3.6}$$

in cui la matrice invertibile M è detta di *precondizionamento*.

3.1. Il metodo di Richardson precondizionato con parametro fisso α ottimale. Per un opportuno parametro di accelerazione $\alpha > 0$ (da non confondersi con quello di SOR), si può fornire un'ovvia generalizzazione del metodo (18)

$$M(x^{(k+1)} - x^{(k)}) = \alpha\, r^{(k)},\ k \geq 0. \tag{3.7}$$

Evidentemente (18) corrisponde alla scelta $\alpha = 1$.

Il parametro $\alpha > 0$ viene scelto cosí da minimizzare il raggio spettrale della matrice di iterazione. In questo caso si vede che da

$$M(x^{(k+1)} - x^{(k)}) = \alpha\,(b - Ax^{(k)}) \tag{3.8}$$

necessariamente

$$Mx^{(k+1)} = Mx^{(k)} + \alpha\,(b - Ax^{(k)}) = (M - \alpha A)x^{(k)} + \alpha\, b, \tag{3.9}$$

e quindi con le precedenti notazioni

$$M_\alpha = \frac{M}{\alpha},\ N_\alpha = \frac{M - \alpha\, A}{\alpha} \tag{3.10}$$

per cui la matrice di iterazione $R_\alpha = M_\alpha^{-1} N_\alpha$ diventa

$$C = M^{-1}(M - \alpha\, A) = I - \alpha\, M^{-1} A. \tag{3.11}$$

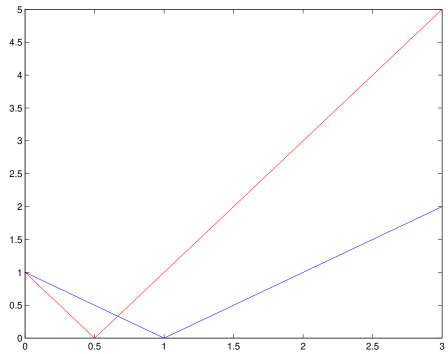

FIGURA 3.1. *Grafici di* $|1 - \alpha\lambda_{max}|$ *e* $|1 - \alpha\lambda_{min}|$ *(rispettivamente in rosso e in blu).*

Se $M^{-1}A$ è definita positiva e λ_{min} e λ_{max} sono rispettivamente il minimo e massimo autovalore di $M^{-1}A$, allora il valore ottimale del parametro α, cioè quello per cui è minimo il raggio spettrale della matrice d'iterazione $M - \alpha\, A$ è

$$\alpha_{\text{ott}} = \frac{2}{\lambda_{\text{min}} + \lambda_{\text{max}}} \tag{3.12}$$

ed in corrispondenza si ha che la matrice di iterazione $R_{\alpha_{\text{ott}}}$ ha raggio spettrale

$$\alpha_{\text{ott}} = \frac{\lambda_{\text{max}} - \lambda_{\text{min}}}{\lambda_{\text{min}} + \lambda_{\text{max}}} \tag{3.13}$$

Per capirlo si dimostra dapprima che qualsiasi sia $\lambda \in [\lambda_{\text{min}}, \lambda_{\text{max}}]$ si ha

$$|1 - \alpha\lambda| \leq \max(|1 - \alpha\lambda_{\text{min}}|, |1 - \alpha\lambda_{\text{max}}|)$$

e che

$$\min_{\alpha \in \mathbb{R}} \max(|1 - \alpha\lambda_{\text{min}}|, |1 - \alpha\lambda_{\text{max}}|)$$

lo si ottiene quando la retta $y = \alpha\lambda_{\text{max}} - 1$ interseca la retta $y = 1 - \alpha\lambda_{\text{min}}$, che è proprio per $\alpha = \alpha_{\text{ott}}$.

Si osservi che la scelta di α non dipende dall'iterazione; di conseguenza (20) definisce il cosidetto *metodo di Richardson stazionario precondizionato*, per distinguerlo dal *metodo di Richardson non stazionario precondizionato*

$$M(x^{(k+1)} - x^{(k)}) = \alpha_k\,(b - Ax^{(k)}). \tag{3.14}$$

con α_k che non è necessariamente costante.

3.2. Il metodo del gradiente coniugato. Un classico metodo di Richardson non stazionario è quello del *gradiente* (detto anche di *discesa più ripida*). 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

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

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.

Si dimostra [0, p.341] che il parametro α_k ottimale cosicchè $\phi(x^{(k+1)})$ sia minimo una volta scelta $p^{(k)}$ è

$$\alpha_k = \frac{(r^{(k)})^T p^{(k)}}{(p^{(k)})^T A p^{(k)}}$$

Nel metodo del gradiente si sceglie quale direzione $p^{(k)} = \text{grad}(\phi(x))|_{x=x^{(k)}}$. Ma se $r^{(k)} = b - Ax^{(k)}$, allora

$$\begin{aligned} \text{grad}(\phi(x))|_{x=x^{(k)}} &= \frac{1}{2}\text{grad}(x^T Ax)|_{x=x^{(k)}} - \text{grad}(b^T x)|_{x=x^{(k)}} \\ &= Ax^{(k)} - b = -r^{(k)} \end{aligned} \tag{3.15}$$

e quindi $p^{(k)} = r^{(k)}$ (è essenziale la direzione ma non il segno e per convincersene si calcoli la successione anche con segno opposto $p^{(k)} = -r^{(k)}$ per parametro α_k ottimale).

Di conseguenza il metodo del gradiente è definito dalla successione tipica dei metodi di Richardson non stazionari

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

dove

$$\alpha_k = \frac{(r^{(k)})^T p^{(k)}}{(p^{(k)})^T A p^{(k)}} = \frac{\|r^{(k)}\|_2^2}{(r^{(k)})^T A r^{(k)}}.$$

3.3. Una stima dell'errore per alcuni metodi di Richardson. Per quanto riguarda una stima d'errore, citiamo il seguente teorema (cf. [0, p.148]).

TEOREMA 3.1. *Siano A e M due matrice simmetriche e definite positive e si consideri un metodo di Richardson $M(x^{(k+1)} - x^{(k)}) = \alpha_k r^{(k)}$, dove*

1. *$M = I$ e*

$$\alpha_k = \frac{(z^{(k)})^T r^{(k)}}{(z^{(k)})^T A z^{(k)}}, \;\; z^{(k)} = M^{-1} r^{(k)}$$

2. *oppure M invertibile con la scelta (non dinamica) del parametro α_k*

$$\alpha_k = \alpha_{opt} = \frac{2}{\lambda_{min} + \lambda_{max}}.$$

Posto $\|v\|_A := \sqrt{v^T A v}$, $e^{(k)} = x^{(k)} - x$, si ha

$$\|e^{(k)}\|_A \leq \left(\frac{\kappa(M^{-1}A) - 1}{\kappa(M^{-1}A) + 1} \right)^k \|e^{(0)}\|_A$$

dove $\kappa(M^{-1}A)$ è il numero di condizionamento della matrice $M^{-1}A$.

Discutiamo l'asserto.

1. Nel caso del metodo del gradiente, che corrisponde alla scelta $M = I$ e $z^{(k)} = r^{(k)}$, vale quindi la stima

$$\|e^{(k)}\|_A \leq \left(\frac{\kappa(A) - 1}{\kappa(A) + 1}\right)^k \|e^{(0)}\|_A$$

che mostra che più grande è il numero di condizionamento $\kappa(A)$ più è vicino a 1 la quantità $\frac{\kappa(A)-1}{\kappa(A)+1}$ il che giustifica una *possibile* convergenza lenta del metodo.
2. Nel caso del metodo precondizionato, si vede che una scelta quasi ottimale è quella per cui $\kappa(M^{-1}A)$ è vicino a 1. Osserviamo d'altra parte che non si può scegliere $M = A$ in quanto con facili calcoli ci si accorge che non si potrebbe calcolare il valore $x^{(k+1)}$ a partire da $x^{(k)}$.

4. Matrici simmetriche definite positive: il metodo del gradiente coniugato. Il metodo del gradiente coniugato (di cui forniremo solo il codice e alcune brevi indicazioni) fu descritto nel 1952 da Hestenes e Stiefel ma per quanto destasse subito l'interesse dell'ambiente matematico non venne molto utilizzato fino al 1971, quando Reid suggerì il suo utilizzo per la risoluzione di sistemi sparsi (cioè con molte componenti nulle) di grandi dimensioni [0], [0].

La successione delle iterazioni del gradiente coniugato è quella propria dei metodi di discesa,

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

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

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

Con questa scelta si prova che

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

cioè i vettori $p^{(k)}$ e $p^{(k-1)}$ sono A-*coniugati*.

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

- Sia

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

per $k \geq 1$. Allora la k-sima iterata dal metodo del gradiente coniugato minimizza il funzionale ϕ nell'insieme $x^{(0)} + \mathcal{K}_k$ [0, p.12].
- Se A è una matrice simmetrica e definita positiva di ordine n, si può dimostrare che il metodo 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.
- Si può dimostrare [0, p. 279] 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). Esistono varie versioni di questa disuguaglianza. Ad esempio in [0, p. 151]:

$$\|e_k\|_A \leq \left(\frac{2c^k}{1 + 2c^k} \right) \|e_0\|_A$$

dove

$$c := \frac{\sqrt{K_2(A)} - 1}{\sqrt{K_2(A)} + 1}.$$

- Sia A simmetrica e definita positiva. Si supponga che ci siano esattamente $k \leq n$ autovalori distinti di A. Allora il metodo del gradiente coniugato converge in al più k iterazioni.
- Sia A simmetrica e definita positiva. Si supponga b sia combinazione lineare di $k \leq n$ autovettori distinti di A. Allora il metodo del gradiente coniugato con la scelta $x^{(0)} = 0$ converge in al più k iterazioni.

L'analisi del metodo è piuttosto complessa. Qualora interessati si confronti con [0, p. 562-569], [0, p. 272-283], [0, p. 340-356], [0, p. 11-29], [0, p. 145-153].

5. Convergenza dei Jacobi, Gauss-Seidel ed SOR. Lo studio della convergenza dei metodi di Jacobi, Gauss-Seidel ed SOR [0] è un proposito complicato e ci limiteremo a citare, senza dimostrazione, alcuni classici risultati [0, p. 231-315].

Ricordiamo che

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

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

e per almeno un indice s si abbia

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

Ad esempio la matrice

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

è a predominanza diagonale (per righe).

2. A è a predominanza diagonale in senso stretto (per righe) se per ogni $i = 1, \ldots, n$ risulta

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

Ad esempio la matrice

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

è a predominanza diagonale in senso stretto (per righe).
3. A è a predominanza diagonale per colonne (in senso stretto) se A^T è a predominanza diagonale per righe (in senso stretto).
4. A è tridiagonale se $a_{i,j} = 0$ per $|i-j| > 1$. Ad esempio la matrice

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

è tridiagonale.
5. A è definita positiva se e solo se i suoi autovalori sono positivi.
La matrice

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

è definita positiva come si può vedere usando i seguenti comandi Matlab/Octave

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

6. A di ordine $n \geq 2$ è riducibile se esiste una matrice di permutazione Π e un intero k con $0 < k < n$, tale che

$$B = \Pi A \Pi^T = \begin{pmatrix} A_{1,1} & A_{1,2} \\ 0 & A_{2,2} \end{pmatrix}$$

in cui $A_{1,1} \in C^{k\times k}$, $A_{2,2} \in C^{(n-k)\times(n-k)}$. Se A non è riducibile si dice che A è irriducibile.

Il metodo di Jacobi risulta convergente in uno dei seguenti casi [0, p. 247]:

1. A è a predominanza diagonale in senso stretto;
2. A è a predominanza diagonale ed è irriducibile;
3. A è a predominanza diagonale in senso stretto per colonne;
4. A è a predominanza diagonale per colonne ed è irriducibile.

TEOREMA 5.1. *Sia A una matrice quadrata a predominanza diagonale. Allora il metodo di Jacobi converge alla soluzione di $Ax = b$, qualsiasi sia il punto $x^{(0)}$ iniziale.* *Dimostrazione.* Supponiamo che A sia a predominanza diagonale in senso stretto per righe. Allora per ogni $i = 1, \ldots, n$ risulta

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

Nel caso del metodo di Jacobi

$$M = D,\ N = E + F,\ P = M^{-1}N = D^{-1}(E + F), \tag{5.1}$$

da cui

$$P_{i,j} = \begin{cases} \frac{a_{i,j}}{a_{i,i}} & \text{se } i \neq j \\ 0 & \text{se } i = j \end{cases}$$

Di conseguenza

$$\|P\|_\infty = \max_i \sum_{j=1}^{n} |P_{i,j}| = \max_i \sum_{j=1}^{n} \frac{|a_{i,j}|}{|a_{i,i}|} < 1$$

ed essendo $\rho(P) \leq \|P\|_1 < 1$ abbiamo che il metodo di Jacobi è convergente. □

TEOREMA 5.2. *Il metodo di Gauss-Seidel risulta convergente in uno dei seguenti casi [0, p. 249]:*

1. *A è a predominanza diagonale in senso stretto.*
2. *Sia A una matrice simmetrica definita positiva, non singolare con elementi principali $a_{i,i} \neq 0$. Allora Gauss-Seidel è convergente se e solo se A è definita positiva.*

TEOREMA 5.3. *Per matrici tridiagonali (a blocchi) $A = (a_{i,j})$ 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).*

TEOREMA 5.4. *Sia A simmetrica con elementi diagonali positivi. Allora il metodo SOR converge se e solo se $0 < w < 2$ e A è definita positiva [0, p.215].*

6. Test d'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

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

6.1. Sul criterio dello step. Posto $\Delta^{(k)} := x^{(k+1)} - x^{(k)}$ e $e^{(k)} = x^* - x^{(k)}$, essendo

$$\begin{aligned} e^{(k)} &= x^* - x^{(k)} = (Px^* + c) - (Px^{(k)} + c) \\ &= P(x^* - x^{(k)}) = Pe^{(k-1)} \end{aligned} \tag{6.1}$$

abbiamo

$$\begin{aligned} \|e^{(k)}\|_2 &= \|x^* - x^{(k)}\|_2 = \|(x^* - x^{(k+1)}) + (x^{(k+1)} - x^{(k)})\|_2 \\ &= \|e^{(k+1)} + \Delta^{(k)}\|_2 = \|Pe^{(k)} + \Delta^{(k)}\|_2 \leq \|P\|_2 \cdot \|e^{(k)}\|_2 + \|\Delta^{(k)}\|_2 \end{aligned} \tag{6.2}$$

Fissata dall'utente una tolleranza *tol*, si desidera interrompere il processo iterativo quando $|x^* - x^{(k)}| \leq \text{tol}$. Non disponendo di x^*, il test *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 $\{\lambda_k\}_k$ Di conseguenza, se P è simmetrica

$$\begin{aligned} \|P\|_2 &= \sqrt{\rho(PP^T)} = \sqrt{\rho(U\Lambda U^T (U\Lambda U^T)^T)} \\ &= \sqrt{\rho(U\Lambda^2 U^T)} \end{aligned} \tag{6.3}$$

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

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

e quindi ricaviamo

$$\begin{aligned} \|P\|_2 &= \sqrt{\rho(\Lambda^2)} = \sqrt{\max_k |\lambda_k^2|} \\ &= \sqrt{(\max_k |\lambda_k|^2)} = \sqrt{(\max_k |\lambda_k|)^2} \\ &= \max_k |\lambda_k| = \rho(P) \end{aligned} \tag{6.4}$$

Di conseguenza da (31)

$$\begin{aligned} \|e^{(k)}\|_2 &\leq \|P\|_2 \cdot \|e^{(k)}\|_2 + \|\Delta^{(k)}\|_2 \\ &= \rho(P) \cdot \|e^{(k)}\|_2 + \|\Delta^{(k)}\|_2 \end{aligned} \tag{6.5}$$

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

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

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

6.2. Sul criterio del residuo. Si definisce *residuo* alla k-sima iterazione la quantità

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

ed essendo $b = Ax^*$ abbiamo

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

da cui

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

Interromperemo il processo iterativo quando $r^{(k)} \leq \text{tol}$, desiderando sia pure

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

Notiamo che

1. essendo A invertibile e $r^{(k)} = Ae^{(k)}$ ricaviamo $e^{(k)} = A^{-1}r^{(k)}$ da cui

$$\|e^{(k)}\| = \|A^{-1}r^{(k)}\| \leq \|A^{-1}\|\|r^{(k)}\|;$$

2. poichè $b = Ax^*$ abbiamo $\|b\| \leq \|A\|\|x^*\|$ e quindi

$$\frac{1}{\|x^*\|} \leq \frac{\|A\|}{\|b\|}.$$

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

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

Il criterio d'arresto $\frac{\|r^{(k)}\|}{\|b\|} \leq \text{tol}$ è quindi molto conservativo quando $\kappa(A) \gg 1$.

7. Metodi iterativi in Matlab.

7.1. Metodo di Jacobi in Matlab. Un codice Matlab/Octave del metodo di Jacobi,fornito in internet presso il sito di Netlib

http://www.netlib.org/templates/matlab/

è il seguente

```
function [x, error, iter, flag]  = jacobi(A, x, b, max_it, tol)

%  -- Iterative template routine --
%     Univ. of Tennessee and Oak Ridge National Laboratory
%     October 1, 1993
%     Details of this algorithm are described in "Templates for the
```

```
%     Solution of Linear Systems: Building Blocks for Iterative
%     Methods", Barrett, Berry, Chan, Demmel, Donato, Dongarra,
%     Eijkhout, Pozo, Romine, and van der Vorst, SIAM Publications,
%     1993. (ftp netlib2.cs.utk.edu; cd linalg; get templates.ps).
%
% [x, error, iter, flag]  = jacobi(A, x, b, max_it, tol)
%
% jacobi.m solves the linear system Ax=b using the Jacobi Method.
%
% 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

  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
```

Il codice di `jacobi` utilizza una funzione `split` che serve per calcolare le matrici M, N che definiscono l'iterazione del metodo di Jacobi:

```
function [ M, N, b ] = split( A, b, w, flag )
%
% function [ M, N, b ] = split_matrix( A, b, w, flag )
%
% split.m sets up the matrix splitting for the stationary
```

```
% iterative methods: jacobi and sor (gauss-seidel when w = 1.0 )
%
% 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;

% END split.m
```

Ricordiamo che la funzione `split` non coincide con quella predefinita nelle ultime releases di Matlab/Octave. Qualora la funzione `split` che vogliamo utilizzare sia salvata della directory corrente, una volta richiamata, i workspace di Matlab/Octave utilizzano proprio questa e non quella descritta per altri usi in Matlab/Octave. Inoltre per quanto riguarda `tril` e `triu` in `split` dall'help di Matlab si capisce che estraggono rispettivamente la parte triangolare inferiore e superiore di una matrice:

```
>> help tril

 TRIL Extract lower triangular part.
    TRIL(X) is the lower triangular part of X.
    TRIL(X,K) is the elements on and below the K-th diagonal
    of X .  K = 0 is the main diagonal, K > 0 is above the
    main diagonal and K < 0 is below the main diagonal.

    See also TRIU, DIAG.

>> help triu

 TRIU Extract upper triangular part.
    TRIU(X) is the upper triangular part of X.
    TRIU(X,K) is the elements on and above the K-th diagonal of
    X.  K = 0 is the main diagonal, K > 0 is above the main
    diagonal and K < 0 is below the main diagonal.
```

```
    See also TRIL, DIAG.

>> A=[1 2 3; 4 5 6; 7 8 9]
A =
     1     2     3
     4     5     6
     7     8     9
>> tril(A)
ans =
     1     0     0
     4     5     0
     7     8     9
>> triu(A)
ans =
     1     2     3
     0     5     6
     0     0     9
>> tril(A,-1)
ans =
     0     0     0
     4     0     0
     7     8     0
>> triu(A,1)
ans =
     0     2     3
     0     0     6
     0     0     0
>> triu(A,-1)
ans =
     1     2     3
     4     5     6
     0     8     9
>>
```

La routine `jacobi` è scritta da esperti di algebra lineare e si interrompe quando la norma 2 dello step relativo

$$\frac{\|x^{(k+1)} - x^{(k)}\|_2}{\|x^{(k+1)}\|_2}$$

è inferiore ad una tolleranza `tol` prefissata oppure un numero massimo di iterazioni `max_it` è raggiunto. Ricordiamo che se $v = (v_i)_{i=1,\ldots,n}$ è un elemento di $\mathbb{R}^n$ allora

$$\|v\|_2 = \sqrt{\sum_{i=1}^{n} v_i^2}.$$

Problema: cosa succede quando la matrice diagonale estratta da A è singolare? cosa succede quando partendo da $x_0 \neq 0$, si ha per qualche indice $k > 0$ che $x_k = 0$?

7.2. Metodo di Gauss-Seidel in Matlab. La versione di Gauss-Seidel con la scelta del parametro ω è nota in letteratura come **SOR**, acronimo di *successive over relaxation*. Una versione di SOR scaricabile presso il sito di Netlib [0] è la seguente

```
function [x, error, iter, flag]  = sor(A, x, b, w, max_it, tol)

%  -- Iterative template routine --
%     Univ. of Tennessee and Oak Ridge National Laboratory
%     October 1, 1993
%     Details of this algorithm are described in "Templates for the
%     Solution of Linear Systems: Building Blocks for Iterative
%     Methods", Barrett, Berry, Chan, Demmel, Donato, Dongarra,
%     Eijkhout, Pozo, Romine, and van der Vorst, SIAM Publications,
%     1993. (ftp netlib2.cs.utk.edu; cd linalg; get templates.ps).
%
% [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

  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
```

Come per il metodo di Jacobi, il processo si interrompe quando la norma 2 dello step relativo

$$\frac{\|x^{(k+1)} - x^{(k)}\|_2}{\|x^{(k+1)}\|_2}$$

è inferiore ad una tolleranza `tol` prefissata oppure un numero massimo di iterazioni `max_it` è raggiunto.

Per ulteriori dettagli si consulti ad esempio [0, p. 313-339].

7.3. Metodo del gradiente coniugato in Matlab. Per quanto riguarda il codice del Gradiente Coniugato, un esempio è il file `cg.m` tratto da Netlib [0]:

```
function [x, error, iter, flag] = cg(A, x, b, M, max_it, tol)

%  -- Iterative template routine --
%     Univ. of Tennessee and Oak Ridge National Laboratory
%     October 1, 1993
%     Details of this algorithm are described in "Templates for the
%     Solution of Linear Systems: Building Blocks for Iterative
%     Methods", Barrett, Berry, Chan, Demmel, Donato, Dongarra,
%     Eijkhout, Pozo, Romine, and van der Vorst, SIAM Publications,
%     1993. (ftp netlib2.cs.utk.edu; cd linalg; get templates.ps).
%
%  [x, error, iter, flag] = cg(A, x, b, M, max_it, tol)
%
% cg.m solves the symmetric positive definite linear system Ax=b
% using the Conjugate Gradient method with preconditioning.
%
% input   A        REAL symmetric positive definite matrix
%         x        REAL initial guess vector
%         b        REAL right hand side vector
%         M        REAL preconditioner matrix
%         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

  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

  for iter = 1:max_it                       % begin iteration

     z  = M \ r;
```

```
    rho = (r'*z);

    if ( iter > 1 ),                       % direction vector
       beta = rho / rho_1;
       p = z + beta*p;
    else
       p = z;
    end

    q = A*p;
    alpha = rho / (p'*q );
    x = x + alpha * p;                    % update approximation vector

    r = r - alpha*q;                      % compute residual
    error = norm( r ) / bnrm2;            % check convergence
    if ( error <= tol ), break, end

    rho_1 = rho;

  end

  if ( error > tol ) flag = 1; end        % no convergence

% END cg.m
```

Osserviamo che il procedimento itera finchè un numero massimo di iterazioni è raggiunto oppure la norma 2 del residuo (relativo)

$$\frac{\|b - Ax^{(k)}\|_2}{\|b\|_2}$$

immagazzinata nella variabile `error` risulta inferiore ad una tolleranza prefissata `tol`. In questo caso il criterio d'arresto del metodo del gradiente coniugato è diverso da quello dello step relativo utilizzato nelle precedenti versioni di `Jacobi` ed `SOR`.

8. Un esperimento numerico. 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

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

La matrice A è facilmente esprimibile utilizzando la funzione `makefish` scaricabile in [0]

```
function mat = makefish(siz);
% make a Poisson matrix

leng = siz*siz;
dia = zeros(siz,siz);
off = -eye(siz,siz);
 for i=1:siz, dia(i,i)=4; end;
 for i=1:siz-1, dia(i,i+1)=-1; dia(i+1,i)=-1; end;
mat = zeros(leng,leng);
 for ib=1:siz,
  mat(1+(ib-1)*siz:ib*siz,1+(ib-1)*siz:ib*siz) = dia; end;
 for ib=1:siz-1,
  mat(1+(ib-1)*siz:ib*siz,1+ib*siz:(ib+1)*siz) = off;
  mat(1+ib*siz:(ib+1)*siz,1+(ib-1)*siz:ib*siz) = off; end;
return;
```

Vediamo un esempio:

```
>> makefish(3)

ans =

     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

>>
```

che evidentemente è una matrice di Poisson con B matrice quadrata di ordine 3

```
B =   4    -1     0
     -1     4    -1
      0    -1     4
```

Per ulteriori dettagli sulle origini della matrice di Poisson, si considerino ad esempio [0, p. 557], [0, p. 283], [0, p. 334]. Le matrici di Poisson sono evidentemente simmetriche, tridiagonali a blocchi, diagonalmente dominanti e dal primo e dal secondo teorema di Gerschgorin [0, p. 76-80], [0, p. 955] si può provare che sono non singolari. In particolare si può mostrare che A è definita positiva. 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
>>
```

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);
```

A tal proposito consideriamo l'm-file

```
demo_algebra_lineare.m
```

contenente il codice

```
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.

% 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 [GAU.SEI.] [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

[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 [GRA.CON.] [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);
```

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

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

Una breve analisi ci dice che

1. Come previsto dalla teoria, il metodo di Gauss-Seidel converge in approssimativamente metà iterazioni di Jacobi;
2. Il metodo SOR ha quale costante quasi ottimale $w = 1.350$;
3. 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:

```
>> maxit=25; tol=0;
>> siz=5; A = makefish(siz); b=ones(size(A,1),1);
>> [x_gc, error_gc, iter_gc, flag_gc] = cg(A, x, b, M, maxit, tol);
>> error_gc
error_gc =
  3.6287e-039
>>
```

Il residuo relativo, seppur non nullo è molto piccolo.

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 [0, Teor.5.10, p.333]

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

e il raggio spettrale della matrice di iterazione vale $\omega^* - 1$. dove $\rho(S)$ è il massimo degli autovalori in modulo della matrice S (il cosidetto raggio spettrale) e B_J la matrice di iterazione di Jacobi. 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
>> 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
>>
```

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.

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);


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
>>
```

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

$$R(B_J) := -\ln(\rho(B_J)) \tag{8.1}$$
$$R(B_{GS}) := -\ln(\rho(B_{GS})) \tag{8.2}$$
$$R(B_{GS}) := 2R(B_J) \tag{8.3}$$

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

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

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
>>
```

Al momento non consideriamo il metodo del gradiente coniugato poichè non è di tipo stazionario.

9. Facoltativo: Altre matrici interessanti. La matrice di Hilbert.. Per vedere alcuni comandi di base aiutiamoci con delle matrici predefinite in Matlab/Octave. Digitiamo nella shell di Matlab/Octave >> `help elmat`. In Matlab 6.5 abbiamo

```
>> help elmat

  Elementary matrices and matrix manipulation.

  Elementary matrices.
    zeros       - Zeros array.
    ones        - Ones array.
    eye         - Identity matrix.
    repmat      - Replicate and tile array.
    rand        - Uniformly distributed random numbers.
    randn       - Normally distributed random numbers.
    linspace    - Linearly spaced vector.
    logspace    - Logarithmically spaced vector.
    freqspace   - Frequency spacing for frequency response.
    meshgrid    - X and Y arrays for 3-D plots.
    :           - Regularly spaced vector and index into matrix.

  ...

  Specialized matrices.
    compan      - Companion matrix.
```

```
    gallery     - Higham test matrices.
    hadamard    - Hadamard matrix.
    hankel      - Hankel matrix.
    hilb        - Hilbert matrix.
    invhilb     - Inverse Hilbert matrix.
    magic       - Magic square.
    pascal      - Pascal matrix.
    rosser      - Classic symmetric eigenvalue test problem.
    toeplitz    - Toeplitz matrix.
    vander      - Vandermonde matrix.
    wilkinson   - Wilkinson's eigenvalue test matrix.
```

Questo ci dice che Matlab ha predefinito un set di matrici di particolare interesse. Se possibile si suggerisce di provare i metodi che andremo ad introdurre con una matrice facente parte della `gallery` di Matlab. Ciò non appare possibile nelle recenti releases di Octave, come GNU Octave 2.1.73. Da Matlab 6.5

```
>> help gallery

 GALLERY Higham test matrices.
    [out1,out2,...] = GALLERY(matname, param1, param2, ...)
    takes matname, a string that is the name of a matrix family, and
    the family's input parameters. See the listing below for available
    matrix families. Most of the functions take an input argument
    that specifies the order of the matrix, and unless otherwise
    stated, return a single output.
    For additional information, type "help private/matname", where matname
    is the name of the matrix family.

    cauchy   Cauchy matrix.
    chebspec Chebyshev spectral differentiation matrix.
    chebvand Vandermonde-like matrix for the Chebyshev polynomials.
    chow     Chow matrix -- a singular Toeplitz lower Hessenberg matrix.
    circul   Circulant matrix.

    ...

    poisson  Block tridiagonal matrix from Poisson's equation (sparse).
    prolate  Prolate matrix -- symmetric, ill-conditioned Toeplitz matrix.
    randcolu Random matrix with normalized cols and specified singular
values.
    randcorr Random correlation matrix with specified eigenvalues.
    randhess Random, orthogonal upper Hessenberg matrix.
    rando    Random matrix with elements -1, 0 or 1.
    randsvd  Random matrix with pre-assigned singular values and specified
             bandwidth.
    redheff  Matrix of 0s and 1s of Redheffer.
    riemann  Matrix associated with the Riemann hypothesis.
    ris      Ris matrix -- a symmetric Hankel matrix.
    smoke    Smoke matrix -- complex, with a "smoke ring" pseudospectrum.
    toeppd   Symmetric positive definite Toeplitz matrix.
    toeppen  Pentadiagonal Toeplitz matrix (sparse).
    tridiag  Tridiagonal matrix (sparse).
```

```
    triw     Upper triangular matrix discussed by Wilkinson and others.
    wathen   Wathen matrix -- a finite element matrix (sparse, random
entries).
    wilk     Various specific matrices devised/discussed by Wilkinson.
             (Two output arguments)

    GALLERY(3) is a badly conditioned 3-by-3 matrix.
    GALLERY(5) is an interesting eigenvalue problem.  Try to find
    its EXACT eigenvalues and eigenvectors.

    See also MAGIC, HILB, INVHILB, HADAMARD, WILKINSON, ROSSER, VANDER.
```

10. Facoltativo: gli esempi visti in Matlab funzionano anche in Octave.. Rivediamo gli esperimenti in una recente release di Octave, come GNU Octave 2.1.73.

```
octave:12> makefish(3)
ans =

   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

octave:13> A=makefish(5);
octave:14> m=min(eig(A))
m = 0.53590
octave:15> cond(A)
ans = 13.928
octave:16> b=ones(size(A,1),1);
octave:17> demo_algebra_lineare


 [JACOBI  ] [STEP REL., NORMA 2]: 8.73e-09 [REL.ERR.]: 5.65e-08
            [ITER.]: 116 [FLAG]: 0

 [GAU.SEI.] [STEP REL., NORMA 2]: 9.22e-09 [REL.ERR.]: 2.76e-08
            [ITER.]:  61 [FLAG]: 0

 [SOR OTT.] [STEP REL., NORMA 2]: 2.31e-09 [REL.ERR.]: 1.10e-09
            [ITER.]:  21 [FLAG]: 0 [w]: 1.350

 [GRA.CON.] [STEP REL., NORMA 2]: 4.67e-17 [REL.ERR.]: 1.85e-16
            [ITER.]:   5 [FLAG]: 0
octave:18> format long;
octave:19> D=diag(diag(A));
octave:20> size(D)
ans =
```

```
  25   25
octave:21> BJ=eye(size(A))-inv(D)*A;
octave:22> s=eig(BJ);
octave:23> s_abs=abs(s);
octave:24> rho=max(s_abs);
octave:25> w=2/(1+sqrt(1-rho^2))
w = 1.33333333333333
octave:26> maxit=50; tol=10^(-8);
octave:27> b=ones(size(A,1),1);
octave:28> [x_sor,error_sor,iter_sor,flag_sor]=sor(A,x,b,w,maxit,tol);
octave:29> iter_sor
iter_sor = 22
octave:30> raggispettrali

         [RAGGIO SPETTRALE][JACOBI]: 0.866025403784439
         [RAGGIO SPETTRALE][GAUSS-SEIDEL]: 0.750000000000000
         [RAGGIO SPETTRALE][SOR BEST]: 0.350000000000000
         [RAGGIO SPETTRALE][SOR OPT]: 0.333333380472264
octave:31>  0.866025403784439^2
ans = 0.750000000000001
octave:32>
```

RIFERIMENTI BIBLIOGRAFICI

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