

Capitolo 3

Calcolo matriciale: metodi iterativi

Nella risoluzione di sistemi di equazioni lineari la scelta dell'algoritmo risolutivo più efficace richiede un'analisi delle caratteristiche del sistema, ad esempio numero e tipo delle equazioni. In molte applicazioni i sistemi sono di **grandi dimensioni (di ordine elevato)**, cioè con un grande numero di equazioni e quindi di incognite, e **sparsi**, cioè i coefficienti di molte incognite sono uguali a zero (questo significa che ogni equazione coinvolge solo un piccolo numero di incognite).

♣ **Esempio 3.1.** Si consideri un oggetto bidimensionale. Si assuma per semplicità che l'oggetto sia quadrato e sia suddiviso in $n = d^2$ quadrati (vedi Figura 3.1, in cui $d = 8$), detti *pixel*, numerati da 1 a n . Una sorgente di raggi-X emette un fascio di raggi che, dopo aver attraversato l'oggetto, è intercettato da un rivelatore che ne misura l'assorbimento totale da parte dell'oggetto. Si vuole calcolare il coefficiente di assorbimento di ogni pixel in cui è stato suddiviso l'oggetto.

Tale tecnica, detta *tomografia*, è utilizzata in medicina diagnostica; l'oggetto è una sezione piana del corpo umano e viene irradiata con raggi-X al fine di scoprire eventuali anomalie organiche (radiografia, TAC, ...).

Per ogni raggio che attraversa l'oggetto l'assorbimento totale del raggio è dato dalla somma degli assorbimenti nei pixel. Indicato con b_i l'assorbimento totale del raggio i -mo, misurato dal rivelatore, con x_j , $j = 1, \dots, n$, il coefficiente di assorbimento del j -mo pixel (incognite del problema) e con a_{ij} il peso del contributo del j -mo pixel all'assorbimento totale dell' i -mo raggio, si ha:

$$a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n = b_i . \quad (3.1)$$

Se n è il numero di raggi proiettati, si ottiene un sistema di n equazioni in n incognite, che nel seguito si chiamerà sistema PIXEL, la cui soluzione costituisce l'insieme dei coefficienti di assorbimento dei pixel in cui è stato suddiviso l'oggetto.

Supponendo che i raggi siano proiettati perpendicolarmente (vedi Figura 3.1), ogni raggio attraversa solo d pixel. Di conseguenza, la generica equazione (3.1) del sistema PIXEL ha almeno $n - d$ coefficienti a_{ij} uguali a zero, cioè quelli relativi ai pixel non attraversati dall' i -mo raggio e che, quindi, non danno contributo al suo assorbimento. Il numero totale di coefficienti nulli del sistema PIXEL è quindi $n \times (n - d)$. Una misura di "quanto è sparso" il sistema PIXEL si ha considerando il rapporto tra il numero di coefficienti nulli e il numero complessivo dei coefficienti ($n \times n$), cioè:

$$SP = \frac{n(n - d)}{n^2} = 1 - \frac{d}{n}.$$

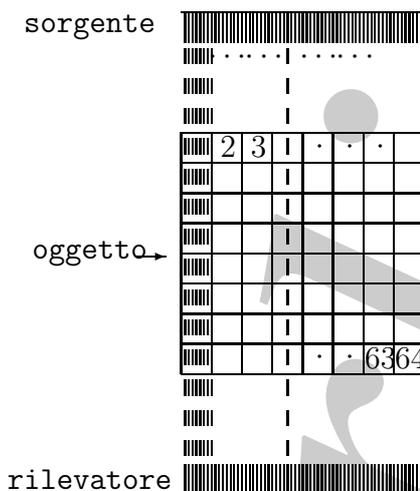


Figura 3.1: Analisi della struttura interna di un oggetto mediante proiezione di raggi-X.

d	n^2	$n(n - d)$	SP
16	65536	61440	0.937
32	1048576	1015808	0.968
64	16777216	16515072	0.984
128	268435456	266338304	0.992

Tabella 3.1: Grado di sparsità del sistema PIXEL al variare di d

Nel caso considerato in Figura 3.1 si ha:

$$\begin{aligned}
 d &= 8 \\
 n &= d^2 = 64 \\
 n \times (n - d) &= 3584 \\
 n \times n &= 4096
 \end{aligned}$$

e, quindi, $SP = 0.875$. In altri termini, il numero di coefficienti nulli del sistema PIXEL è l'87.5% del numero complessivo di coefficienti. In Tabella 3.1 sono riportati i valori di SP al variare di d tra 16 e 128. Come si aspettava, la "sparsità" del sistema PIXEL aumenta al crescere di d e quindi delle sue dimensioni. Si osserva, infine, che, nella pratica, per effettuare un'analisi sufficientemente accurata della struttura interna dell'oggetto, è necessario considerare sia un numero di pixel elevato sia un numero di raggi elevato. In generale $n > 10^5$ e, di conseguenza, il sistema di equazioni risultante è da considerarsi di grandi dimensioni e con sparsità elevata.



L'esempio mostra come, in un'applicazione concreta, le caratteristiche del modello nu-

merico utilizzato per descrivere il problema da risolvere e la necessità di ottenere risultati ragionevolmente accurati, possano condurre a sistemi lineari di grandi dimensioni. Inoltre, tali sistemi possono essere sparsi. A tale riguardo, si osservi che si è introdotto in maniera intuitiva un parametro per misurare “quanto è sparso” un sistema di equazioni lineari. Si può ora dare la seguente:

Definizione 3.1. (Grado di sparsità)

Dato un sistema lineare di n equazioni in n incognite, sia p il numero dei coefficienti diversi da zero. Si definisce **grado di sparsità** del sistema la quantità:

$$SP = \frac{n^2 - p}{n^2} = 1 - \frac{p}{n^2}.$$

Si ha $0 \leq SP \leq 1$ ¹.

Se il sistema lineare è di grandi dimensioni può risultare poco efficiente o addirittura “impraticabile” utilizzare per la sua risoluzione un metodo diretto, come ad esempio il metodo di eliminazione di Gauss. Infatti, il metodo di eliminazione di Gauss richiede un numero di operazioni floating-point tale che, per sistemi con $n \geq 10^4$, il tempo totale di CPU relativo alla risoluzione del sistema su un calcolatore attuale può essere anche dell’ordine di qualche ora, come mostra la seguente Tabella in cui sono riportati i tempi di CPU al variare di n , supponendo di utilizzare un calcolatore in grado di eseguire 3×10^9 operazioni floating-point al secondo (3 Gflops).

¹In generale, quando si parla di grado di sparsità si fa riferimento ad una matrice (nel caso di un sistema di equazioni, la matrice è quella dei coefficienti del sistema).

Data una matrice A di dimensione $n \times m$, se p è il numero degli elementi non nulli, si ha:

$$SP = \frac{nm - p}{nm} = 1 - \frac{p}{nm}.$$

Se tutti gli elementi di A sono non nulli, allora $p = n \times m$ e si ha $SP = 0$. Se, al contrario, tutti gli elementi di A sono nulli (in tal caso A è la matrice nulla), allora $p = 0$ e $SP = 1$. Quindi, si ha $0 \leq SP \leq 1$.

In particolare, considerando le matrici quadrate ($n = m$) non singolari (cioè con determinante non nullo), una matrice che ha il minor numero di elementi non nulli è quella diagonale. Tale matrice ha, in questa classe di matrici, il massimo grado di sparsità, che è dato da:

$$SP = \frac{n(n-1)}{n^2} = 1 - \frac{1}{n}.$$

Si può affermare che una matrice (o un sistema lineare) ha un elevato grado di sparsità se SP è molto vicino a 1.

Si osservi, infine, che, data una matrice A quadrata di ordine n , se d è il numero di elementi diversi da zero su ciascuna riga (come nella matrice dei coefficienti del sistema PIXEL dell’esempio 3.1), si ha:

$$SP = \frac{n(n-d)}{n^2} = 1 - \frac{d}{n}.$$

n	tempo di CPU
1000	0.1111 sec.
10000	111.1111 sec.
50000	13.8889×10^3 sec. \simeq 3.8580 ore

Inoltre, se il sistema ha anche un grado di sparsità elevato è importante utilizzare metodi *ad hoc* che sfruttino, da un punto di vista della complessità computazionale, il fatto che la maggior parte dei coefficienti sono nulli. Anche in tale ottica, può risultare svantaggioso risolvere il sistema con un metodo diretto. Per chiarire questa considerazione si supponga di voler risolvere un sistema di 10 equazioni in 10 incognite. Il metodo di eliminazione di Gauss applicato a tale sistema richiede circa 400 operazioni floating-point per ottenere la soluzione. Se il sistema ha solo 2 coefficienti non nulli per ciascuna equazione, il numero totale di coefficienti diversi da zero è 20, cioè la quinta parte del numero complessivo di coefficienti del sistema. Sarebbe, quindi, desiderabile ottenere la soluzione con un numero di operazioni uguale a $400/5$. Sfortunatamente, la complessità del metodo di eliminazione di Gauss è, in generale, sempre la stessa, anche se il sistema presenta molti coefficienti uguali a zero. Ciò è dovuto al fatto che il metodo di eliminazione di Gauss modifica i coefficienti del sistema², durante il processo di trasformazione del sistema dato in un sistema triangolare equivalente.

♣ **Esempio 3.2.** Si consideri il seguente sistema lineare di 3 equazioni in 3 incognite:

$$\begin{cases} 3x_1 + 4x_2 + x_3 = 7 \\ 2x_1 + \quad + 4x_3 = 6 \\ x_1 + 2x_2 = 3. \end{cases}$$

Dopo il primo passo del metodo di eliminazione di Gauss tale sistema è trasformato nel sistema equivalente:

$$\begin{cases} 3x_1 + 4x_2 + x_3 = 7 \\ -8/3x_2 + 10/3x_3 = 4/3 \\ 2/3x_2 - 1/3x_3 = 2/3. \end{cases}$$

Si osserva che i coefficienti a_{22} e a_{33} , inizialmente nulli, sono divenuti non nulli (rispettivamente $-8/3$ e $-1/3$) dopo l'applicazione del primo passo del metodo. ♣

Il metodo di eliminazione di Gauss, almeno nella sua forma classica³, non “sfrutta” il

²A causa di tali modifiche può accadere che coefficienti uguali a zero diventino non nulli. Tale fenomeno è detto **fill-in**.

³È importante ricordare che per alcune classi di sistemi di equazioni lineari sparsi, in cui i coefficienti non nulli sono disposti secondo un preciso schema (in tal caso si parla di **sistemi sparsi strutturati**), esistono opportune versioni del metodo di eliminazione di Gauss o, in generale, opportuni metodi diretti (ad esempio il metodo di Cholesky per sistemi lineari a banda simmetrici definiti positivi), che consentono di minimizzare il fenomeno del fill-in e, di conseguenza, di ridurre la complessità computazionale.

fatto che il sistema ha molti coefficienti nulli. Tale inefficienza, che è di tutti i metodi diretti, è ancora più evidente per sistemi sparsi di dimensioni maggiori rispetto a quelle del sistema dell'esempio 3.2.

Quando il metodo di eliminazione di Gauss richiede troppo spazio e/o troppo tempo, una valida alternativa è costituita dai **metodi iterativi**. Essi consentono di sfruttare l'eventuale sparsità del sistema perché non modificano i suoi coefficienti. Si analizzano perciò le caratteristiche fondamentali dei metodi iterativi e gli aspetti di base per un loro utilizzo efficiente. In particolare, tale analisi è svolta, inizialmente, attraverso l'esame di due tra i più noti ed antichi metodi iterativi, il **metodo di Jacobi** ed il **metodo di Gauss-Seidel**.

3.1 I metodi di Jacobi e Gauss-Seidel

♣ **Esempio 3.3.** Si consideri il seguente sistema lineare di 3 equazioni in 3 incognite:

$$\begin{cases} 8x_1 - x_2 + 2x_3 = 56 \\ x_1 - 4x_2 + x_3 = -1 \\ x_1 + 2x_2 - 5x_3 = -37. \end{cases} \quad (3.2)$$

Per risolvere tale sistema si procede nel modo seguente. Si riscrive il sistema (3.2) in modo che a primo membro della i -ma equazione, $i = 1, 2, 3$, compaia solo l'incognita x_i . Questa semplice "riscrittura" può sempre essere fatta⁴, eventualmente, modificando l'ordine delle equazioni:

$$\begin{cases} 8x_1 = x_2 - 2x_3 + 56 \\ -4x_2 = -x_1 - x_3 - 1 \\ -5x_3 = -x_1 - 2x_2 - 37. \end{cases} \quad (3.3)$$

Si dividono ambo i membri della i -ma equazione del sistema (3.3), $i = 1, 2, 3$, per il coefficiente di x_i . Il sistema (3.2) è quindi trasformato nel sistema equivalente:

$$\begin{cases} x_1 = \frac{1}{8}(x_2 - 2x_3 + 56) \\ x_2 = -\frac{1}{4}(-x_1 - x_3 - 1) \\ x_3 = -\frac{1}{5}(-x_1 - 2x_2 - 37). \end{cases} \quad (3.4)$$

Questa rappresentazione del sistema è il punto di partenza per costruire un processo iterativo: infatti, se si scelgono ad arbitrio dei valori per le incognite x_1 , x_2 e x_3 utilizzandoli nel secondo membro delle equazioni (3.4), si ottengono nuovi valori per x_1 , x_2 e x_3 . Indicati con $x_1^{(0)}$, $x_2^{(0)}$ e $x_3^{(0)}$ i valori iniziali scelti si ha:

$$\begin{cases} x_1^{(1)} = \frac{1}{8}(x_2^{(0)} - 2x_3^{(0)} + 56) \\ x_2^{(1)} = -\frac{1}{4}(-x_1^{(0)} - x_3^{(0)} - 1) \\ x_3^{(1)} = -\frac{1}{5}(-x_1^{(0)} - 2x_2^{(0)} - 37). \end{cases}$$

⁴Se il sistema è non singolare.

Il passo successivo consiste nell'utilizzare nel secondo membro delle equazioni (3.4) i nuovi valori $x_1^{(1)}$, $x_2^{(1)}$ e $x_3^{(1)}$ al posto dei valori iniziali corrispondenti, cioè:

$$\begin{cases} x_1^{(2)} = \frac{1}{8}(x_2^{(1)} - 2x_3^{(1)} + 56) \\ x_2^{(2)} = -\frac{1}{4}(-x_1^{(1)} - x_3^{(1)} - 1) \\ x_3^{(2)} = -\frac{1}{5}(-x_1^{(1)} - 2x_2^{(1)} - 37). \end{cases}$$

In generale, se $x_1^{(k)}$, $x_2^{(k)}$ e $x_3^{(k)}$ sono i valori delle incognite x_1 , x_2 e x_3 ottenuti dopo k passi, si calcolano nuovi valori $x_1^{(k+1)}$, $x_2^{(k+1)}$ e $x_3^{(k+1)}$ mediante le equazioni:

$$\begin{cases} x_1^{(k+1)} = \frac{1}{8}(x_2^{(k)} - 2x_3^{(k)} + 56) \\ x_2^{(k+1)} = -\frac{1}{4}(-x_1^{(k)} - x_3^{(k)} - 1) \\ x_3^{(k+1)} = -\frac{1}{5}(-x_1^{(k)} - 2x_2^{(k)} - 37). \end{cases} \quad (3.5)$$

Il procedimento descritto è, quindi, di tipo iterativo; a partire da valori iniziali arbitrari delle incognite ($x_1^{(0)}$, $x_2^{(0)}$, $x_3^{(0)}$), si esegue una sequenza di passi, detti **iterazioni**, in ognuno dei quali si calcolano nuovi valori per le incognite “sostituendo” i valori calcolati al passo precedente. Si noti inoltre che ad ogni iterazione i coefficienti del sistema (3.4) non cambiano e sono semplicemente legati in maniera opportuna ai coefficienti del sistema iniziale (3.2).

Le (3.5) definiscono la generica iterazione del procedimento descritto, noto con il nome di **metodo di Jacobi**. Nella Tabella 3.2 sono riportati i valori⁵, arrotondati alla sesta cifra decimale, ottenuti nelle prime 9 iterazioni del metodo di Jacobi, avendo scelto come valori iniziali $x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = 0$. La soluzione del sistema (3.2) è $x_1 = 5.000$, $x_2 = 4.000$ e $x_3 = 10.000$. Si osserva che già dopo 5 iterazioni si hanno valori che sono sufficientemente vicini alla soluzione. ♣

Si consideri il seguente sistema lineare di n equazioni in n incognite:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2 \\ \dots \quad \dots \quad \dots \quad \dots = \dots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n. \end{cases} \quad (3.6)$$

Per risolvere tale sistema si procede nel modo seguente. Si riscrive il sistema (3.6) in modo che a primo membro della i -ma equazione, $i = 1, \dots, n$, compaia solo l'incognita

⁵ Tali valori e tutti i risultati sperimentali riportati in seguito si riferiscono ad elaborazioni effettuate su un processore Pentium IV con le seguenti caratteristiche hardware:

- Aritmetica Standard IEEE-754
- 1500 MHz clock
- 256 KB cache
- 512 MB RAM memory
- 40 GB disk

utilizzando il linguaggio di programmazione Fortran. Si osservi che il sistema aritmetico usato dal processore è quello standard IEEE.

x_i . Questa semplice “riscrittura” può sempre essere fatta⁶, eventualmente modificando l’ordine delle equazioni:

$$\begin{cases} a_{11}x_1 = -a_{12}x_2 - a_{13}x_3 - \dots + b_1 \\ a_{22}x_2 = -a_{21}x_1 - a_{23}x_3 - \dots + b_2 \\ \dots & \dots & \dots & \dots & \dots \\ a_{kk}x_k = -a_{k1}x_1 - a_{k2}x_2 - \dots + b_k \\ \dots & \dots & \dots & \dots & \dots \\ a_{nn}x_n = -a_{n1}x_1 - a_{n2}x_2 - \dots + b_n. \end{cases} \quad (3.7)$$

Si dividono ambo i membri della i -ma equazione del sistema (3.7), $i = 1, \dots, n$, per il coefficiente di x_i . Il sistema (3.6) è quindi trasformato nel sistema equivalente:

$$\begin{cases} x_1 = -\frac{1}{a_{11}}(a_{12}x_2 + a_{13}x_3 + \dots - b_1) \\ x_2 = -\frac{1}{a_{22}}(a_{21}x_1 + a_{23}x_3 + \dots - b_2) \\ \dots & \dots & \dots & \dots & \dots \\ x_k = -\frac{1}{a_{kk}}(a_{k1}x_1 + a_{k2}x_2 + \dots - b_k) \\ \dots & \dots & \dots & \dots & \dots \\ x_n = -\frac{1}{a_{nn}}(a_{n1}x_1 + a_{n2}x_2 + \dots - b_n). \end{cases} \quad (3.8)$$

Questa rappresentazione del sistema è il punto di partenza per costruire un processo iterativo: infatti, se si scelgono ad arbitrio dei valori per le incognite x_1, x_2, \dots, x_n , utilizzandoli nel secondo membro delle equazioni (3.8), si ottengono nuovi valori per x_1, x_2, \dots, x_n . Indicati con $x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)}$ i valori iniziali scelti si ha:

$$\begin{cases} x_1^{(1)} = -\frac{1}{a_{11}}(a_{12}x_2^{(0)} + a_{13}x_3^{(0)} + \dots - b_1) \\ x_2^{(1)} = -\frac{1}{a_{22}}(a_{21}x_1^{(0)} + a_{23}x_3^{(0)} + \dots - b_2) \\ \dots & \dots & \dots & \dots & \dots \\ x_k^{(1)} = -\frac{1}{a_{kk}}(a_{k1}x_1^{(0)} + a_{k2}x_2^{(0)} + \dots - b_k) \\ \dots & \dots & \dots & \dots & \dots \\ x_n^{(1)} = -\frac{1}{a_{nn}}(a_{n1}x_1^{(0)} + a_{n2}x_2^{(0)} + \dots - b_n) \end{cases}$$

Il passo successivo consiste nell’utilizzare, nel secondo membro delle equazioni (3.8), i nuovi valori $x_1^{(1)}, x_2^{(1)}, \dots, x_n^{(1)}$ al posto dei valori iniziali corrispondenti, cioè:

$$\begin{cases} x_1^{(2)} = -\frac{1}{a_{11}}(a_{12}x_2^{(1)} + a_{13}x_3^{(1)} + \dots - b_1) \\ x_2^{(2)} = -\frac{1}{a_{22}}(a_{21}x_1^{(1)} + a_{23}x_3^{(1)} + \dots - b_2) \\ \dots & \dots & \dots & \dots & \dots \\ x_k^{(2)} = -\frac{1}{a_{kk}}(a_{k1}x_1^{(1)} + a_{k2}x_2^{(1)} + \dots - b_k) \\ \dots & \dots & \dots & \dots & \dots \\ x_n^{(2)} = -\frac{1}{a_{nn}}(a_{n1}x_1^{(1)} + a_{n2}x_2^{(1)} + \dots - b_n). \end{cases}$$

⁶Se il sistema è non singolare.

In generale, se $x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)}$ sono i valori delle incognite x_1, x_2, \dots, x_n ottenuti dopo k passi, si calcolano i nuovi valori $x_1^{(k+1)}, x_2^{(k+1)}, \dots, x_n^{(k+1)}$ mediante le equazioni:

$$\begin{cases} x_1^{(k+1)} = -\frac{1}{a_{11}}(a_{12}x_2^{(k)} + a_{13}x_3^{(k)} + \dots - b_1) \\ x_2^{(k+1)} = -\frac{1}{a_{22}}(a_{21}x_1^{(k)} + a_{23}x_3^{(k)} + \dots - b_2) \\ \dots \dots \dots \dots \dots \dots \\ x_k^{(k+1)} = -\frac{1}{a_{kk}}(a_{k1}x_1^{(k)} + a_{k2}x_2^{(k)} + \dots - b_k) \\ \dots \dots \dots \dots \dots \dots \\ x_n^{(k+1)} = -\frac{1}{a_{nn}}(a_{n1}x_1^{(k)} + a_{n2}x_2^{(k)} + \dots - b_n). \end{cases} \quad (3.9)$$

Il procedimento descritto è, quindi, di tipo iterativo; a partire da valori iniziali arbitrari, delle incognite $(x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)})$, si esegue una sequenza di passi, detti **iterazioni**, in ognuno dei quali si calcolano nuovi valori per le incognite “sostituendo” i valori calcolati al passo precedente. Si noti inoltre che ad ogni iterazione i coefficienti del sistema (3.8) non cambiano e sono semplicemente legati in maniera opportuna ai coefficienti del sistema iniziale (3.6).

Le (3.9) definiscono la generica iterazione del procedimento descritto, noto con il nome di **metodo di Jacobi**.

Nella notazione utilizzata, gli indici in alto si riferiscono al numero di iterazioni; in altre parole, $x_1^{(k+1)}, x_2^{(k+1)}, x_n^{(k+1)}$ sono i valori ottenuti alla $(k + 1)$ -ma iterazione utilizzando i valori $x_1^{(k)}, x_2^{(k)}$ e $x_n^{(k)}$ calcolati nell'iterazione precedente.

♣ **Esempio 3.4.** Riprendiamo l'esempio 3.3. Considerando le (3.5), si osserva che quando $x_1^{(k+1)}$ è stato calcolato, è possibile utilizzare tale valore al posto di $x_1^{(k)}$, per determinare il valore $x_2^{(k+1)}$. Analogamente, per calcolare $x_3^{(k+1)}$ è possibile utilizzare i valori di x_1 e x_2 già calcolati alla $(k + 1)$ -ma iterazione. Quanto detto conduce al seguente metodo iterativo:

$$\begin{cases} x_1^{(k+1)} = \frac{1}{8}(x_2^{(k)} - 2x_3^{(k)} + 56) \\ x_2^{(k+1)} = -\frac{1}{4}(-x_1^{(k+1)} - x_3^{(k)} - 1) \\ x_3^{(k+1)} = -\frac{1}{5}(-x_1^{(k+1)} - 2x_2^{(k+1)} - 37), \end{cases} \quad (3.10)$$

noto con il nome di **metodo di Gauss-Seidel (G-S)**.

Nella Tabella 3.3 sono riportati i valori ottenuti nelle prime 6 iterazioni del metodo di G-S, scegliendo come valori iniziali $x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = 0$.

Si osserva che i valori ottenuti con il metodo di G-S si avvicinano alla soluzione più rapidamente, cioè con un minore numero di iterazioni, di quelli ottenuti con il metodo di Jacobi. Tale fenomeno si verifica spesso, ma non sempre: infatti, per alcuni sistemi di equazioni lineari, il metodo di Jacobi genera valori che si avvicinano alla soluzione del sistema, mentre ciò non accade per il metodo di G-S. ♣

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$
1	7.000000	0.250000	7.400000
2	5.181250	3.850000	8.900000
3	5.256250	3.770312	9.976250
4	4.977227	4.058125	9.959375
5	5.017422	3.984150	10.018700
6	4.993345	4.009029	9.997145
7	5.001842	3.997622	10.002280
8	4.999133	4.001031	9.999417
9	5.000275	3.999638	10.000240
\vdots	\vdots	\vdots	\vdots

Tabella 3.2: Valori ottenuti nelle prime 9 iterazioni del metodo di Jacobi applicato al sistema (3.2)

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$
1	7.000000	2.000000	9.600000
2	4.850000	3.862500	9.915000
3	5.004063	3.979766	9.992719
4	4.999291	3.998003	9.999060
5	4.999986	3.999761	9.999902
6	4.999995	3.999974	9.999989
\vdots	\vdots	\vdots	\vdots

Tabella 3.3: Valori ottenuti nelle prime 6 iterazioni del metodo di G-S applicato al sistema (3.2)

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$
1	1.000000	-1.000000	-3.000000
2	5.000000	-3.000000	-3.000000
3	1.000000	1.000000	1.000000
4	1.000000	1.000000	1.000000
\vdots	\vdots	\vdots	\vdots

Tabella 3.4: Valori ottenuti nelle prime 4 iterazioni del metodo di Jacobi applicato al sistema (3.12)

Riprendiamo il sistema (3.9). Si osserva che quando $x_1^{(k+1)}$ è stato calcolato, è possibile utilizzare tale valore al posto di $x_1^{(k)}$, per determinare il valore $x_2^{(k+1)}$. Analogamente, per calcolare $x_n^{(k+1)}$ è possibile utilizzare i valori di $x_1^{(k+1)}, x_2^{(k+1)}, \dots, x_{n-1}^{(k+1)}$ già calcolati alla $(k + 1)$ -ma iterazione. Quanto detto conduce al seguente metodo iterativo:

$$\begin{cases} x_1^{(k+1)} = -\frac{1}{a_{11}}(a_{12}x_2^{(k)} + a_{13}x_3^{(k)} + \dots - b_1) \\ x_2^{(k+1)} = -\frac{1}{a_{22}}(a_{21}x_1^{(k+1)} + a_{23}x_3^{(k)} + \dots - b_2) \\ \dots \\ x_n^{(k+1)} = -\frac{1}{a_{nn}}(a_{n1}x_1^{(k+1)} + a_{n2}x_2^{(k+1)} + \dots - b_n), \end{cases} \quad (3.11)$$

noto con il nome di **metodo di Gauss-Seidel (G-S)**.

♣ **Esempio 3.5.** Si consideri il sistema lineare:

$$\begin{cases} x_1 - 2x_2 + 2x_3 = 1 \\ -x_1 + x_2 - x_3 = -1 \\ -2x_1 - 2x_2 + x_3 = -3, \end{cases} \quad (3.12)$$

la cui soluzione è $x_i = 1, i = 1, 2, 3$.

Nelle Tabelle 3.4 e 3.5 sono riportati alcuni valori ottenuti risolvendo tale sistema con i metodi di Jacobi e di G-S. Si nota che i valori relativi al metodo di Jacobi si avvicinano alla soluzione (nel caso particolare i valori sono proprio uguali alla soluzione) già dopo 3 iterazioni, mentre i valori generati dal metodo di G-S si allontanano sempre di più ad ogni iterazione.



k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$
1	1.000000	0.000000	-1.000000
2	3.000000	1.000000	5.000000
3	-7.000000	-3.000000	-23.000000
4	41.000000	17.000000	113.000000
5	-191.000000	-79.000000	-543.000000
6	929.000000	385.000000	2625.000000
\vdots	\vdots	\vdots	\vdots

Tabella 3.5: Valori ottenuti nelle prime 6 iterazioni del metodo di G-S applicato al sistema (3.12)

Si consideri ora un sistema lineare di n equazioni in n incognite:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2 \\ \dots \quad \dots \quad \dots \quad \dots = \dots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n, \end{cases}$$

che può essere scritto anche nelle due forme equivalenti:

$$\sum_{j=1}^n a_{ij}x_j = b_i, \quad i = 1, \dots, n, \quad (3.13)$$

$$Ax = b,$$

con $A = [a_{ij}]^7$ matrice di dimensione $n \times n$, $b = [b_i]$ e $x = [x_i]$ vettori di dimensione n . In relazione a tale sistema, le (3.5) si generalizzano come segue:

METODO DI JACOBI

$$\begin{aligned} x_i^{(k+1)} &= \left(b_i - a_{i1}x_1^{(k)} - \dots - a_{i,i-1}x_{i-1}^{(k)} - a_{i,i+1}x_{i+1}^{(k)} - \dots - a_{in}x_n^{(k)} \right) / a_{ii} = \\ &= \left(b_i - \sum_{j=1, j \neq i}^n a_{ij}x_j^{(k)} \right) / a_{ii}, \quad i = 1, \dots, n; \quad k \geq 0. \end{aligned} \quad (3.14)$$

⁷Si assume qui e nel seguito che la matrice dei coefficienti $A \in \mathbb{R}^{n \times n}$, del sistema (3.13) sia non singolare, cioè con determinante diverso da zero, e, quindi, che il sistema (3.13) ammetta una ed una sola soluzione.

Ovviamente, affinché le (3.14) siano definite è necessario che gli elementi diagonali di A , a_{ii} , $i = 1, \dots, n$, siano diversi da zero⁸.

In relazione al sistema (3.13), il metodo di G-S è definito da:

METODO DI GAUSS-SEIDEL

$$x_i^{(k+1)} = \left(b_i - a_{i1}x_1^{(k+1)} - \dots - a_{i,i-1}x_{i-1}^{(k+1)} - a_{i,i+1}x_{i+1}^{(k)} - \dots - a_{in}x_n^{(k)} \right) / a_{ii} =$$

$$= \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij}x_j^{(k)} \right) / a_{ii}, \quad i = 1, \dots, n; \quad k \geq 0.$$

(3.15)

Come per il metodo di Jacobi, si assume che gli elementi diagonali di A siano diversi da zero.

3.1.1 Primi esempi di algoritmi

Un primo esempio di algoritmo per la risoluzione di un sistema di equazioni lineari con il metodo iterativo di Jacobi è illustrato nella **Procedura 3.1**. Essa è scritta nel linguaggio *pascal-like* e prevede come dati di input la dimensione del sistema n , la matrice dei coefficienti del sistema, rappresentata dall'array A , il vettore dei termini noti, rappresentato dall'array b , i valori iniziali contenuti nell'array $xold$ ed il numero di iterazioni da effettuare, $MaxIt$. I soli dati di output sono i valori ottenuti dopo l'esecuzione di $MaxIt$ iterazioni del metodo di Jacobi, immagazzinati nell'array x . Si osservi che, nella procedura in esame, l'array $xold$ contiene i valori calcolati all'iterazione precedente, mentre l'array x contiene i nuovi valori da calcolare.

Una prima versione dell'algoritmo basato sul metodo di G-S è invece illustrato nella **Procedura 3.2**. Si osservi che in tale procedura, a differenza di quella relativa al metodo di Jacobi, non è necessario utilizzare l'array $xold$, poiché, per calcolare i nuovi valori $x(i)$ alla generica iterazione, si utilizzano i valori $x(j)$, $j < i$, già calcolati in quella stessa iterazione.

⁸Se il sistema (3.13) è non singolare, ciò può essere sempre ottenuto scambiando in modo opportuno le equazioni.

```
procedure Jacobi1(in:  $n, A, b, xold, MaxIt$  ; out:  $x$ )  
  
  /# SCOPO: risoluzione di  $Ax=b$  con il metodo di Jacobi  
  
  /# SPECIFICHE PARAMETRI:  
  /# PARAMETRI DI INPUT:  
    var:  $n$       : intero      {dimensione del sistema}  
    var:  $A(n, n)$  : array di reali {coefficienti del sistema}  
    var:  $b(n)$     : array di reali {termini noti del sistema}  
    var:  $xold(n)$  : array di reali {valori iniziali}  
    var:  $MaxIt$   : intero      {numero di iterazioni da effettuare}  
  
  /# PARAMETRI DI OUTPUT:  
    var:  $x(n)$    : array di reali {soluzione calcolata}  
  
  /# VARIABILI LOCALI:  
    var:  $i, j, k$  : interi      {contatori}
```

Procedura 3.1: Algoritmo di Jacobi (prima versione) - continua

```

/# INIZIO ISTRUZIONI:
  for  $k = 1, MaxIt$  do {ciclo sul numero di iterazioni}
    for  $i = 1, n$  do {ciclo per il calcolo di  $x^{(k)}$ }
       $x(i) := b(i)$ ;
      for  $j = 1, n$  do {ciclo per il calcolo di  $x_i^{(k)}$ }
        if  $i \neq j$  then
           $x(i) := x(i) - A(i, j) * xold(j)$ ;
        endif
      endfor
       $x(i) := x(i) / A(i, i)$ ;
    endfor
    for  $i = 1, n$  do {aggiornamento di  $xold$ }
       $xold(i) := x(i)$ ;
    endfor
  endfor
return  $x$ 
end Jacobi1

```

Procedura 3.1: Algoritmo di Jacobi (prima versione) - fine

```
procedure G-S1(in:  $n, A, b, x, MaxIt$  ; out:  $x$ )  
  
  /# SCOPO: risoluzione di  $Ax=b$  con il metodo di Gauss-Seidel  
  
  /# SPECIFICHE PARAMETRI:  
  /# PARAMETRI DI INPUT:  
    var:  $n$       : intero      {dimensione del sistema}  
    var:  $A(n, n)$  : array di reali {coefficienti del sistema}  
    var:  $b(n)$    : array di reali {termini noti del sistema}  
    var:  $x(n)$    : array di reali {valori iniziali}  
    var:  $MaxIt$   : intero      {numero di iterazioni da effettuare}  
  
  /# PARAMETRI DI OUTPUT:  
    var:  $x(n)$    : array di reali {soluzione calcolata}  
  
  /# VARIABILI LOCALI:  
    var:  $i, j, k$  : interi      {contatori}
```

Procedura 3.2: Algoritmo di G-S (prima versione) - continua

```

/# INIZIO ISTRUZIONI:
  for k = 1, MaxIt do                                {ciclo sul numero di iterazioni}
    for i = 1, n do                                  {ciclo per il calcolo di x^{(k)}}
      x(i) := b(i);
      for j = 1, n do                                {ciclo per il calcolo di x_i^{(k)}}
        if i ≠ j then
          x(i) := x(i) - A(i, j) * x(j);
        endif
      endfor
      x(i) := x(i)/A(i, i);
    endfor
  endfor
return x
end G-S1

```

Procedura 3.2: Algoritmo di G-S (prima versione) - fine

3.1.2 Complessità computazionale

Si analizza ora il numero di operazioni floating-point richieste dal metodo di Jacobi. Considerando la generica iterazione del metodo, si osserva che il calcolo di ciascuna componente di $x^{(k+1)}$ richiede $n - 1$ moltiplicazioni, $n - 2$ addizioni, 1 sottrazione e 1 divisione. Contando una divisione come una moltiplicazione ed una sottrazione come un'addizione, il numero totale di operazioni richieste da un'iterazione del metodo di Jacobi è:

$$n^2\mathcal{M} + n(n - 1)\mathcal{A}, \quad (3.16)$$

dove \mathcal{A} denota un'addizione e \mathcal{M} denota una moltiplicazione e, dunque, una stima del numero di flops, T_{jac} , richieste da una iterazione è

$$T_{jac} \simeq n^2. \quad (3.17)$$

Dalla (3.17) si rileva che il costo di ciascuna iterazione del metodo è equivalente a quello del calcolo del prodotto tra una matrice ed un vettore. Si osservi inoltre che il numero di operazioni richieste è circa uguale al numero dei coefficienti diversi da zero. Nel derivare la stima (3.17) si è supposto che, per ciascuna componente di $x^{(k+1)}$, tutte le operazioni in (3.14) siano effettivamente eseguite. Se invece il sistema lineare da risolvere ha dei coefficienti a_{ij} nulli, le operazioni nelle quali si utilizzano tali coefficienti possono non essere eseguite e, quindi, non devono essere considerate nel calcolo del numero di flops richieste dal metodo di Jacobi. Per chiarire questa considerazione si assuma che

il sistema abbia solo p coefficienti non nulli, con $p < n^2$. Di conseguenza, il calcolo di $x^{(k+1)}$ richiede un numero di flops circa uguale a p . In tal caso, la (3.17) si generalizza nella:

$$T_{jac} \simeq p = n^2(1 - (1 - p/n^2)) = n^2(1 - SP), \quad (3.18)$$

dove SP è il grado di sparsità del sistema. Si osservi che, quando $SP = 0$ (cioè, tutti i coefficienti sono non nulli), dalla (3.18) si ottiene la (3.17). Inoltre, la (3.18) mostra che se il sistema ha un grado di sparsità elevato (che significa $p \ll n^2$ e, quindi, SP molto vicino a 1), il costo di un'iterazione del metodo di Jacobi si riduce notevolmente rispetto al costo per un sistema non sparso. Come già detto ciò rappresenta uno dei vantaggi dell'utilizzo dei metodi iterativi nella risoluzione di sistemi di dimensioni elevate. Su tale aspetto si tornerà nei prossimi paragrafi.

Dalla (3.15), si verifica che tutte le considerazioni appena fatte sulla complessità computazionale del metodo di Jacobi si possono applicare anche al metodo di G-S giungendo alle stesse conclusioni, e quindi:

$$T_{gs} \simeq n^2(1 - SP).$$

3.1.3 Interpretazione geometrica

Si consideri il sistema lineare di due equazioni:

$$\begin{cases} 2x_1 - x_2 = 0 \\ x_1 - 3x_2 = 0, \end{cases} \quad (3.19)$$

ciascuna delle quali, in un sistema di riferimento di assi cartesiani in cui x_1 è l'asse delle ascisse e x_2 è l'asse delle ordinate, rappresenta una retta del piano; il loro punto di intersezione, l'origine, è la soluzione del sistema (vedi Figura 3.2). Il metodo di Jacobi applicato a tale sistema ha la forma:

$$\begin{cases} x_1^{(k+1)} = \frac{1}{2}x_2^{(k)} \\ x_2^{(k+1)} = \frac{1}{3}x_1^{(k)}. \end{cases}$$

Quindi, se $x^{(0)}$ è il punto iniziale, la prima equazione è risolta rispetto alla variabile x_1 con la variabile x_2 fissata. Dalla Figura 3.2 si evince che geometricamente ciò significa determinare sulla retta definita da tale equazione il punto P che ha la stessa ordinata di $x^{(0)}$. Procedendo allo stesso modo per la seconda equazione, si determina sulla retta ad essa relativa il punto Q che ha la stessa ascissa di $x^{(0)}$. Il nuovo punto $x^{(1)}$ si ottiene geometricamente come estremo del vettore somma dei vettori definiti dai punti

$$(x^{(0)}, P) \quad \text{e} \quad (x^{(0)}, Q)$$

così determinati, puntati in $x^{(0)}$. Nelle iterazioni successive si procede in maniera analoga e si osserva che ci si avvicina sempre di più alla soluzione.

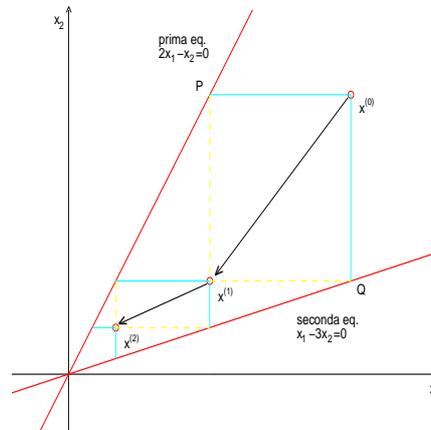


Figura 3.2: Interpretazione geometrica del metodo di Jacobi applicato al sistema (3.19).

Infine, in Figura 3.3 è illustrata l'interpretazione geometrica del metodo di G-S applicato al sistema (3.19):

$$\begin{cases} x_1^{(k+1)} = \frac{1}{2}x_2^{(k)} \\ x_2^{(k+1)} = \frac{1}{3}x_1^{(k+1)}. \end{cases}$$

A partire da $x^{(0)}$, il primo passo è identico a quello di Jacobi. Il punto $x^{(1)}$, invece, si ottiene a partire dal punto determinato sulla prima retta e considerando il punto sulla seconda retta che ha la stessa ascissa. Confrontando le Figure 3.2 e 3.3 si può osservare graficamente che per il sistema in esame il metodo di G-S si avvicina alla soluzione più velocemente del metodo di Jacobi.

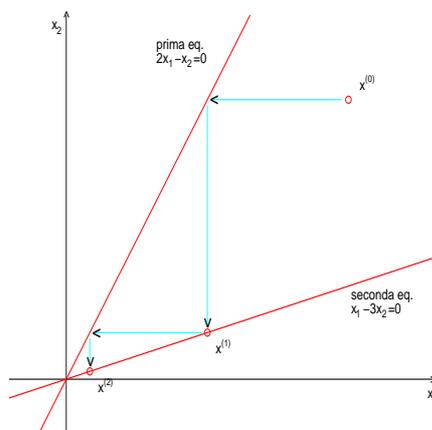


Figura 3.3: Interpretazione geometrica del metodo di G-S applicato al sistema (3.19).

3.2 Convergenza

Si è visto che gli algoritmi relativi ai metodi di Jacobi e di G-S generano n successioni di valori $\{x_i^{(k)}\}$, $i = 1, \dots, n$, mediante le equazioni (3.14) o (3.15). Quindi, il primo problema relativo all'utilizzo di tali algoritmi è la determinazione di condizioni che assicurino che tali valori, al crescere di n , si avvicinino sempre di più alla soluzione del sistema. In altre parole, è necessario analizzare il problema della **convergenza** dei metodi di Jacobi e G-S, da cui questi algoritmi derivano.

Indicata con $x^* = [x_i^*]$ la soluzione del sistema (3.13), si vogliono determinare condizioni opportune per le quali si abbia:

$$\lim_{k \rightarrow \infty} x_i^{(k)} = x_i^*, \quad i = 1, \dots, n.$$

♣ **Esempio 3.6.** Si consideri il sistema lineare:

$$\begin{cases} x_1 + 2x_2 & = 1 \\ x_1 + x_2 + x_3 & = 0 \\ 2x_2 + x_3 & = 1, \end{cases} \quad (3.20)$$

la cui soluzione è $x_1^* = -1/3$, $x_2^* = 2/3$, $x_3^* = -1/3$. Risolvendo tale sistema con i metodi di Jacobi e di G-S, partendo dai valori iniziali $x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = 0$, si hanno i risultati riportati nelle Tabelle 3.6 e 3.7: entrambi i metodi generano valori che non si avvicinano alla soluzione.

♣

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$
1	1.000000	0.000000	1.000000
2	1.000000	-2.000000	1.000000
3	5.000000	-2.000000	5.000000
4	5.000000	-10.000000	5.000000
5	21.000000	-10.000000	21.000000
6	21.000000	-42.000000	21.000000
\vdots	\vdots	\vdots	\vdots
15	21845.00	-10922.00	21845.00
\vdots	\vdots	\vdots	\vdots

Tabella 3.6: Valori ottenuti nelle prime 15 iterazioni del metodo di Jacobi applicato al sistema (3.20).

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$
1	1.000000	-1.000000	3.000000
2	3.000000	-6.000000	13.000000
3	13.000000	-26.000000	53.000000
4	53.000000	-106.000000	213.000000
5	213.000000	-426.000000	853.000000
6	853.000000	-1706.000000	3413.000000
\vdots	\vdots	\vdots	\vdots
15	$2.2 \cdot 10^8$	$-4. \cdot 10^8$	$8.9 \cdot 10^8$
\vdots	\vdots	\vdots	\vdots

Tabella 3.7: Valori ottenuti nelle prime 15 iterazioni del metodo di G-S applicato al sistema (3.20).

♣ **Esempio 3.7.** Si consideri il sistema lineare:

$$\begin{cases} x_1 - 3x_2 = 0 \\ 2x_1 - x_2 = 0, \end{cases} \quad (3.21)$$

ottenuto dal sistema (3.19) semplicemente scambiando le equazioni e la cui soluzione è $x_1^* = 0$, $x_2^* = 0$. Per tale sistema, partendo dai valori iniziali $x_1^{(0)} = x_2^{(0)} = 0.5$, sia il metodo di Jacobi sia il metodo di G-S non convergono. Dopo 10 iterazioni si ha infatti:

$$\begin{aligned} (\text{Jacobi}) \quad & x_1 = 3888; \quad x_2 = 3888, \\ (G-S) \quad & x_1 = 1.51165 \times 10^7; \quad x_2 = 3.02331 \times 10^7. \end{aligned}$$

Tale fenomeno si evidenzia anche nelle Figure 3.4 e 3.5, dove è illustrata l'interpretazione geometrica dei metodi di Jacobi e G-S applicati al sistema in esame. Ricordando che per il sistema (3.19) entrambi i metodi convergono, si deduce che l'ordine delle equazioni può influire sulla convergenza dei metodi in esame (tale questione verrà ripresa nei prossimi paragrafi).

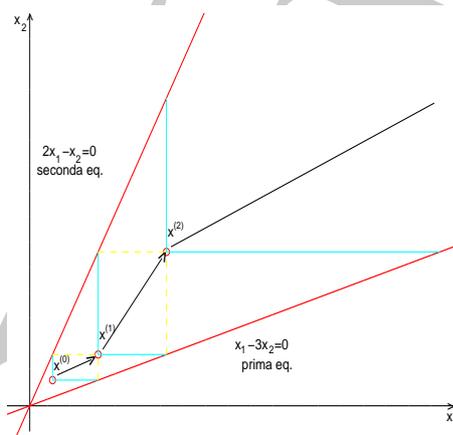


Figura 3.4: Interpretazione geometrica del metodo di Jacobi applicato al sistema (3.21).

♣

Allo scopo di illustrare le idee di base relative allo studio della convergenza dei metodi

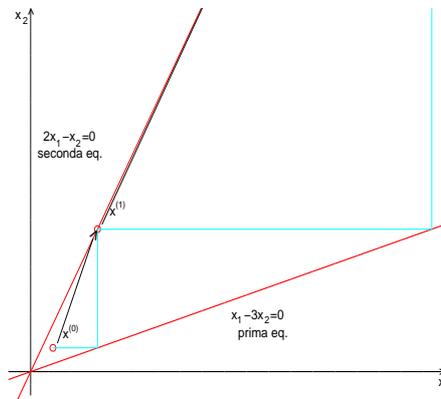


Figura 3.5: Interpretazione geometrica del metodo di G-S applicato al sistema (3.21).

di Jacobi e di G-S si osservi che entrambi i metodi possono essere espressi nella forma:

$$\begin{aligned} x_i^{(k+1)} &= c_{i1}x_1^{(k)} + \cdots + c_{in}x_n^{(k)} + d_i = \\ &= \sum_{j=1}^n c_{ij}x_j^{(k)} + d_i, \quad i = 1, \dots, n, \quad k \geq 0, \end{aligned} \quad (3.22)$$

o equivalentemente:

$$x^{(k+1)} = Cx^{(k)} + d, \quad k \geq 0,$$

dove $C = [c_{ij}]$ è una matrice $n \times n$ e $d = [d_i]$ è un vettore di dimensione n .

È immediato verificare, dalla (3.14), che per il metodo di Jacobi si ha:

$$c_{ij} = \begin{cases} 0 & i = j \\ -a_{ij}/a_{ii} & i \neq j \end{cases}, \quad d_i = b_i/a_{ii}. \quad (3.23)$$

Per il metodo di G-S scritto in forma (3.22)⁹ si ha:

$$c_{ij} = \begin{cases} -(\sum_{l=1}^{i-1} a_{il}c_{lj})/a_{ii}, & j = 1, 2, \dots, i \\ -(\sum_{l=1}^{i-1} a_{il}c_{lj} + a_{ij})/a_{ii}, & j = i+1, \dots, n \end{cases} \quad (3.24)$$

$$d_i = (b_i - \sum_{l=1}^{i-1} a_{il}d_l)/a_{ii}, \quad i = 1, \dots, n.$$

Si osservi che per il metodo di G-S non è conveniente utilizzare l'equazione (3.22) con i coefficienti c_{ij} dati dalle (3.24). Le equazioni (3.15) risultano molto più "semplici". La

⁹Non è altrettanto immediato scrivere il metodo di G-S nella forma (3.22). Tuttavia ciò può essere fatto con un procedimento per induzione. Si osservi innanzitutto che la (3.15) per $i = 1$ è già in forma (3.22) con $c_{11} = 0$ e $c_{1j} = -a_{1j}/a_{11}$, $j = 2, \dots, n$ e $d_1 = b_1/a_{11}$. Si assuma che $x_2^{(k+1)}, \dots, x_{i-1}^{(k+1)}$ dati dalla (3.15) abbiano la forma (3.22). Si dimostra ora che anche $x_i^{(k+1)}$ può essere scritto in forma (3.22).

sola ragione per cui si è introdotta la forma (3.22) è quella di facilitare una discussione elementare sulla convergenza dei metodi iterativi in esame.

Definizione 3.2. (Metodo convergente)

Si dice che un metodo del tipo (3.22) è **convergente** se, qualunque siano i valori iniziali, le successioni $\{x_i^{(k)}\}$, $i = 1, \dots, n$, convergono, cioè se

$$\lim_{k \rightarrow \infty} x_i^{(k)} = \bar{x}_i, \quad i = 1, \dots, n, \quad \forall x_i^{(0)}, \quad (3.25)$$

oppure, in modo completamente equivalente, se:

$$\lim_{k \rightarrow \infty} x^{(k)} = \bar{x}, \quad \forall x^{(0)},$$

dove $x^{(k)}$ e \bar{x} sono i vettori di componenti, rispettivamente, $(x_1^{(k)}, \dots, x_n^{(k)})$ e $(\bar{x}_1, \dots, \bar{x}_n)$.

Si osservi che, se il metodo (3.22) converge, passando al limite in entrambi i membri delle (3.22), si ha:

$$\bar{x}_i = \sum_{j=1}^n c_{ij} \bar{x}_j + d_i, \quad i = 1, \dots, n. \quad (3.26)$$

che, in forma compatta, può scriversi come:

$$\bar{x} = C\bar{x} + d, \quad \text{o equivalentemente } [(I - C)\bar{x} = d]$$

Se ora si considera il metodo di Jacobi e si assume che sia convergente, indicato con \bar{x} il limite, dalle (3.23) si ha:

$$\bar{x}_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1, j \neq i}^n a_{ij} \bar{x}_j \right) = \bar{x}_i + \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^n a_{ij} \bar{x}_j \right), \quad i = 1, \dots, n,$$

Infatti, considerando l'espressione di $x_i^{(k+1)}$ in (3.15), si ha:

$$\begin{aligned} x_i^{(k+1)} &= \frac{1}{a_{ii}} \left[b_i - \sum_{l=1}^{i-1} a_{il} x_l^{(k+1)} - \sum_{l=i+1}^n a_{il} x_l^{(k)} \right] \\ &= \frac{1}{a_{ii}} \left[b_i - \sum_{l=1}^{i-1} a_{il} \left(\sum_{j=1}^n c_{lj} x_j^{(k)} + d_l \right) - \sum_{l=i+1}^n a_{il} x_l^{(k)} \right] \\ &= \sum_{j=1}^i \left(\sum_{l=1}^{i-1} \frac{-a_{il} c_{lj}}{a_{ii}} \right) x_j^{(k)} + \sum_{j=i+1}^n \left(\sum_{l=1}^{i-1} \frac{-a_{il} c_{lj}}{a_{ii}} - \frac{a_{ij}}{a_{ii}} \right) x_j^{(k)} \\ &\quad + \frac{1}{a_{ii}} \left(b_i - \sum_{l=1}^{i-1} a_{il} d_l \right) \end{aligned}$$

da cui segue la (3.24).

e, di conseguenza:

$$\sum_{j=1}^n a_{ij} \bar{x}_j = b_i, \quad i = 1, \dots, n,$$

cioè, il limite è soluzione del sistema $Ax = b$ e, quindi, $\bar{x}_i = x_i^*$, $i = 1, \dots, n$.
Si può perciò affermare:

Proprietà 3.1. *Se il metodo di Jacobi converge, allora il limite è la soluzione del sistema (3.13).*

Con un ragionamento analogo, si può verificare che anche il metodo di G-S gode della proprietà suddetta, che è detta **consistenza** del metodo con il sistema (3.13). È evidente che tale proprietà è essenziale per un qualsiasi metodo iterativo del tipo (3.22).

Definizione 3.3. (Metodo consistente)

*Un metodo del tipo (3.22) è **consistente**¹⁰ con il sistema (3.13) se, quando converge, il limite è la soluzione del sistema (3.13).*

Ad ogni iterazione di un metodo iterativo del tipo (3.22), il vettore $x^{(k)}$ rappresenta un'approssimazione della soluzione del sistema (3.13).

Definizione 3.4. (Errore di troncamento analitico)

Sia $\{x^k\}$ la successione generata dal metodo iterativo:

$$x_i^{(k+1)} = \sum_{j=1}^n c_{ij} x_j^{(k)} + d_i, \quad i = 1, \dots, n, \quad k \geq 0.$$

*Si definisce **errore di troncamento analitico alla k-ma iterazione del metodo**, il vettore:*

$$e^{(k)} = x^{(k)} - x^*. \tag{3.27}$$

In base alla definizione precedente è evidente che la convergenza di un metodo del tipo (3.22) alla soluzione del sistema (3.13) è equivalente alla convergenza della successione degli errori a zero, cioè alla condizione:

$$\lim_{k \rightarrow \infty} |e_i^{(k)}| = 0, \quad i = 1, \dots, n, \tag{3.28}$$

dove $e_i^{(k)} = (x_i^{(k)} - x_i^*)$ è l' i -ma componente di $e^{(k)}$.

¹⁰Nei prossimi paragrafi il concetto di consistenza verrà ripreso con maggiore approfondimento.

♣ **Esempio 3.8.** Si consideri il sistema

$$\begin{cases} 8x_1 - x_2 + 2x_3 = 56 \\ x_1 - 4x_2 + x_3 = -1 \\ x_1 + 2x_2 - 5x_3 = -37. \end{cases} \quad (3.29)$$

Nella Tabella che segue sono riportati i valori assoluti delle componenti dell'errore relativo alle prime 9 iterazioni (vedi Tabella 3.2) risolvendo il sistema con il metodo di Jacobi. Si osserva che tali quantità si avvicinano a zero al crescere del numero di iterazioni, come ci si aspetta sempre quando il metodo è convergente alla soluzione.

k	$ e_1^{(k)} $	$ e_2^{(k)} $	$ e_3^{(k)} $
1	2.000000	3.750000	2.600000
2	0.181250	0.150000	1.100000
3	0.256250	0.229688	0.023750
4	0.022773	0.058125	0.040625
5	0.017422	0.015850	0.018700
6	0.006655	0.009029	0.002855
7	0.001842	0.002378	0.002280
8	0.000867	0.001031	0.000583
9	0.000275	0.000362	0.000239

♣

Per un qualsiasi vettore x di dimensione n e per una qualsiasi matrice A di dimensione $n \times n$, si introducono, inoltre, le quantità:

$$\|x\| = \max(|x_1|, \dots, |x_n|), \quad (3.30)$$

$$\|A\| = \max_{1 \leq i \leq n} \left(\sum_{j=1}^n |a_{ij}| \right), \quad (3.31)$$

le quali rappresentano una misura della grandezza rispettivamente degli elementi del vettore e dei coefficienti della matrice. Tali quantità, che in generale si denotano con $\|x\|_\infty$ e $\|A\|_\infty$, sono la **norma del massimo**¹¹ del vettore x e della matrice A .

¹¹Questa ed altre norme, sia di vettori sia di matrici, sono definite nel §B.3 dell'Appendice B e sono utilizzate spesso nel testo.

♣ **Esempio 3.9.** Si consideri il sistema (3.29). La matrice C relativa al metodo di Jacobi espresso nella forma (3.22) applicato a tale sistema è:

$$C = \begin{bmatrix} 0 & 0.125 & -0.25 \\ 0.25 & 0 & 0.25 \\ 0.2 & 0.4 & 0 \end{bmatrix}.$$

Si ha quindi $\|C\| = 0.6$. ♣

Una semplice condizione sufficiente per la convergenza dei metodi iterativi della forma (3.22) è data dal seguente Teorema:

Teorema 3.1. *Il metodo iterativo:*

$$x_i^{(k+1)} = \sum_{j=1}^n c_{ij} x_j^{(k)} + d_i, \quad i = 1, \dots, n, \quad k \geq 0,$$

è convergente se $\|C\| < 1$.

Dimostrazione Se $\|C\| < 1$, la matrice $I - C$ è non singolare (vedi **Teoremi B.5 e B.12** dell'Appendice **B** della **Parte 1**) e, quindi, il sistema lineare $x = Cx + d$ ha un'unica soluzione, \bar{x} . Posto:

$$e^{(k)} = x^{(k)} - \bar{x}, \quad \forall k \geq 0,$$

dalle (3.22) e dalle (3.26) si ha:

$$e_i^{(k+1)} = x_i^{(k+1)} - \bar{x}_i = \sum_{j=1}^n c_{ij} (x_j^{(k)} - \bar{x}_j) = \sum_{j=1}^n c_{ij} e_j^{(k)}, \quad i = 1, \dots, n, \quad (3.32)$$

da cui:

$$|e_i^{(k+1)}| \leq \sum_{j=1}^n |c_{ij}| |e_j^{(k)}|, \quad i = 1, \dots, n. \quad (3.33)$$

Considerando la (3.30), dalla (3.33) si ha:

$$|e_i^{(k+1)}| \leq \sum_{j=1}^n |c_{ij}| \|e^{(k)}\|, \quad i = 1, \dots, n, \quad (3.34)$$

da cui, utilizzando la (3.31), si ottiene:

$$|e_i^{(k+1)}| \leq \|C\| \|e^{(k)}\|, \quad i = 1, \dots, n,$$

e quindi:

$$\|e^{(k+1)}\| \leq \|C\| \|e^{(k)}\|.$$

Allo stesso modo si può dimostrare che:

$$\|e^{(k)}\| \leq \|C\| \|e^{(k-1)}\|,$$

da cui segue:

$$\|e^{(k+1)}\| \leq \|C\| \|e^{(k)}\| \leq \|C\|^2 \|e^{(k-1)}\|.$$

Ripetendo altre $k - 1$ volte tale procedimento, si ottiene:

$$\|e^{(k+1)}\| \leq \|C\|^{k+1} \|e^{(0)}\|. \quad (3.35)$$

Se $\|C\| < 1$, il secondo membro della (3.35) converge a zero per $k \rightarrow \infty$, qualunque sia $e^{(0)}$. Si ha quindi:

$$\lim_{k \rightarrow \infty} \|e^{(k+1)}\| = 0, \quad \forall e^{(0)},$$

che equivale a:

$$\lim_{k \rightarrow \infty} |e_i^{(k+1)}| = 0, \quad i = 1, \dots, n, \quad \forall e_i^{(0)},$$

da cui:

$$\lim_{k \rightarrow \infty} x_i^{(k)} = \bar{x}_i, \quad i = 1, \dots, n, \quad \forall x_i^{(0)},$$

cioè, il metodo è convergente a \bar{x} . ■

Osservazione 3.1. *Il Teorema precedente stabilisce una condizione solo sufficiente per la convergenza basata sull'ordine di grandezza dei coefficienti c_{ij} in (3.22). Come già detto esistono altre misure della grandezza di tali coefficienti e per tali misure vale il risultato stabilito dal Teorema 3.1¹².*

Si consideri il metodo di Jacobi espresso nella forma (3.22), dove i coefficienti c_{ij} e d_i sono dati dalle (3.23). Utilizzando la (3.31) si ha:

$$\|C\| = \max_{1 \leq i \leq n} \left(\sum_{j=1}^n |c_{ij}| \right) = \max_{1 \leq i \leq n} \left(\sum_{j=1, j \neq i}^n |a_{ij}| / |a_{ii}| \right).$$

Quindi, se:

$$\sum_{j=1, j \neq i}^n |a_{ij}| / |a_{ii}| < 1, \quad i = 1, \dots, n,$$

che è equivalente alla condizione:

$$\sum_{j=1, j \neq i}^n |a_{ij}| < |a_{ii}|, \quad i = 1, \dots, n, \quad (3.37)$$

¹²Ad esempio, se si considera la **norma del valore assoluto** (o **norma 1**):

$$\|x\|_1 = |x_1| + \dots + |x_n| = \sum_{i=1}^n |x_i|,$$

$$\|A\|_1 = \max_{1 \leq j \leq n} \left(\sum_{i=1}^n |a_{ij}| \right), \quad (3.36)$$

si può verificare che vale ancora la relazione (3.35); il Teorema 3.1 è quindi vero anche se si considera la norma suddetta.

si ha $\|C\| < 1$. Un sistema di equazioni lineari in cui i coefficienti a_{ij} soddisfano le (3.37) è detto a **diagonale strettamente dominante**.

Si può quindi enunciare il seguente risultato:

Corollario 3.1. *Se il sistema lineare (3.13) è a diagonale strettamente dominante, il metodo di Jacobi è convergente.*

Poichè i risultati dimostrati in questo paragrafo forniscono condizioni solo sufficienti per la convergenza, un metodo iterativo del tipo (3.22) può convergere anche se il sistema non è a diagonale strettamente dominante o, in generale, non soddisfa l'ipotesi del Teorema 3.1.

♣ **Esempio 3.10.** Considerato il sistema lineare:

$$\begin{cases} x_1 - 2x_2 + 2x_3 = 1 \\ 2x_1 + x_2 + 2x_3 = 1 \\ x_1 + x_2 + x_3 = 1, \end{cases}$$

la cui soluzione è $x_1^* = -3, x_2^* = 1, x_3^* = 3$, è immediato verificare che esso non è a diagonale strettamente dominante. Inoltre, considerato il metodo di Jacobi applicato a tale sistema:

$$\begin{cases} x_1^{(k+1)} = +2x_2^{(k)} - 2x_3^{(k)} + 1 \\ x_2^{(k+1)} = -2x_1^{(k)} - 2x_3^{(k)} + 1 \\ x_3^{(k+1)} = -x_1^{(k)} - x_2^{(k)} + 1, \end{cases}$$

si verifica che i coefficienti c_{ij} del metodo non soddisfano l'ipotesi del Teorema 3.1. Tuttavia, il metodo di Jacobi converge alla soluzione del sistema, come mostrano i valori riportati nella seguente Tabella.

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$
0	1.000000	1.000000	1.000000
1	1.000000	-3.000000	-1.000000
2	-3.000000	1.000000	3.000000
3	-3.000000	1.000000	3.000000
\vdots	\vdots	\vdots	\vdots



3.3 Un semplice criterio di arresto

Un aspetto fondamentale legato all'utilizzo efficace dei metodi iterativi in esame è la scelta di un opportuno criterio per decidere quando arrestare il processo iterativo. In generale, un criterio di arresto soddisfacente deve essere tale che il suo utilizzo conduca ad un risultato sufficientemente accurato, o meglio, consenta di ottenere l'accuratezza desiderata. Appare evidente, quindi, che il criterio più naturale è richiedere che la "distanza" tra $x^{(k)}$ e la soluzione del sistema, e cioè l'errore di troncamento analitico, sia abbastanza piccola. Se si utilizza la norma del massimo per misurare la grandezza dell'errore, una possibile condizione per l'arresto del metodo iterativo è richiedere che l'errore assoluto, $\|x^{(k+1)} - x^*\|$, sia minore di una tolleranza Tol:

$$\|x^{(k+1)} - x^*\| \leq \text{Tol}, \quad (3.38)$$

che rappresenta l'accuratezza richiesta.

♣ **Esempio 3.11.** Si consideri il sistema lineare:

$$\begin{cases} 4x_1 + x_2 - x_3 = -64 \\ 3x_1 + 5x_2 + x_3 = 721 \\ x_1 - 4x_2 + 6x_3 = 807, \end{cases}$$

la cui soluzione è $x_1^* = 9$, $x_2^* = 99$, $x_3^* = 199$. Nella Tabella che segue sono riportati i valori ottenuti risolvendo tale sistema con il metodo di Jacobi, partendo dai valori iniziali $x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = 0$. Inoltre, per ciascuna iterazione, è indicato l'errore assoluto. Si osserva che se si utilizzasse come criterio di arresto la condizione (3.38) con Tol = 10^{-3} , il metodo si fermerebbe dopo 17 iterazioni e si otterrebbe un'approssimazione della soluzione corretta a 3 cifre decimali. Analogamente, con Tol = 10^{-4} si otterrebbe, dopo 21 iterazioni, un'accuratezza fino a 4 cifre decimali.

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$\ x^{(k)} - x^*\ $
⋮	⋮	⋮	⋮	⋮
10	9.05555	98.91611	199.04701	$.83887 \times 10^{-1}$
11	9.03272	98.95727	198.93482	$.65183 \times 10^{-1}$
12	8.99439	98.99340	198.96606	$.33941 \times 10^{-1}$
13	8.99316	99.01016	198.99654	$.10156 \times 10^{-1}$
14	8.99660	99.00479	199.00791	$.79100 \times 10^{-2}$
15	9.00078	99.00046	199.00376	$.37636 \times 10^{-2}$
16	9.00083	98.99878	199.00018	$.12201 \times 10^{-2}$
17	9.00035	98.99947	198.99905	$.95100 \times 10^{-3}$
18	8.99989	98.99998	198.99959	$.41216 \times 10^{-3}$
19	8.99990	99.00015	199.00000	$.14545 \times 10^{-3}$
20	8.99996	99.00006	199.00011	$.11333 \times 10^{-3}$
21	9.00001	99.00000	199.00004	$.44536 \times 10^{-4}$
22	9.00001	98.99998	199.00000	$.17207 \times 10^{-4}$
23	9.00000	98.99999	198.99999	$.13391 \times 10^{-4}$
⋮	⋮	⋮	⋮	⋮



Come mostra l'esempio 3.11, se nella relazione (3.38) si pone $\text{To1} = 10^{-m}$, si richiede che il metodo iterativo si arresti quando è stata ottenuta un'approssimazione della soluzione corretta a m cifre decimali. Tuttavia, l'utilizzo della (3.38) presuppone, come criterio di arresto, la conoscenza della soluzione del sistema, che in generale non è nota a priori. È necessario, allora, determinare stime calcolabili dell'errore di troncamento analitico ad ogni iterazione.

Teorema 3.2. *Dato il metodo iterativo:*

$$x_i^{(k+1)} = \sum_{j=1}^n c_{ij} x_j^{(k)} + d_i, \quad i = 1, \dots, n, \quad k \geq 0,$$

si assuma che sia consistente con il sistema lineare $Ax = b$. Se $\|C\| < 1$ si ha:

$$\|x^{(k+1)} - x^*\| \leq \frac{\|C\|}{1 - \|C\|} \|x^{(k+1)} - x^{(k)}\|, \quad k \geq 0. \quad (3.39)$$

Dimostrazione Poichè il metodo è consistente, se $\|C\| < 1$ esso converge alla soluzione del sistema, x^* . Valgono quindi le (3.32), con x_i^* al posto di \bar{x}_i , che possono essere riscritte nel modo seguente:

$$x_i^{(k+1)} - x_i^* = \sum_{j=1}^n c_{ij} (x_j^{(k+1)} - x_j^*) - \sum_{j=1}^n c_{ij} (x_j^{(k+1)} - x_j^{(k)}), \quad i = 1, \dots, n,$$

da cui

$$|x_i^{(k+1)} - x_i^*| \leq \sum_{j=1}^n |c_{ij}| |x_j^{(k+1)} - x_j^*| + \sum_{j=1}^n |c_{ij}| |x_j^{(k+1)} - x_j^{(k)}|, \quad i = 1, \dots, n,$$

e quindi:

$$\|x^{(k+1)} - x^*\| \leq \|C\| \|x^{(k+1)} - x^*\| + \|C\| \|x^{(k+1)} - x^{(k)}\|.$$

Poichè $\|C\| < 1$ per ipotesi, si ottiene:

$$\|x^{(k+1)} - x^*\| \leq \frac{\|C\|}{1 - \|C\|} \|x^{(k+1)} - x^{(k)}\|.$$



♣ **Esempio 3.12.** Relativamente alla risoluzione del sistema dell'esempio 3.11, nella Tabella che segue è riportato, per ciascuna iterazione, sia l'errore assoluto, sia la norma del massimo della differenza tra il vettore calcolato in quella iterazione ed il vettore ottenuto nell'iterazione precedente. Si osserva che tale valore costituisce una buona stima dell'errore assoluto.

k	$\ x^{(k)} - x^*\ $	$\ x^{(k)} - x^{(k-1)}\ $
\vdots	\vdots	\vdots
10	$.83887 \times 10^{-1}$	$.25544 \times 10^0$
11	$.65183 \times 10^{-1}$	$.11220 \times 10^0$
12	$.33941 \times 10^{-1}$	$.38338 \times 10^{-1}$
13	$.10156 \times 10^{-1}$	$.30478 \times 10^{-1}$
14	$.79100 \times 10^{-2}$	$.11373 \times 10^{-1}$
15	$.37636 \times 10^{-2}$	$.43332 \times 10^{-2}$
16	$.12201 \times 10^{-2}$	$.35861 \times 10^{-2}$
17	$.95100 \times 10^{-3}$	$.11285 \times 10^{-2}$
18	$.41216 \times 10^{-3}$	$.53885 \times 10^{-3}$
19	$.14545 \times 10^{-3}$	$.41671 \times 10^{-3}$
20	$.11333 \times 10^{-3}$	$.10878 \times 10^{-3}$
21	$.44536 \times 10^{-4}$	$.68793 \times 10^{-4}$
22	$.17207 \times 10^{-4}$	$.47863 \times 10^{-4}$
23	$.13391 \times 10^{-4}$	$.10962 \times 10^{-4}$
\vdots	\vdots	\vdots

♣

Il Teorema e l'esempio precedenti mostrano, quindi, che una stima dell'errore di troncamento analitico è data dalla differenza tra i vettori ottenuti in due iterazioni successive del metodo iterativo. Di conseguenza un possibile criterio di arresto "effettivamente utilizzabile", perchè basato su quantità calcolate, è il seguente:

$$\|x^{(k+1)} - x^{(k)}\| \leq \text{To1}. \quad (3.40)$$

Inoltre, in base al risultato stabilito dal Teorema 3.2, se si arresta il metodo iterativo quando è verificata la condizione (3.40), per l'errore assoluto di troncamento analitico si ha la maggiorazione:

$$\|x^{(k+1)} - x^*\| \leq \frac{\|C\|}{1 - \|C\|} \text{To1}. \quad (3.41)$$

Se, in particolare, si desidera stimare l'errore relativo di troncamento analitico:

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

basta osservare che, dalla (3.39), segue:

$$\frac{\|x^{(k+1)} - x^*\|}{\|x^*\|} \leq \frac{\|C\|}{1 - \|C\|} \frac{\|x^{(k+1)} - x^{(k)}\|}{\|x^*\|}.$$

Poichè la soluzione x^* non è nota, nella pratica si sostituisce ad essa il valore corrente $x^{(k+1)}$, ottenendo così il seguente criterio di arresto basato sulla distanza relativa tra i valori ottenuti in due iterazioni successive:

$$\frac{\|x^{(k+1)} - x^{(k)}\|}{\|x^{(k+1)}\|} \leq \text{To1}. \quad (3.42)$$

♣ **Esempio 3.13.** Si consideri ancora la risoluzione del sistema dell'esempio 3.11. Nella Tabella seguente sono riportati i valori dell'errore relativo e della distanza relativa tra i valori ottenuti in due iterazioni successive del metodo di Jacobi. Si osserva, inanzitutto, che l'errore relativo fornisce informazioni sull'accuratezza in termini di cifre significative. Inoltre la quantità $\|x^{(k)} - x^{(k-1)}\|/\|x^{(k)}\|$ costituisce una stima affidabile dell'errore relativo.

k	$\frac{\ x^{(k)} - x^*\ }{\ x^*\ }$	$\frac{\ x^{(k)} - x^{(k-1)}\ }{\ x^{(k)}\ }$
⋮	⋮	⋮
10	$.42154 \times 10^{-3}$	$.12833 \times 10^{-2}$
11	$.32755 \times 10^{-3}$	$.56398 \times 10^{-3}$
12	$.17056 \times 10^{-3}$	$.19269 \times 10^{-3}$
13	$.51035 \times 10^{-4}$	$.15316 \times 10^{-3}$
14	$.39749 \times 10^{-4}$	$.57151 \times 10^{-4}$
15	$.18912 \times 10^{-4}$	$.21774 \times 10^{-4}$
16	$.61311 \times 10^{-5}$	$.18021 \times 10^{-4}$
17	$.47789 \times 10^{-5}$	$.56707 \times 10^{-5}$
18	$.20711 \times 10^{-5}$	$.27078 \times 10^{-5}$
19	$.73090 \times 10^{-6}$	$.20940 \times 10^{-5}$
20	$.56949 \times 10^{-6}$	$.54663 \times 10^{-6}$
21	$.22380 \times 10^{-6}$	$.34569 \times 10^{-6}$
22	$.86467 \times 10^{-7}$	$.24052 \times 10^{-6}$
23	$.67290 \times 10^{-7}$	$.55086 \times 10^{-7}$
⋮	⋮	⋮

♣

Il costo computazionale relativo all'utilizzo dei criteri di arresto (3.40) e (3.42) è solo quello del calcolo delle quantità $\|x^{(k+1)} - x^{(k)}\|$ e $\|x^{(k+1)}\|$, in quanto i vettori che si considerano sono già calcolati durante il processo iterativo.

Nella prima versione degli algoritmi relativi ai metodi iterativi di Jacobi e G-S (**Procedure 3.1** e **3.2**) l'unica condizione per l'arresto del processo iterativo è il valore di **MaxIt**, che rappresenta il numero di iterazioni da eseguire. Una seconda versione di tali algoritmi, che include sia un criterio di arresto basato sulla relazione (3.42) sia una condizione di arresto basata su un numero massimo di iterazioni (**MaxIt**), è riportata nelle **Procedure 3.3** e **3.4**. Tali procedure forniscono, in output, anche una stima (**Err**) dell'errore relativo, cioè la quantità:

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

dove $x^{(k+1)}$ è l'ultimo valore calcolato prima dell'arresto del processo iterativo. Inoltre, altri due parametri di output delle procedure sono: la variabile `Iflag`, che segnala se è stata raggiunta l'accuratezza desiderata e la variabile `It`, che indica il numero di iterazioni che sono state eseguite.

```

procedure Jacobi2(in:  $n, A, b, xold, MaxIt, Tol$  ; out:  $x, Err, Iflag, It$ )

  /# SCOPO: risoluzione di un sistema di equazioni lineari
    con il metodo iterativo di Jacobi

  /# SPECIFICHE PARAMETRI:
  /# PARAMETRI DI INPUT:

  var:  $n$       : intero      {dimensione del sistema}
  var:  $A(n, n)$  : array di reali {coefficienti del sistema}
  var:  $b(n)$    : array di reali {termini noti del sistema}
  var:  $xold(n)$  : array di reali {valori iniziali}
  var:  $MaxIt$   : intero      {numero massimo di iterazioni}
  var:  $Tol$     : reale       {tolleranza usata nel criterio di arresto}

  /# PARAMETRI DI OUTPUT:

  var:  $x(n)$    : array di reali {soluzione calcolata}
  var:  $Err$     : reale       {stima dell'errore}
  var:  $Iflag$   : intero      {= 0 se la tolleranza non è soddisfatta}
                                     {= 1 se la tolleranza è soddisfatta}
  var:  $It$      : intero      {numero di iterazioni eseguite}

  /# VARIABILI LOCALI:

  var:  $i, j$     : interi      {contatori}
  var:  $xnorm$    : reale       {norma del vettore all'ultimo passo}
  var:  $dnorm$    : reale       {norma della diff. tra i vettori ottenuti}
                                     {in due iterazioni successive}

```

Procedura 3.3: Algoritmo di Jacobi (seconda versione) - continua

```

/# INIZIO ISTRUZIONI:
  It := 0;                                {inizializzazione di It}
  Iflag := 0;                             {inizializzazione di Iflag}
  repeat                                  {ciclo principale con}
    xnorm := 0.;                          {condizioni di arresto}
    dnorm := 0.;
    for i = 1, n do                        {ciclo per il calcolo di  $x^{(k)}$ }
      x(i) := b(i);
      for j = 1, n do                      {ciclo per il calcolo di  $x_i^{(k)}$ }
        if i  $\neq$  j then
          x(i) := x(i) - A(i, j) * xold(j);
        endif
      endfor
      x(i) := x(i)/A(i, i);
      if |x(i)| > xnorm then                {calcolo di  $\|x^{(k+1)}\|$ }
        xnorm := |x(i)|;
      endif
      if |x(i) - xold(i)| > dnorm then      {calcolo di  $\|x^{(k+1)} - x^{(k)}\|$ }
        dnorm := |x(i) - xold(i)|;
      endif
    endfor
    Err := dnorm/xnorm;                    {stima dell'errore}
  endrepeat

```

Procedura 3.3: Algoritmo di Jacobi (seconda versione) - continua

```
if  $Err \leq Tol$  then                                {controllo condizione di arresto}
     $Iflag := 1$ ;
endif
 $It := It + 1$ ;                                       {aggiornamento del contatore  $It$  }
for  $i = 1, n$  do                                     {aggiornamento di  $xold$ }
     $xold(i) := x(i)$ ;
endfor
until ( $It = MaxIt$  or  $Iflag = 1$ )                 {verifica condizioni di arresto}
return  $x, Err, Iflag, It$ 
end Jacobi2
```

Procedura 3.3: Algoritmo di Jacobi (seconda versione) - fine

```

procedure G-S2(in:  $n, A, b, xold, MaxIt, Tol$  ; out:  $x, Err, Iflag, It$ )
  /# SCOPO: risoluzione di un sistema di equazioni lineari
    con il metodo iterativo di Gauss-Seidel

  /# SPECIFICHE PARAMETRI:
  /# PARAMETRI DI INPUT:
    var:  $n$       : intero      {dimensione del sistema}
    var:  $A(n, n)$  : array di reali {coefficienti del sistema}
    var:  $b(n)$    : array di reali {termini noti del sistema}
    var:  $xold(n)$  : array di reali {valori iniziali}
    var:  $MaxIt$   : intero      {numero massimo di iterazioni}
    var:  $Tol$     : reale       {tolleranza usata nel criterio di arresto}

  /# PARAMETRI DI OUTPUT:
    var:  $x(n)$    : array di reali {soluzione calcolata}
    var:  $Err$     : reale       {stima dell'errore}
    var:  $Iflag$   : intero      {= 0 se la tolleranza non è soddisfatta}
                                     {= 1 se la tolleranza è soddisfatta}
    var:  $It$      : intero      {numero di iterazioni eseguite}

  /# VARIABILI LOCALI:
    var:  $i, j$     : interi      {contatori}
    var:  $xnorm$   : reale       {norma del vettore all'ultimo passo}
    var:  $dnorm$   : reale       {norma della diff. tra i vettori ottenuti}
                                     {in due iterazioni successive}

```

Procedura 3.4: Algoritmo di Gauss-Seidel (seconda versione) - continua

```

/# INIZIO ISTRUZIONI:
  It := 0;                                     {inizializzazione di It}
  Iflag := 0;                                 {inizializzazione di Iflag}
  for i = 1, n do                             {inizializzazione di x}
    x(i) := xold(i);
  endfor
  repeat                                     {ciclo principale con}
    xnorm := 0.;                             {condizioni di arresto}
    dnorm := 0.;
    for i = 1, n do                           {ciclo per il calcolo di x(k)}
      x(i) := b(i);
      for j = 1, n do                         {ciclo per il calcolo di xi(k)}
        if i ≠ j then
          x(i) := x(i) - A(i, j) * x(j);
        endif
      endfor
      x(i) := x(i)/A(i, i);
      if |x(i)| > xnorm then                  {calcolo di ||x(k+1)||}
        xnorm := |x(i)|;
      endif
      if |x(i) - xold(i)| > dnorm then       {calcolo di ||x(k+1) - x(k)||}
        dnorm := |x(i) - xold(i)|;
      endif
    endfor
    Err := dnorm/xnorm;                      {stima dell'errore}
    if Err ≤ Tol then                       {controllo condizione di arresto}
      Iflag := 1;
    endif

```

Procedura 3.4: Algoritmo di Gauss-Seidel (seconda versione) - continua

```

    It := It + 1;           {aggiornamento del contatore It }
  for i = 1, n do         {aggiornamento di xold}
    xold(i) := x(i);
  endfor
  until (It = MaxIt or Iflag = 1) {verifica condizioni di arresto}
  return x, Err, Iflag, It
end G-S2

```

Procedura 3.4: Algoritmo di Gauss-Seidel (seconda versione) - fine

3.4 Un esempio di software matematico per i metodi iterativi

A partire dalle procedure descritte nel precedente paragrafo, si possono costruire elementi di software matematico, per la risoluzione di sistemi di equazioni lineari mediante i metodi di Jacobi e Gauss-Seidel. Uno degli scopi di tale software è rendere trasparente, a chi le utilizza, i dettagli dell'algoritmo su cui si basano e di fornire la soluzione a partire da un insieme minimo di dati di input relativi al sistema da risolvere. Se, ad esempio, si utilizza il linguaggio di programmazione FORTRAN, le testate delle subroutine relative alle procedure Jacobi2 e G-S2 possono essere le seguenti:

```
SUBROUTINE JACOBI2(N,A,LDA,B,XOLD,MAXIT,TOL,X,ERR,IFLAG,IT)
```

```
  SUBROUTINE G-S2(N,A,LDA,B,XOLD,MAXIT,TOL,X,ERR,IFLAG,IT)
```

Di seguito è illustrato un esempio di programma chiamante la subroutine JACOBI2, per risolvere il sistema lineare:

$$\begin{cases} 3x_1 + 0.5x_2 + 0.3x_3 = 3.8 \\ 0.5x_1 + 2x_2 + 0.9x_3 = 3.4 \\ -0.1x_1 + 0.6x_2 + x_3 = 1.5, \end{cases}$$

la cui soluzione è $x^* = (1, 1, 1)$, con $\text{MAXIT}=50$, $\text{TOL}=0.000001$ e $x_i^{(0)} = 0.$, $i = 1, 2, 3$. In tale esempio si è supposto che i dati relativi alla matrice dei coefficienti ed al termine noto del sistema da risolvere, siano forniti da una subroutine (**SISTEMA**), che deve essere sviluppata dall'utente. Se si vuole risolvere un sistema lineare diverso è sufficiente modificare solo tale subroutine.

```

      PROGRAM JACOBI
C
C dichiarazione delle variabili
C
      REAL A(10,10), B(50), XOLD(10), X(10), TOL, ERR
      INTEGER N, LDA, MAXIT, CONV, K, IT, IFLAG, I
C
C inizializzazione delle variabili di input
C
      N=3
      LDA=10
      MAXIT=50
      TOL=0.000001
      DO 10 I=1,N
          XOLD(I)=0.0
10    CONTINUE
C
C chiamata di SISTEMA per generare A e b
C
      CALL SISTEMA(N,A,LDA,B)
C
C chiamata di JACOBI2
C
      CALL JACOBI2(N,A,LDA,B,XOLD,MAXIT,TOL,
* X,ERR,IFLAG,IT)
C
C stampa dei risultati
C
      IF (IFLAG .EQ. 1) THEN
          WRITE(*,*) Soluzione           =,(X(I),I=1,N)
          WRITE(*,*) Errore              =,ERR
          WRITE(*,*) Numero Iter.        =,IT
          WRITE(*,*) Iflag                =,IFLAG
      ELSE

```

Esempio di programma chiamante FORTRAN per la subroutine JACOBI2
- continua

```
        WRITE(*,*) Tolleranza non
* raggiunta
    ENDIF
    STOP
    END

C
C subroutine che genera la matrice
C ed il vettore dei termini noti
C
        SUBROUTINE SISTEMA(N,A,LDA,B)
C
C dichiarazione dei parametri
C
        REAL A(LDA,N), B(N)
        INTEGER N, LDA
C
C definizione di A e di B
C
        A(1,1)=3.
        A(1,2)=0.5
        A(1,3)=0.3
        A(2,1)=0.5
        A(2,2)=2.
        A(2,3)=0.9
        A(3,1)=-0.1
        A(3,2)=0.6
        A(3,3)=1.

        B(1)=3.8
        B(2)=3.4
        B(3)=1.5

        RETURN
    END
```

Esempio di programma chiamante FORTRAN per la subroutine JACOBI2
- fine

Sistema (a)	IT	x_1	x_2	x_3	ERR
Jacobi	48	5.199987	6.799983	3.399991	$0.98 \cdot 10^{-6}$
G-S	26	5.199993	6.799994	3.399997	$0.70 \cdot 10^{-6}$

Tabella 3.8: Schema riassuntivo relativo alla risoluzione del sistema (a) con i metodi di Jacobi e G-S.

L'esecuzione del programma che implementa la versione in singola precisione della subroutine JACOBI2, con l'esempio di programma chiamante appena illustrato, fornisce i risultati seguenti:

Soluzione = .9999998 .9999997 .9999995
 Errore = 9.53675E-07
 Numero iter. = 26
 Iflag = 1

♣ **Esempio 3.14.** Si risolvano i seguenti sistemi lineari:

$$(a) \begin{cases} 3x_1 - 2x_2 = 2 \\ -x_1 + 2x_2 - x_3 = 5 \\ -x_2 + 2x_3 = 0 \end{cases} \quad (b) \begin{cases} 4x_1 + 2x_2 + x_3 = 1 \\ x_1 + 3x_2 + x_3 = 0 \\ 2x_2 - 3x_3 = 1 \end{cases}$$

$$x_1^* = 5.2, x_2^* = 6.8, x_3^* = 3.4$$

$$x_1^* = 1/3, x_2^* = 0, x_3^* = -1/3$$

$$(c) \begin{cases} 4x_1 - 2x_2 = 0 \\ -2x_1 + 4x_2 - 2x_3 = 2 \\ -2x_2 + 4x_3 = 0 \end{cases}$$

$$x_1^* = 0.5, x_2^* = 1, x_3^* = 0.5$$

con i metodi di Jacobi e di Gauss-Seidel.

Modificando opportunamente la subroutine SISTEMA in base al sistema da risolvere, sono stati costruiti dei programmi FORTRAN che implementano le subroutine JACOBI2 e G-S2 e programmi chiamanti simili all'esempio precedente. In particolare si è utilizzato MAXIT=50, TOL= 10^{-6} e come valori iniziali $x_i^{(0)} = 0, i = 1, 2, 3$.

Nelle Tabelle 3.8, 3.9, 3.10 sono riportati i risultati ottenuti dall'esecuzione dei suddetti programmi per risolvere i sistemi in esame. Si osservi che IT indica il numero di iterazioni eseguite per soddisfare il criterio di arresto, x_1, x_2, x_3 sono gli ultimi valori calcolati prima dell'arresto del processo iterativo, ERR rappresenta la stima finale dell'errore relativo; la soluzione ottenuta è un'approssimazione della soluzione accurata a 7 cifre significative. Infine, dai valori di IT si evince che, per i sistemi in esame, il metodo di Gauss-Seidel converge alla soluzione più velocemente rispetto al metodo di Jacobi.



Sistema (b)	IT	x_1	x_2	x_3	ERR
Jacobi	16	0.333333	0.000002	-0.333333	$0.60 \cdot 10^{-6}$
G-S	4	0.333333	0.000000	-0.333333	$0.10 \cdot 10^{-6}$

Tabella 3.9: Schema riassuntivo relativo alla risoluzione del sistema (b) con i metodi di Jacobi e G-S.

Sistema (c)	IT	x_1	x_2	x_3	ERR
Jacobi	38	0.499999	0.999998	0.499999	$0.95 \cdot 10^{-6}$
G-S	20	0.499999	0.999999	0.499999	$0.95 \cdot 10^{-6}$

Tabella 3.10: Schema riassuntivo relativo alla risoluzione del sistema (c) con i metodi di Jacobi e G-S.

3.5 Efficienza

Nel paragrafo 3.1 si è analizzata la complessità di tempo di una iterazione dei metodi di Jacobi e di G-S e si è visto che il numero di flops è circa uguale al numero di coefficienti non nulli del sistema. Il risultato di tale analisi si estende evidentemente ad un generico metodo iterativo della forma:

$$x_i^{(k+1)} = \sum_{j=1}^n c_{ij} x_j^{(k)} + d_i, \quad i = 1, \dots, n, \quad k \geq 0, \quad (3.43)$$

per cui, indicato con $T_{it}(n)$ il relativo numero di flops, si ha:

$$T_{it}(n) \simeq n^2(1 - SP),$$

con SP grado di sparsità del sistema. Da ciò discende che la complessità di tempo totale è data da:

$$T(n) = k \cdot T_{it}(n) \simeq kn^2(1 - SP), \quad (3.44)$$

dove k è il numero di iterazioni che sono eseguite. Dalla (3.44) si evince, pertanto, che uno dei fattori da cui dipende l'efficienza di un metodo iterativo è il numero di iterazioni che sono eseguite e, quindi, la **velocità** con la quale la successione, generata dal metodo, converge alla soluzione del sistema.

Confrontando tale quantità con la complessità di tempo del metodo di eliminazione di Gauss, $T_{gauss} \simeq n^3/3$, si evince che risulta più efficiente utilizzare il metodo di Jacobi o il metodo di G-S, e cioè $T_{it}^{tot} < T_{gauss}$, se si effettua un numero di iterazioni k tale che

$$k < \frac{n}{3(1 - SP)}.$$

Ad esempio, se $SP = 0.99$, cioè se nel sistema, solo l'1% dei coefficienti è non nullo, si ha:

$$T_{it}^{tot}(n) \simeq k \cdot n^2 \cdot 0.01,$$

e, quindi, per un sistema lineare con tale grado di sparsità, è più efficiente utilizzare un metodo iterativo se si esegue un numero di iterazioni minore di circa $33n$.

3.5.1 Velocità di convergenza

Si assume che il metodo iterativo (3.43) sia consistente e che sia $\|C\| < 1$. In tali ipotesi, il metodo è convergente a x^* , dove x^* è la soluzione del sistema $Ax = b$.

Assunto, $e^{(k)} = x^{(k)} - x^*$, l'errore alla k -ma iterazione, il punto di partenza per ottenere una stima della velocità di convergenza del metodo iterativo è considerare la disuguaglianza:

$$\|e^{(k)}\| \leq \|C\|^k \|e^{(0)}\|, \tag{3.45}$$

che si ottiene dalla (3.35), dimostrata nel Teorema 3.1, con x^* al posto di \bar{x} . La (3.45) mette in relazione l'errore al passo k con l'errore iniziale e mostra che quanto più piccola è la quantità $\|C\|$ tanto più rapidamente converge a zero $\|e^{(k)}\|$ e quindi tanto più basso sarà il numero di iterazioni necessario per rendere la norma dell'errore minore di una prefissata tolleranza.

♣ **Esempio 3.15.** Se $\|C\| = \frac{1}{2}$ e $\|e^{(0)}\| = 1$, dalla (3.45) si ha $\|e^{(k)}\| \leq (\frac{1}{2})^k$; in tal caso, l'errore sarà minore di 10^{-7} dopo 24 iterazioni (infatti $k = 24$ è il più piccolo numero intero positivo per cui $(\frac{1}{2})^k \leq 10^{-7}$). D'altra parte, se $\|C\| = 0.8$, sono necessarie 73 iterazioni per avere $\|e^{(k)}\| \leq 10^{-7}$. ♣

Se le due quantità, $\|e^{(0)}\|$ e $\|C\|$, sono note (o possono essere facilmente calcolate), utilizzando la (3.45) è possibile ottenere una stima del minimo numero di iterazioni necessarie per rendere la norma dell'errore minore di una quantità $To1$ fissata. Infatti, si deve avere che:

$$\|C\|^k \|e^{(0)}\| \leq To1. \tag{3.46}$$

Applicando la funzione logaritmo naturale (\ln) ad entrambi i membri della disuguaglianza (3.46) si ha:

$$k \cdot \ln(\|C\|) + \ln(\|e^{(0)}\|) \leq \ln(To1). \tag{3.47}$$

Poichè $\|C\| < 1$ per ipotesi, si ha $\ln(\|C\|) < 0$ e, quindi, dalla (3.47) si ricava che una stima del numero minimo di iterazioni per ottenere la stabilita riduzione dell'errore è:

$$k \geq \frac{\ln(To1) - \ln(\|e^{(0)}\|)}{\ln(\|C\|)}. \tag{3.48}$$

Osservazione 3.2. *Esistono vari problemi legati all'utilizzo della stima (3.48). Il primo è che, non essendo nota in generale la soluzione del sistema, la quantità $\ln(\|e^{(0)}\|)$ non è calcolabile. Un altro problema è dovuto al fatto che la stima (3.48) dipende dalla norma che si utilizza, come mostra l'esempio 3.15. A conferma di ciò si consideri il sistema lineare:*

$$\begin{cases} x_1 + 0.4x_2 - 0.1x_3 = 1 \\ 0.3x_1 + x_2 - 0.001x_3 = -2.5 \\ 0.1x_1 + 0.4x_2 + x_3 = 3.0, \end{cases}$$

la cui soluzione è $x_1^* = 2.7315$, $x_2^* = -3.3154$, $x_3^* = 4.0530$. Si verifica facilmente che, applicando il metodo di Jacobi a tale sistema, partendo da $x_i^{(0)} = 0$, $i = 1, 2, 3$, si ha $\|e^{(0)}\|_\infty = 4.0530$ e $\|C\|_\infty = 0.5$, mentre $\|e^{(0)}\|_1 = 10.0999$ e $\|C\|_1 = 0.8$. Nel primo caso la (3.48) mostra che sono necessarie circa 25 iterazioni affinché l'errore sia minore di 10^{-7} , mentre nel secondo caso ne sono necessarie circa 82. Anche da tale esempio si evince, quindi, che il numero di iterazioni, che si stima essere necessario per rendere l'errore minore di To1 , dipende dalla norma che si utilizza per misurare l'ordine di grandezza della matrice C ¹³.

3.5.2 Un algoritmo per la memorizzazione dei coefficienti di una matrice ad elevato grado di sparsità

La risoluzione di un sistema lineare di grandi dimensioni comporta, in generale, un'elevata complessità di spazio, oltre che di tempo. Se il sistema è sparso è possibile utilizzare opportune tecniche per memorizzare in forma compatta solo i coefficienti diversi da zero; ciò può consentire una risoluzione più efficiente anche di sistemi lineari di grandi dimensioni.

Uno dei più noti schemi per la memorizzazione di una matrice sparsa è il cosiddetto **schema dei tre vettori**. La matrice $A \in \mathbb{R}^{n \times n}$ è rappresentata utilizzando tre vettori, R , C , J . Il vettore R contiene gli elementi della matrice diversi da zero, riga per riga. L'indice di colonna in A dell'elemento $R(k)$ è dato da $C(k)$, mentre $J(i)$, $i = 1, \dots, n$, fornisce la posizione, nell'array R , del primo elemento diverso da zero della i -ma riga.

♣ **Esempio 3.16.** Utilizzando lo schema dei tre vettori, la matrice

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

¹³Nei paragrafi successivi si vedrà che esiste un'altra stima della velocità di convergenza, che a differenza della (3.48) non dipende dalla norma utilizzata e quindi risulta ovviamente più affidabile.

è memorizzata nel modo seguente:

$$R = (7, -3, -1, 2, 8, 1, -3, 5, -1, 4, -2, 6);$$

$$C = (1, 3, 5, 1, 2, 3, 1, 4, 2, 5, 4, 6);$$

$$J = (1, 4, 6, 7, 9, 11, 13).$$

Si noti che l'ultimo elemento di J è uguale al numero totale degli elementi di R più uno. ♣

Nel caso in cui la matrice è simmetrica, è possibile utilizzare uno schema di memorizzazione più efficiente di quello appena illustrato, lo **schema del profilo**. Si fa uso di due array, R e M : il primo contiene, per ogni riga della matrice A , gli elementi compresi tra il primo non nullo su quella riga e l'elemento diagonale. Il secondo array, $M(i)$, $i = 1, \dots, n$, contiene la posizione, in R , dell'elemento diagonale della i -ma riga.

♣ **Esempio 3.17.** Secondo lo schema del profilo, la matrice

$$A = \begin{pmatrix} 25 & 3 & 0 & 0 & 0 \\ 3 & 21 & 2 & 4 & 0 \\ 0 & 2 & 23 & 0 & 0 \\ 0 & 4 & 0 & 22 & 1 \\ 0 & 0 & 0 & 1 & 20 \end{pmatrix}$$

è memorizzata mediante

$$R = (25, 3, 21, 2, 23, 4, 0, 22, 1, 20);$$

$$M = (1, 3, 5, 8, 10).$$

Utilizzando tale schema, si osserva che il generico elemento a_{ij} di A è memorizzato in R se $i - j < M(i) - M(i - 1)$. Inoltre $a_{ij} = R(p)$, con $p = M(i) - i + j$. ♣

Negli esempi precedenti, date le dimensioni della matrice A , non si ha un vantaggio evidente nell'utilizzo degli schemi descritti rispetto alle tecniche di memorizzazione usuali (**array bidimensionale**); tuttavia, il vantaggio è più sensibile per matrici di grandi dimensioni.

♣ **Esempio 3.18.** Si consideri una matrice quadrata A di dimensione $n = 100$ con $s = 400$ elementi diversi da zero. Se si usa per la sua memorizzazione l'usuale array bidimensionale, la complessità di spazio è $n \times n = 10000$, mentre se si sceglie lo schema dei tre vettori la complessità si riduce a $s + s + n + 1 = 901$, con un risparmio, quindi, di circa il 91%. ♣

Se k è il numero di iterazioni eseguite, la complessità di tempo totale è:

$$T_{it}^{tot}(n) \simeq k \cdot n^2(1 - SP).$$

Nella **Procedura 3.5** è illustrata una nuova versione dell'algoritmo relativo al metodo di Jacobi, in cui si assume che la matrice dei coefficienti del sistema sia memorizzata con lo schema dei tre vettori. Nella notazione utilizzata, i tre vettori R , C , J , sono rappresentati rispettivamente dagli array \mathbf{r} , \mathbf{c} , \mathbf{ia} .

```

procedure Jacobi3(in:  $n, s, r, c, ia, b, xold, MaxIt, Tol$  ; out:  $x, Err, Iflag, It$ )

  /* SCOPO: risoluzione di un sistema di equazioni lineari
     con il metodo iterativo di Jacobi;
     la matrice è memorizzata con lo schema dei tre vettori */

  /* SPECIFICHE PARAMETRI:
     /* PARAMETRI DI INPUT:
     var:  $n$       : intero      {dimensione del sistema}
     var:  $s$       : intero      {numero coefficienti non nulli}
     var:  $r(s)$    : array di reali {coefficienti non nulli riga per riga}
     var:  $c(s)$    : array di interi {indici di colonna degli elementi di  $r$ }
     var:  $ia(n)$   : array di interi { $ia(i) =$  posizione in  $r$  del primo}
                                     {elemento non nullo della  $i$ -ma riga}
     var:  $b(n)$    : array di reali {termini noti del sistema}
     var:  $xold(n)$  : array di reali {valori iniziali}
     var:  $MaxIt$   : intero      {numero massimo di iterazioni}
     var:  $Tol$     : reale       {tolleranza usata nel criterio di arresto}

     /* PARAMETRI DI OUTPUT:
     var:  $x(n)$    : array di reali {soluzione calcolata}
     var:  $Err$     : reale       {stima dell'errore}
     var:  $Iflag$   : intero      {= 0 se la tolleranza non è soddisfatta}
                                     {= 1 se la tolleranza è soddisfatta}
     var:  $It$     : intero      {numero di iterazioni eseguite}

     /* VARIABILI LOCALI:
     var:  $i, j, l$  : interi      {contatori}
     var:  $diff$    : intero      {numero degli elementi non nulli di}
                                     {ciascuna riga della matrice dei coefficienti}
     var:  $xnorm$   : reale       {norma del vettore calcolato all'ultimo passo}
     var:  $dnorm$   : reale       {norma della differenza tra i vettori ottenuti}
                                     {in due iterazioni successive}
     var:  $d$       : reale       {elemento diagonale, riga per riga}

```

Procedura 3.5: Algoritmo di Jacobi (terza versione) - continua

```

/# INIZIO ISTRUZIONI:
  It := 0;                                     {inizializzazione di It ed Iflag}
  Iflag := 0;
  repeat                                       {ciclo principale con}
    l := 0;                                   {condizioni di arresto}
    xnorm := 0.;
    dnorm := 0.;
    for i = 1, n do                            {ciclo per il calcolo di x(k)}
      x(i) := b(i);
      diff := ia(i + 1) - ia(i);
      for j = 1, diff do                       {ciclo per il calcolo di xi(k)}
        l := l + 1;
        if c(l) ≠ i then
          x(i) := x(i) - r(l) * xold(c(l));
        else
          d := r(l);
        endif
      endfor
      x(i) := x(i)/d;
      if |x(i)| > xnorm then                   {calcolo di ||x(k+1)||}
        xnorm := |x(i)|;
      endif
    endrepeat
  
```

Procedura 3.5: Algoritmo di Jacobi (terza versione) - continua

```

    if  $|x(i) - xold(i)| > dnorm$  then           {calcolo di  $\|x^{(k+1)} - x^{(k)}\|$ }
         $dnorm := |x(i) - xold(i)|$ ;
    endif
endfor
Err :=  $dnorm/xnorm$ ;                          {calcolo della stima dell'errore}
if  $Err \leq Tol$  then                          {controllo condizione di arresto}
    Iflag := 1;
endif
It := It + 1;                                  {aggiornamento del contatore It}
for  $i = 1, n$  do                                {aggiornamento di xold}
     $xold(i) := x(i)$ ;
endfor
until ( $It = MaxIt$  or  $Iflag = 1$ )             {verifica delle condizioni di arresto}
return  $x, Err, Iflag, It$ 
end Jacobi3

```

Procedura 3.5: Algoritmo di Jacobi (terza versione) - fine

3.6 Un esempio di programma MATLAB per i metodi iterativi

In questo paragrafo si descrive l'utilizzo del software MATLAB per lo sviluppo di un programma che implementi la **Procedura 3.3** relativa al metodo di Jacobi.

Allo stato attuale non sono presenti funzioni per la risoluzione di sistemi di equazioni lineari con metodi iterativi. Tuttavia, MATLAB fornisce la possibilità di utilizzare un insieme di istruzioni, come un linguaggio di programmazione, per lo sviluppo di funzioni che risolvono problemi specifici.

Un esempio di programma MATLAB per la risoluzione di sistemi lineari con il metodo di Jacobi è riportato nella **Funzione 1**. Tale funzione ha 4 parametri di output, cioè quelli racchiusi tra parentesi quadre dopo la parola chiave **function**. La funzione ha 5 parametri di input, quelli specificati nelle parentesi tonde dopo il nome della funzione. Il simbolo % si utilizza per inserire i commenti. La funzione **norm(x,inf)** calcola

la norma del massimo del vettore \mathbf{x} . La funzione **length(b)** restituisce, in output, la lunghezza del vettore \mathbf{b} . Segue un esempio di utilizzo della funzione in esame.

♣ **Esempio 3.19.** Si risolva il sistema lineare:

$$\begin{cases} 3x_1 + 0.5x_2 + 0.3x_3 = 3.8 \\ 0.5x_1 + 2x_2 + 0.9x_3 = 3.4 \\ -0.1x_1 + 0.6x_2 + x_3 = 1.5, \end{cases}$$

la cui soluzione è $x^* = (1, 1, 1)$, utilizzando la Funzione 1.

Le seguenti istruzioni definiscono le variabili che servono come parametri di input per la funzione.

```
>> A=[3 0.5 0.3; 0.5 2 0.9; -0.1 0.6 1];
>> b=[3.8; 3.4; 1.5];
>> x=[0; 0; 0];
>> Tol=0.000001;
>> MaxIt=50;
```

L'istruzione che segue chiama la funzione.

```
>> [x,ERR,IFLAG,IT] = Jacobi(A,b,x,Tol,MaxIt)
```

L'output è:

```
x =
    0.99999984417416
    0.99999968961009
    0.99999966015567

ERR =
    9.221026131790661e-07

IFLAG =
    1

IT =
    26
```

♣

```

function [x,ERR,IFLAG,IT] = Jacobi(A,b,x,Tol,MaxIt);
%
%
% -----
%
% SCOPO
% =====
% Calcola la soluzione di un sistema di equazioni lineari
% mediante il metodo iterativo di Jacobi.
%
%
% PARAMETRI DI INPUT
% =====
%
% A - Matrice dei coefficienti del sistema.
%
% b - Vettore dei termini noti.
%
% x - Valori iniziali.
%
% Tol - Tolleranza per il criterio di arresto.
%
% MaxIt - Numero massimo di iterazioni.
%
%
% PARAMETRI DI OUTPUT
% =====
%
% x - Soluzione.
%
% ERR - Stima dell'errore relativo.
%
% IFLAG - Vale 1 se si è raggiunta la tolleranza,
%         0 se si è raggiunto il numero massimo di iterazioni.
%
% IT - Numero di iterazioni eseguite.
%
%
% -----
%
%
%
```

Funzione 1:

Funzione MATLAB per la risoluzione di un sistema di equazioni lineari con il metodo di Jacobi - continua

```

n=length(b);           % lunghezza di b =
                        % = ordine del sistema
xold=x;               % vettore di appoggio
ERR=1.0;              % inizializzazione della stima dell'errore
IFLAG=0;              % inizializzazione di IFLAG
IT=0;                 % inizializzazione del numero di iterazioni

while ( ERR>Tol & IT<MaxIt) % k-ma iterazione
    for i=1:n
        sum=b(i)-(A(i, [1:i-1,i+1:n]))*...
            (xold([1:i-1,i+1:n]));
        x(i)=sum/A(i,i); % i-ma componente della
                        % k-ma approssimazione
    end
    ERR=norm(x-xold, inf)/norm(x,inf); % stima dell'errore relativo
    xold=x;
    IT=IT+1;
end

if (ERR ≤ Tol) % controllo sulla stima errore
    IFLAG=1;
end

```

Funzione 1:

Funzione MATLAB per la risoluzione di un sistema di equazioni lineari con il metodo di Jacobi - fine

3.7 I Metodi iterativi stazionari

3.7.1 Introduzione

In questo paragrafo, attraverso una riformulazione in termini matriciali, sono ripresi ed approfonditi i concetti di base discussi nel paragrafo precedente. Inoltre, sono illustrate alcune tecniche per accelerare la convergenza di metodi iterativi. In particolare l'attenzione è rivolta alla classe di metodi noti con il nome di **metodi iterativi stazionari**.

Si consideri il sistema di equazioni lineari:

$$Ax = b, \quad (3.49)$$

con $A = [a_{ij}] \in \mathbb{R}^{n \times n}$ **non singolare**, $x = [x_i]$ e $b = [b_i] \in \mathbb{R}^n$. Sia $x^* = A^{-1}b$ la sua soluzione.

Si possono dare le seguenti definizioni generali:

Definizione 3.5. (Metodo iterativo)

Una famiglia di funzioni $\{G_k\}$:

$$G_k : D_k \subseteq (\mathbb{R}^n)^{k+p} = \underbrace{\mathbb{R}^n \times \dots \times \mathbb{R}^n}_{k+p \text{ volte}} \rightarrow \mathbb{R}^n, \quad k = 0, 1, \dots$$

definisce un **processo iterativo (metodo iterativo)** con p **punti iniziali** e con dominio $D^* \subseteq D_0$, se D^* è non vuoto e se per ogni punto $(x^{(0)}, \dots, x^{(-p+1)}) \in D^*$, la successione $\{x^{(k)}\}$ generata mediante le:

$$x^{(k+1)} = G_k(x^{(k)}, \dots, x^{(-p+1)}), \quad k = 0, 1, \dots \tag{3.50}$$

è tale che $(x^{(k)}, \dots, x^{(-p+1)}) \in D_k, \forall k \geq 0$. Un punto x^* tale che $\lim_{k \rightarrow \infty} x^{(k)} = x^*$ è detto un **limite** del processo iterativo.

La seguente definizione caratterizza un processo iterativo del tipo (3.50).

Definizione 3.6. (Metodo iterativo stazionario)

Un metodo iterativo del tipo (3.50) a s passi è detto **stazionario** se $G_k = G$ e $D_k = D, \forall k \geq 0$. Quindi, in tal caso si ha:

$$x^{(k+1)} = G(x^{(k)}, \dots, x^{(k-s+1)}), \quad k = 0, 1, \dots$$

La funzione G è detta la **funzione di iterazione del metodo**.

I metodi di Jacobi e di Gauss-Seidel possono essere scritti nella forma:

$$x^{(k+1)} = Cx^{(k)} + d, \quad k = 0, 1, 2, \dots, \tag{3.51}$$

dove $C = [c_{ij}] \in \mathbb{R}^{n \times n}$ è detta **matrice di iterazione** e $d = [d_i] \in \mathbb{R}^n$. La (3.51) definisce la generica iterazione di un metodo iterativo **stazionario ad un passo**. Il termine *stazionario* si riferisce al fatto che nel metodo, a parte ovviamente la sequenza $\{x^{(k)}\}$, non ci sono quantità che variano al variare delle iterazioni (la matrice C ed il vettore d sono costanti). L'espressione *ad un passo* sta ad indicare che il vettore alla generica iterazione si ottiene utilizzando solo il vettore calcolato alla iterazione precedente.

♣ **Esempio 3.20.** Altri esempi di metodi iterativi sono i seguenti:

- **metodo non stazionario ad un passo**

$$x^{(k+1)} = C_{k+1}x^{(k)} + d^{(k+1)}, \quad k = 0, 1, \dots$$

- metodo stazionario a due passi

$$x^{(k+1)} = Cx^{(k)} + Bx^{(k-1)} + d, \quad k = 0, 1, \dots$$

- metodo non stazionario a due passi

$$x^{(k+1)} = C_{k+1}x^{(k)} + B_{k+1}x^{(k-1)} + d^{(k+1)}, \quad k = 0, 1, \dots$$

- metodo stazionario a s passi

$$x^{(k+1)} = \sum_{i=1}^s Cx^{(k+1-i)} + d, \quad k = 0, 1, \dots$$

♣

Definizione 3.7. (Metodo iterativo a s passi)

Un processo iterativo (3.50) è detto un **metodo a s passi**, se $p = s - k$ e se

$$G_k : D_k \subseteq (\mathbb{R}^n)^s \rightarrow \mathbb{R}^n, \quad k = 0, 1, \dots$$

e quindi

$$x^{(k+1)} = G_k(x^{(k)}, \dots, x^{(k-s+1)}), \quad k = 0, 1, \dots$$

I metodi iterativi concettualmente più semplici, ed in molti casi più utilizzati, sono quelli **stazionari ad un passo** ($p=1$), cioè quelli del tipo:

$$x^{(k+1)} = G(x^{(k)}), \quad k = 0, 1, \dots \quad (3.52)$$

Affinchè un metodo iterativo della forma (3.51) costituisca una procedura effettivamente utilizzabile per risolvere il sistema (3.49), è indispensabile che esso sia consistente.

Se la successione generata dal metodo iterativo (3.51) converge, indicato con \bar{x} il limite, risulta

$$\bar{x} = \lim_{k \rightarrow \infty} x^{(k+1)} = C \left(\lim_{k \rightarrow \infty} x^{(k)} \right) + d = C\bar{x} + d,$$

cioè il limite \bar{x} è soluzione del sistema lineare:

$$x = Cx + d, \quad \text{o equivalentemente} \quad (I - C)x = d, \quad (3.53)$$

il quale è detto il **sistema associato** al sistema (3.49). Allora, per i metodi iterativi del tipo (3.51) la definizione di consistenza può essere così riformulata:

Definizione 3.8. (Metodo consistente)

Un metodo iterativo del tipo (3.51) si dice **consistente** con il sistema di equazioni lineari (3.49), se la matrice $(I - C)$ è non singolare e, detta \bar{x} l'unica soluzione del sistema di equazioni lineari $(I - C)x = d$, si ha che \bar{x} è anche la soluzione del sistema $Ax = b$.

3.8 Metodi basati sulla decomposizione della matrice (splitting)

La tecnica più comune per sviluppare un metodo iterativo del tipo (3.51) che sia consistente è quella di considerare una decomposizione (**splitting**) della matrice A della forma:

$$A = M - R, \quad (3.54)$$

dove $M \in \mathbb{R}^{n \times n}$ è una matrice non singolare, detta **matrice di splitting**. Utilizzando tale decomposizione, il sistema lineare (3.49) è trasformato nel sistema:

$$Mx = Rx + b,$$

da cui deriva il metodo iterativo:

$$Mx^{(k+1)} = Rx^{(k)} + b, \quad k = 0, 1, 2, \dots$$

che, essendo M non singolare, può essere anche scritto come:

$$x^{(k+1)} = M^{-1}Rx^{(k)} + M^{-1}b, \quad k = 0, 1, 2, \dots \quad (3.55)$$

Il metodo (3.55) è un metodo del tipo (3.51), dove in particolare:

$$C = M^{-1}R = I - M^{-1}A, \quad d = M^{-1}b. \quad (3.56)$$

Inoltre si ha:

Teorema 3.3. *Posto $A = M - R$, con M matrice non singolare, il metodo iterativo:*

$$x^{(k+1)} = M^{-1}Rx^{(k)} + M^{-1}b, \quad k = 0, 1, 2, \dots$$

è consistente con il sistema $Ax = b$.

Dimostrazione Poichè A e M sono matrici non singolari, in base alla (3.56) anche la matrice $I - C = M^{-1}A$ è non singolare. Sia \bar{x} l'unica soluzione del sistema $x = Cx + d$. Si ha:

$$\bar{x} = C\bar{x} + d = (I - M^{-1}A)\bar{x} + M^{-1}b = \bar{x} - M^{-1}(A\bar{x} - b),$$

da cui discende che $A\bar{x} = b$ e, quindi, \bar{x} è la soluzione del sistema $Ax = b$. In base alla definizione 3.8, il metodo è quindi consistente. ■

È importante osservare che la scelta della matrice di splitting M in (3.54), per ragioni di efficienza, non può essere completamente arbitraria. Dalla (3.55) si osserva infatti che, ad ogni iterazione del metodo, è richiesto il calcolo di un vettore del tipo $y = M^{-1}z$. Di

conseguenza, è importante scegliere la matrice M in modo che la risoluzione di sistemi lineari della forma $My = z$ comporti un “ragionevole” costo computazionale. In generale, la scelta della matrice M cade su matrici per le quali il calcolo di $y = M^{-1}z$ richieda un costo computazionale equivalente a quello di un prodotto matrice-vettore (circa n^2 flops).

♣ **Esempio 3.21.** Si consideri la seguente decomposizione della matrice $A = [a_{ij}] \in \mathbb{R}^{n \times n}$:

$$A = D + L + U, \quad (3.57)$$

dove:

$$D = \begin{pmatrix} a_{11} & 0 & \cdots & \cdots & 0 \\ 0 & a_{22} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{n-1,n-1} & 0 \\ 0 & \cdots & \cdots & 0 & a_{nn} \end{pmatrix}, \quad a_{ii} \neq 0, \quad i = 1, \dots, n$$

$$L = \begin{pmatrix} 0 & \cdots & \cdots & \cdots & 0 \\ a_{21} & 0 & \cdots & \cdots & 0 \\ a_{31} & a_{32} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{n,n-1} & 0 \end{pmatrix}, \quad U = \begin{pmatrix} 0 & a_{12} & \cdots & a_{1,n-1} & a_{1n} \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & a_{n-2,n-1} & a_{n-2,n} \\ 0 & \cdots & \cdots & 0 & a_{n-1,n} \\ 0 & \cdots & \cdots & \cdots & 0 \end{pmatrix}.$$

Utilizzando tale decomposizione e denotando con il simbolo $[Px]_i$ la i -ma componente del prodotto tra una matrice $P \in \mathbb{R}^{n \times n}$ ed un vettore $x \in \mathbb{R}^n$, il metodo di Jacobi:

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

può essere riscritto nel modo seguente:

$$x_i^{(k+1)} = \frac{b_i}{a_{ii}} - \frac{1}{a_{ii}} [(L + U)x^{(k)}]_i, \quad i = 1, \dots, n,$$

da cui:

$$x_i^{(k+1)} = [D^{-1}b]_i - [D^{-1}(L + U)x^{(k)}]_i, \quad i = 1, \dots, n,$$

e, quindi, in forma matriciale:

$$x^{(k+1)} = -D^{-1}(L + U)x^{(k)} + D^{-1}b. \quad (3.58)$$

Dalla (3.58) si evince che il metodo iterativo di Jacobi deriva da uno splitting di A del tipo (3.54), dove in particolare:

$$M_J = D; \quad R_J = -(L + U).$$

Si osservi che la matrice di splitting del metodo di Jacobi è la matrice diagonale i cui elementi sono gli elementi diagonali della matrice A e quindi, poichè tali elementi sono diversi da zero per ipotesi, è non singolare. Di conseguenza, il metodo di Jacobi è consistente (Teorema 3.3).

In conclusione, in forma (3.51) il metodo di Jacobi assume la seguente espressione:

<p>METODO DI JACOBI</p> $x^{(k+1)} = C_J x^{(k)} + d_J$ $C_J = -D^{-1}(L + U) = I - D^{-1}A$ $d_J = D^{-1}b.$
--



♣ **Esempio 3.22.** Il metodo di G-S :

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

può essere riscritto come:

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

da cui, in base della decomposizione (3.57), si ha:

$$[D^{-1}(D + L)x^{(k+1)}]_i = [D^{-1}b]_i - [D^{-1}Ux^{(k)}]_i, \quad i = 1, \dots, n,$$

e quindi, in forma matriciale:

$$x^{(k+1)} = -(D + L)^{-1}Ux^{(k)} + (D + L)^{-1}b. \tag{3.59}$$

Dalla (3.59) si osserva che il metodo di G-S deriva da uno splitting del tipo (3.54), dove, in particolare:

$$M_{GS} = D + L; \quad R_{GS} = -U.$$

La matrice di splitting del metodo di G-S è quindi triangolare inferiore e, poichè gli elementi diagonali di A sono diversi da zero, è non singolare, da cui discende che il metodo di G-S è consistente.

In conclusione, in forma (3.51) esso assume la seguente espressione:

<p>METODO DI GAUSS-SEIDEL</p> $x^{(k+1)} = C_G x^{(k)} + d_G$ $C_G = -(D + L)^{-1}U = I - (D + L)^{-1}A$ $d_G = (D + L)^{-1}b$



3.9 Studio della convergenza

In questo paragrafo si studiano alcune condizioni sotto le quali la successione $\{x^{(k)}\}$ generata da un metodo iterativo del tipo (3.51) converge, qualunque sia la scelta di $x^{(0)}$.

In generale, per la risoluzione di sistemi di equazioni lineari:

$$Ax = b, \quad A \in \mathbb{R}^{n \times n}, \quad x, b \in \mathbb{R}^n, \quad (3.60)$$

si utilizzano processi iterativi lineari, cioè ad esempio metodi stazionari ad un passo del tipo (3.52) in cui la funzione di iterazione G è lineare, $G(x) = Cx + d$, dove C è una matrice $\in \mathbb{R}^{n \times n}$ e d è un vettore $\in \mathbb{R}^n$. Il metodo (3.52) assume quindi la forma:

$$x^{(k+1)} = Cx^{(k)} + d, \quad k = 0, 1, \dots \quad (3.61)$$

Da osservare che in tal caso il dominio della funzione G è tutto \mathbb{R}^n e, di conseguenza, il metodo (3.61) è sempre ben definito.

Considerati due punti x e $y \in \mathbb{R}^n$, si ha:

$$\|Cx + d - Cy - d\| = \|C(x - y)\| \leq \|C\| \|x - y\|.$$

Segue che la funzione $G(x) = Cx + d$ è una contrazione se $\|C\| < 1$. In tale ipotesi, in base al **Teorema 5.8** del Capitolo 5, **Parte 1**, il metodo iterativo (3.61) genera una successione $\{x^{(k)}\}$ che converge ad un punto fisso di G , e quindi, ad una soluzione del sistema lineare $(I - C)x = d$, qualunque sia il punto iniziale $x^{(0)}$. Da osservare, inoltre, che nell'ipotesi che $\|C\| < 1$, la matrice $(I - C)$ è non singolare, e quindi il sistema $(I - C)x = d$ ammette una sola soluzione. Se infine la matrice C è scelta in modo tale che il sistema $(I - C)x = d$ sia equivalente al sistema da risolvere (3.60), allora si può concludere che nell'ipotesi che $\|C\| < 1$ il metodo (3.61) converge alla soluzione di (3.60).

Una condizione necessaria e sufficiente per la convergenza è stabilita dal seguente:

Teorema 3.4. *Il metodo iterativo (3.51) è convergente se e solo se*

$$\rho(C) < 1 \quad (3.62)$$

dove

$$\rho(C) = \max_{i=1, \dots, n} |\lambda_i|,$$

con λ_i i -mo autovalore di C , è il **raggio spettrale** della matrice C .

Dimostrazione Sia $\rho(C) < 1$. In tale ipotesi la matrice $I - C$ è non singolare (vedi **Teorema B.5** dell'**Appendice B, Parte 1**). Sia \bar{x} l'unica soluzione del sistema lineare $x = Cx + d$ e sia

$$e^{(k)} = x^{(k)} - \bar{x}, \quad \forall k \geq 0.$$

Si ha:

$$e^{(k+1)} = x^{(k+1)} - \bar{x} = Cx^{(k)} + d - (C\bar{x} + d) = C(x^{(k)} - \bar{x}) = Ce^{(k)}.$$

Applicando lo stesso ragionamento a $e^{(k)}$ si ottiene:

$$e^{(k+1)} = C(Ce^{(k-1)}) = C^2e^{(k-1)}.$$

Dopo $k + 1$ passi si ha:

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

Poichè $\rho(C) < 1$, dal **Teorema B.17** dell'Appendice B, Parte 1, ($\lim_{k \rightarrow \infty} A^k v = 0, \forall v, \Leftrightarrow \rho(A) < 1$) segue che:

$$\lim_{k \rightarrow \infty} e^k = 0, \quad \forall e^{(0)},$$

e, quindi, il metodo (3.51) converge a \bar{x} qualunque sia $x^{(0)}$.

Si supponga ora che il metodo sia convergente. Indicato con \bar{x} il limite, si ricorda che \bar{x} è soluzione del sistema associato al metodo, cioè $\bar{x} = C\bar{x} + d$. Si supponga per assurdo che $\rho(C) \geq 1$. Sia $\tilde{\lambda}$ un autovalore di C tale che $|\tilde{\lambda}| = \rho(C)$ e sia \tilde{x} un autovettore ad esso corrispondente. Posto $x^{(0)} = \bar{x} - \tilde{x}$ e $e^{(k)} = x^{(k)} - \bar{x}$, si ha:

$$e^{(k)} = C^k e^{(0)} = \tilde{\lambda}^k e^{(0)},$$

da cui, poiché $|\tilde{\lambda}| \geq 1$, si ricava che la successione $\{e^{(k)}\}$ non converge a zero, e che quindi la successione $\{x^{(k)}\}$ non converge a \bar{x} . Ciò contraddice l'ipotesi. ■

♣ **Esempio 3.23.** Nell'esempio 3.3 si è considerato il sistema lineare:

$$\begin{cases} x_1 - 2x_2 + 2x_3 = 1 \\ -x_1 + x_2 - x_3 = -1 \\ -2x_1 - 2x_2 + x_3 = -3, \end{cases}$$

per il quale, come si evince dai risultati riportati nelle Tabelle 3.4 e 3.5, il metodo di Jacobi converge, mentre ciò non accade per il metodo di G-S. Si osservi che, per tale sistema, si ha:

$$\begin{aligned} \rho(C_J) &= 0.1020 \cdot 10^{-4} < 1 \\ \rho(C_G) &= 4.8284 > 1 \end{aligned}$$

che, in base al Teorema 3.4, conferma i risultati ottenuti. Quindi, si può affermare che, in generale, non esiste un legame tra il comportamento (in termini di convergenza) tra i due metodi.

Si consideri ora il sistema lineare:

$$\begin{cases} x_1 - 2x_2 - 2x_3 = 5 \\ -x_1 + x_2 - x_3 = -1 \\ -2x_1 - 2x_2 + x_3 = -3, \end{cases}$$

per il quale si ha:

$$\begin{aligned} \rho(C_J) &= 3.2361 > 1 \\ \rho(C_G) &= 12.3246 > 1 \end{aligned}$$

e quindi entrambi i metodi divergono. Si osservi che, considerando la decomposizione (3.57) per la matrice dei coefficienti di tale sistema, gli elementi della matrice $-(L+U)$ sono non negativi. Si consideri ora il seguente sistema lineare:

$$\begin{cases} 5x_1 - 2x_2 - 1x_3 = 3 \\ -x_1 + 4x_2 - x_3 = 2 \\ -3x_1 - 2x_2 + 7x_3 = 2, \end{cases}$$

per il quale si ha:

$$\rho(C_J) = 0.5944 < 1$$

$$\rho(C_G) = 0.3833 < 1$$

e quindi entrambi i metodi convergono. Si osservi che anche per tale sistema gli elementi della matrice $-(L+U)$ sono non negativi.

Il particolare comportamento (entrambi divergono o convergono) dei metodi di Jacobi e di G-S per gli ultimi due sistemi considerati è stabilito dal seguente:

Teorema 3.5. [Teorema di Stein-Rosenberg] *Posto $A = D + L + U$, se la matrice $-(L+U)$ è non negativa, allora è vera una e una soltanto delle condizioni seguenti*

- $\rho(C_J) = \rho(C_G) = 0$
- $\rho(C_J) = \rho(C_G) = 1$
- $0 < \rho(C_G) < \rho(C_J) < 1$
- $1 < \rho(C_J) < \rho(C_G)$

Quindi, in base a tale Teorema, per i sistemi lineari che ne soddisfano l'ipotesi, i metodi di Jacobi e di G-S o sono entrambi convergenti o sono entrambi divergenti. ♣

Per sintetizzare quanto detto finora, anche alla luce dei Teoremi 3.3 e 3.4, si può concludere che, se valgono le condizioni seguenti:

1. A è non singolare, e quindi il sistema lineare $Ax = b$ ha un'unica soluzione;
2. $C = I - M^{-1}A$, $d = M^{-1}b$, con M matrice di splitting non singolare;
3. $\rho(C) < 1$;

allora, qualunque sia $x^{(0)}$, la successione di vettori $\{x^{(k)}\}$ generata dal metodo (3.51) converge all'unica soluzione di $Ax = b$. Infatti, la condizione 2. assicura che il metodo iterativo (3.51) è consistente, cioè, se converge, converge alla soluzione del sistema $Ax = b$. La condizione 3., invece, assicura che il metodo converge qualunque sia il punto iniziale $x^{(0)}$.

Si osservi che la condizione per la convergenza (3.62) stabilita dal Teorema 3.4 non è di verifica facile perchè richiede il calcolo degli autovalori della matrice di iterazione C , un problema che in generale è computazionalmente molto dispendioso. D'altra parte, nelle applicazioni, è necessario disporre di condizioni facilmente verificabili. In tale ottica,

è importante ricordare la relazione seguente (vedi **Teorema B.12** dell'**Appendice B, Parte 1**):

$$\rho(A) \leq \|A\|, \quad (3.63)$$

che vale per ogni $A \in \mathbb{R}^{n \times n}$ e per ogni norma matriciale $\|\cdot\|$ compatibile con una norma vettoriale (cfr. **Definizione B.5** dell'**Appendice B, nella Parte 1**).

Si riporta inoltre un risultato, che è utile in alcune applicazioni, il quale stabilisce una condizione sufficiente per la convergenza dei metodi iterativi basati sullo splitting di A , nel caso in cui A presenti particolari caratteristiche.

Teorema 3.6. *Se la matrice A del sistema (3.49) è simmetrica e definita positiva¹⁴, posto $A = M - R$, con M matrice non singolare, il metodo iterativo (3.55) converge se la matrice simmetrica:*

$$Q = M + M^T - A,$$

è definita positiva.

Dimostrazione Siano B e G le due matrici simmetriche definite positive tali che $A = B^2$ e $Q = G^2$ (l'esistenza e l'unicità di tali matrici è assicurata dal **Teorema B.19** dell'**Appendice B, Parte 1**). Considerata la matrice di iterazione del metodo, $C = I - M^{-1}A$, si ponga:

$$H = BCB^{-1} = I - BM^{-1}B,$$

da cui, la matrice HH^T , i cui autovalori sono non negativi perchè semidefinita positiva (vedi **Teoremi B.9** e **B.10** dell'**Appendice B, Parte 1**), ha la seguente espressione:

$$\begin{aligned} HH^T &= (I - BM^{-1}B)(I - BM^{-T}B) = I - BM^{-T}B - BM^{-1}B + BM^{-1}AM^{-T}B = \\ &= I - BM^{-1}(M + M^T - A)M^{-T}B = I - T, \end{aligned} \quad (3.64)$$

dove:

$$T = BM^{-1}QM^{-T}B.$$

Poichè $Q = G^2$, la matrice T si scrive anche come:

$$T = (BM^{-1}G)(BM^{-1}G)^T.$$

Poichè B , G , e M sono matrici non singolari, la matrice $BM^{-1}G$ è non singolare. Di conseguenza, la matrice T , essendo il prodotto di una matrice non singolare per la sua trasposta, è simmetrica e definita positiva (**Teorema B.10** dell'**Appendice B, Parte 1**) e ha gli autovalori positivi. Dalla (3.64) segue che gli autovalori della matrice HH^T appartengono all'intervallo $[0, 1[$ e, quindi, $\rho(HH^T) < 1$. Se si considera la norma matriciale $\|\cdot\|_{2,B}$ (vedi **§B.3** dell'**Appendice B** della **Parte 1**), si ha:

$$\|C\|_{2,B} = \|BCB^{-1}\|_2 = \|H\|_2 = (\rho(HH^T))^{1/2} < 1.$$

Essendo, dunque, $\|C\| < 1$ ed, in particolare,

$$\rho(C) \leq \|C\| < 1,$$

per il Teorema 3.4 il metodo è convergente. ■

¹⁴Sistemi lineari con tale caratteristica sono spesso ricorrenti nelle applicazioni. Un esempio è la risoluzione numerica di equazioni differenziali alle derivate parziali di tipo ellittico.

3.10 Velocità di convergenza

Allo scopo di analizzare i fattori che ne determinano la velocità di convergenza, si suppone che il metodo iterativo (3.51) sia consistente e convergente. Posto:

$$e^{(k)} = x^{(k)} - x^*,$$

l'errore di troncamento analitico alla k -ma iterazione del metodo, si osserva (vedi dimostrazione del Teorema 3.4 con x^* al posto di \bar{x}) che vale la relazione:

$$e^{(k)} = C^k e^{(0)}.$$

Denotata con $\|\cdot\|$ una norma vettoriale ed una norma matriciale tra loro compatibili, si ha:

$$\|e^{(k)}\| \leq \|C^k\| \|e^{(0)}\|, \quad (3.65)$$

da cui discende che $\|C^k\|$ rappresenta una misura di quanto si riduce l'errore dopo k iterazioni.

In generale, si desidera arrestare il metodo iterativo quando:

$$\|e^{(k)}\| \leq \text{To1} \|e^{(0)}\|, \quad (3.66)$$

con $\text{To1} < 1$. In base alla (3.65), per soddisfare la (3.66), è sufficiente determinare un numero intero k tale che:

$$\|C^k\| \leq \text{To1}. \quad (3.67)$$

Assumendo che $\|C\| < 1$ (condizione sufficiente per la convergenza), si ha che $\|C^k\| \leq \|C\|^k < 1$, e, applicando la funzione logaritmo naturale ad entrambi i membri della (3.67) e moltiplicandoli per k , si ha:

$$k \geq -\ln(\text{To1}) / \left(-\frac{1}{k} \ln(\|C^k\|)\right).$$

Si evince quindi che il numero minimo di iterazioni, k , necessario per ottenere la richiesta riduzione (3.66) è inversamente proporzionale alla quantità:

$$-\frac{1}{k} \ln(\|C^k\|).$$

Ciò conduce alla seguente definizione:

Definizione 3.9. (Velocità media di convergenza)

Relativamente al metodo iterativo:

$$x^{(k+1)} = Cx^{(k)} + d, \quad k = 0, 1, 2, \dots,$$

si definisce **velocità media di convergenza** per k iterazioni la quantità

$$R_k(C) = -\frac{1}{k} \ln(\|C^k\|).$$

Si osservi che il valore di $R_k(C)$ dipende dalla norma matriciale che si utilizza.

Un'altra definizione di velocità di convergenza, che non presenta tale dipendenza, si ottiene dalla relazione:

$$\lim_{k \rightarrow \infty} \|C^k\|^{1/k} = \rho(C),$$

che vale se $\rho(C) < 1$. Si ha dunque la seguente:

Definizione 3.10. (Velocità asintotica di convergenza)

Relativamente al metodo iterativo:

$$x^{(k+1)} = Cx^{(k)} + d, \quad k = 0, 1, 2, \dots,$$

si definisce **velocità asintotica di convergenza**, o più semplicemente **velocità di convergenza**, il numero:

$$R_\infty(C) = \lim_{k \rightarrow \infty} R_k(C) = -\ln(\rho(C)).$$

Il valore di $R_\infty(C)$ è determinato solo dal raggio spettrale della matrice di iterazione e, quindi, è indipendente dalla norma che si utilizza. Inoltre, quanto più piccolo è $\rho(C)$ tanto più grande è $R_\infty(C)$, e quindi tanto più velocemente il metodo converge alla soluzione.

Sulla base della definizione 3.10, si ha che una stima del minimo numero di iterazioni necessarie per ridurre l'errore di un fattore **To1** è data da

$$k \geq \frac{-\ln(\text{To1})}{R_\infty(C)}. \quad (3.68)$$

♣ **Esempio 3.24.** Si considerino i sistemi lineari (a) e (b) dell'esempio 3.14. Per la matrice di iterazione del metodo di Jacobi applicato a tali sistemi si ha:

$$\text{sistema (a)} : \rho(C_J) = 0.7637;$$

$$\text{sistema (b)} : \rho(C_J) = 0.3333.$$

Quindi, il metodo di Jacobi converge più velocemente per il sistema (b). Ciò è confermato dai risultati riportati nelle Tabelle 3.8 e 3.9: il numero di iterazioni effettuate per rendere la norma del massimo dell'errore minore di $\text{To1} = 10^{-6}$ è 48 per il sistema (a) e 16 per il sistema (b).

Posto $\text{To1} = 10^{-6}$ nella (3.68), si ha:

$$\text{sistema (a)} : k \simeq 51;$$

$$\text{sistema (b)} : k \simeq 13,$$

da cui si evince che la (3.68) fornisce una stima attendibile "a priori" del numero di iterazioni necessarie per ottenere la fissata riduzione dell'errore. ♣

♣ **Esempio 3.25.** In relazione al sistema (a) dell'esempio 3.14, per la matrice di iterazione del metodo di G-S si ha $\rho(C_G) = 0.5833$. Quindi, utilizzando la (3.68), si ha che una stima del numero di iterazioni necessarie per ridurre l'errore di $\text{To1} = 10^{-6}$ è $k = 26$. L'attendibilità di tale stima è confermata dai risultati riportati nella Tabella 3.8. Inoltre, come descritto nell'esempio precedente, per la matrice di iterazione del metodo di Jacobi applicato al sistema (a) si ha $\rho(C_J) = 0.7637$ e, quindi, per tale sistema il metodo di G-S converge più velocemente, come confermano di nuovo i risultati riportati nella Tabella 3.8.

♣

Si osservi che, nonostante i risultati mostrati negli esempi precedenti, per alcuni sistemi lineari la stima (3.68) può fornire un valore che risulta sensibilmente inferiore al numero effettivo di iterazioni necessarie. L'utilizzo della seguente stima:

$$k \geq \frac{-\ln(\text{To1})}{R_k(C)},$$

in cui si considera la velocità media di convergenza, può fornire un valore più attendibile.

Osservazione 3.3. *Come illustrato nel §3.10, la velocità di convergenza di un metodo iterativo dipende dalle proprietà spettrali della matrice dei coefficienti; di qui il tentativo di trasformare il sistema lineare in esame in uno equivalente, che abbia, cioè, la stessa soluzione, ma presenti proprietà spettrali più vantaggiose ai fini della convergenza del metodo. La matrice che determina tale trasformazione prende il nome di **precondizionatore**. Ad esempio, se una matrice M approssima, in qualche modo, la matrice dei coefficienti, A , il sistema trasformato*

$$M^{-1}Ax = M^{-1}b$$

ha la stessa soluzione del sistema originale, $Ax = b$, ma le proprietà spettrali della sua matrice dei coefficienti, $M^{-1}A$, potrebbero garantire una più rapida convergenza del metodo iterativo. In generale, comunque, un buon preconditionatore deve migliorare la velocità di convergenza del metodo, ma tanto da bilanciare il costo computazionale richiesto per la sua costruzione e la sua applicazione.

3.11 Accelerazione della convergenza

3.11.1 Metodi di rilassamento: il metodo SOR

L'analisi effettuata nel paragrafo precedente ha mostrato che la convergenza di un metodo iterativo del tipo (3.51) è tanto più rapida quanto più piccolo è il raggio spettrale

della matrice di iterazione del metodo. Di conseguenza, una delle tecniche più naturali per accelerare la convergenza è apportare modifiche al metodo in modo da rendere il più piccolo possibile $\rho(C)$.

Come primo passo in tale direzione, si riscrive il metodo di G-S:

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

nella forma (aggiungendo e sottraendo $a_{ii}x_i^{(k)}$):

$$x_i^{(k+1)} = x_i^{(k)} + r_i^{(k)}, \quad i = 1, \dots, n, \quad (3.69)$$

dove:

$$r_i^{(k)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij} x_j^{(k)}}{a_{ii}}, \quad i = 1, \dots, n.$$

Considerato un numero reale ω , si modificano le (3.69) come segue:

$$x_i^{(k+1)} = x_i^{(k)} + \omega r_i^{(k)}, \quad i = 1, \dots, n. \quad (3.70)$$

Le formule (3.70) definiscono il **metodo SOR** (Successive Over Relaxation) ed ω è detto il **parametro di rilassamento**.

Le (3.70) possono essere scritte come:

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

da cui, considerando la decomposizione (3.57) introdotta nell'esempio 3.23, si ha:

$$[(I + \omega D^{-1}L)x^{(k+1)}]_i = [(1 - \omega)x^{(k)}]_i - [\omega D^{-1}Ux^{(k)}]_i + \omega [D^{-1}b]_i, \quad i = 1, \dots, n,$$

e, quindi, in forma matriciale:

$$x^{(k+1)} = (I + \omega D^{-1}L)^{-1}[(1 - \omega)I - \omega D^{-1}U]x^{(k)} + (I + \omega D^{-1}L)^{-1}\omega D^{-1}b. \quad (3.71)$$

Osservando che:

$$(I + \omega D^{-1}L) = \omega D^{-1}(\omega^{-1}D + L)$$

e

$$(1 - \omega)I - \omega D^{-1}U = \omega D^{-1}[(1 - \omega)\omega^{-1}D - U],$$

la (3.71) diventa:

$$x^{(k+1)} = (\omega^{-1}D + L)^{-1}[(\omega^{-1} - 1)D - U]x^{(k)} + (\omega^{-1}D + L)^{-1}b. \quad (3.72)$$

Dalla (3.72) si osserva che il metodo SOR deriva da uno splitting di A del tipo (3.54), dove, in particolare:

$$M_\omega = \omega^{-1}D + L; \quad R_\omega = (\omega^{-1} - 1)D - U.$$

È immediato verificare che, se $\omega \neq 0$, la matrice di splitting M_ω è non singolare e quindi il metodo SOR è consistente. Inoltre, nel caso $\omega = 1$, il metodo SOR si riduce al metodo di G-S. In conclusione, in forma (3.51) il metodo SOR ha la seguente espressione:

METODO SOR
$x^{(k+1)} = C_\omega x^{(k)} + d_\omega$
$C_\omega = (\omega^{-1}D + L)^{-1}[(\omega^{-1} - 1)D - U] = (\omega^{-1}D + L)^{-1}[\omega^{-1}D - D - U] =$
$= (\omega^{-1}D + L)^{-1}[(\omega^{-1}D + L) - A] = I - (\omega^{-1}D + L)^{-1}A$
$d_\omega = (\omega^{-1}D + L)^{-1}b$

(3.73)

Per la convergenza del metodo SOR è fondamentale il risultato seguente:

Teorema 3.7. *Per la matrice di iterazione del metodo SOR si ha:*

$$\rho(C_\omega) \geq |\omega - 1|.$$

Dimostrazione Considerate le matrici non singolari $E = D^{-1}L$ e $F = D^{-1}U$, la matrice di iterazione del metodo SOR può scriversi come:

$$C_\omega = (I + \omega E)^{-1}[(1 - \omega)I - \omega F].$$

Poichè $(I + \omega E)$ è una matrice triangolare inferiore con elementi diagonali uguali a 1, lo è anche la sua inversa che ha, quindi, determinante uguale a 1. La matrice $[(1 - \omega)I - \omega F]$ è una matrice triangolare superiore con elementi diagonali tutti uguali a $(1 - \omega)$, il cui determinante è quindi $(1 - \omega)^n$. Da quanto detto, il determinante di C_ω , $\det(C_\omega)$, è:

$$\det(C_\omega) = \det((I + \omega E)^{-1}) \cdot \det([(1 - \omega)I - \omega F]) = (1 - \omega)^n.$$

Poichè il prodotto degli autovalori di una matrice è uguale al suo determinante, si ha:

$$\rho(C_\omega) \geq (|\omega - 1|^n)^{1/n} = |\omega - 1|$$

e quindi la tesi. ■

Dal risultato del Teorema precedente, si evince che la condizione $0 < \omega < 2$ è necessaria per la convergenza del metodo SOR, ed è quindi di scarsa utilità nelle applicazioni.

Tuttavia, se la matrice A del sistema lineare da risolvere è simmetrica definita positiva si ha il seguente importante risultato, che è una conseguenza dei Teoremi 3.6 e 3.7:

Corollario 3.2. *Se A è simmetrica e definita positiva, il metodo SOR è convergente se e solo se*

$$0 < \omega < 2.$$

Dimostrazione Che la condizione sia necessaria discende dal Teorema 3.7. Si supponga ora che $0 < \omega < 2$. Ricordando che la matrice di splitting del metodo SOR è $M_\omega = \omega^{-1}D + L$, si ha, essendo A simmetrica, per cui è $L^T = U$, che la matrice $Q = M_\omega + M_\omega^T - A = (2\omega^{-1} - 1)D$ è una matrice diagonale, i cui elementi sono dati da $(2\omega^{-1} - 1)a_{ii}$, $i = 1, \dots, n$, dove a_{ii} , $i = 1, \dots, n$, sono gli elementi diagonali di A . Poichè A è definita positiva, i suoi elementi diagonali sono tutti positivi; di conseguenza, anche la matrice Q , poichè $0 < \omega < 2$, ha tutti gli elementi positivi e, quindi, è definita positiva. In base al Teorema 3.6, si ha che il metodo SOR converge. ■

Il problema cruciale relativo all'utilizzo del metodo SOR è la scelta del parametro di rilassamento ω ; tale scelta determina la velocità di convergenza del metodo e, quindi, la sua maggiore efficacia rispetto ai metodi di Jacobi e di G-S. Si consideri la seguente:

Definizione 3.11. (Valore ottimale)

*Relativamente al metodo SOR, si definisce **valore ottimale di ω** il valore ω_{opt} tale che:*

$$\rho(C_{\omega_{opt}}) = \min_{\omega} \rho(C_{\omega}).$$

In generale, la determinazione del valore ottimale del parametro di rilassamento del metodo SOR è un problema di non facile soluzione. Tuttavia, per alcune importanti classi di sistemi lineari il valore di ω_{opt} è noto o può essere stimato con sufficiente accuratezza. Nel seguito si analizza, in breve, il caso di sistemi lineari in cui la matrice dei coefficienti è **tridiagonale a blocchi**.

3.11.2 Matrici tridiagonali a blocchi

Definizione 3.12. (Matrice tridiagonale a blocchi)

*Si definisce **matrice tridiagonale a blocchi** (nel seguito chiamata **T-matrice**) una matrice $A \in \mathbb{R}^{n \times n}$ del tipo:*

$$A = \begin{pmatrix} D_1 & F_1 & & & & \\ E_2 & D_2 & F_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & E_{p-1} & D_{p-1} & F_{p-1} & \\ & & & E_p & D_p & \end{pmatrix},$$

dove D_i, E_i, F_i sono matrici $n_i \times n_i$, con $\sum_{i=1}^p n_i = n$.

Per le T -matrici valgono i seguenti risultati.

Teorema 3.8. *Dato un sistema lineare $Ax = b$, in cui A è una T -matrice, con matrici diagonali D_i non singolari, si ha:*

$$\rho(C_{GS}) = \rho^2(C_J),$$

e quindi:

$$R_\infty(C_{GS}) = 2R_\infty(C_J).$$

Sulla base del Teorema precedente, si ha che, per sistemi lineari in cui la matrice dei coefficienti è una T -matrice, se il metodo di Jacobi converge, allora converge anche il metodo di G-S con una velocità asintotica doppia.

Teorema 3.9. *Dato un sistema lineare $Ax = b$, in cui A è una T -matrice, con matrici diagonali D_i non singolari, se tutti gli autovalori della matrice di iterazione del metodo di Jacobi, C_J , sono reali e se $\rho(C_J) < 1$, allora il metodo SOR converge per $0 < \omega < 2$. Inoltre si ha:*

- 1) $\omega_{opt} = \frac{2}{1+(1-\rho^2(C_J))^{1/2}}$
- 2) $\rho(C_{\omega_{opt}}) = \omega_{opt} - 1$.

L'interesse pratico dei risultati precedenti è tuttavia legato alla possibilità di disporre di una stima accurata del raggio spettrale della matrice di iterazione del metodo di Jacobi. In alcuni casi tale stima è nota, come mostra l'esempio che segue.

♣ **Esempio 3.26.** Si consideri il seguente problema al contorno per l'equazione di Poisson:

$$\begin{cases} -u_{xx} - u_{yy} = f(x, y) & (x, y) \in \Omega =]0, 1[\times]0, 1[\\ u(x, y) = g(x, y) & (x, y) \in \delta\Omega \end{cases} \quad (3.74)$$

dove $\delta\Omega$ è la frontiera del quadrato unitario, $f(x, y)$ e $g(x, y)$ sono funzioni continue rispettivamente su Ω e $\delta\Omega$. La soluzione numerica di tale problema la si può ottenere mediante il *metodo delle differenze finite* utilizzando uno schema a 5 punti. Più precisamente si considera una reticolazione del dominio quadrato $\Omega \cup \delta\Omega$ in quadratini di lato $h = 1/(m+1)$ e si vuole calcolare un'approssimazione dei valori che la funzione incognita $u(x, y)$ assume nei punti interni di tale reticolazione (quelli sui punti di frontiera sono noti, nota la g , per la condizione imposta nella (3.74)), cioè nei punti:

$$(x_i, y_j), \quad x_i = i \cdot h, \quad y_j = j \cdot h, \quad i, j = 1, \dots, m.$$

A tal fine, nei suddetti punti si approssimano le derivate parziali seconde con le usuali differenze centrali in modo che il problema di partenza (3.74) sia approssimato dal problema discreto:

$$\frac{1}{h^2}(4u_{i,j} - u_{i-1,j} - u_{i+1,j} - u_{i,j+1} - u_{i,j-1}) = f_{i,j}, \quad i, j = 1, \dots, m, \quad (3.75)$$

dove $u_{i,j}$ e $f_{i,j}$ rappresentano i valori approssimati rispettivamente di $u(x_i, y_j)$ e $f(x_i, y_j)$. Numerando i punti interni della reticolazione (a partire da 1) da sinistra a destra e dal basso verso l'alto e portando a termine noto i valori sui punti della frontiera, posto

$$\mathbf{u}^T = (u_{11}, u_{12}, \dots, u_{1m}, u_{21}, u_{22}, \dots, u_{2m}, \dots, u_{m1}, u_{m2}, \dots, u_{mm}),$$

le (3.75) costituiscono un sistema di n equazioni lineari in n incognite, con $n = m^2$:

$$Au = f, \quad (3.76)$$

la cui soluzione rappresenta i valori approssimati della funzione incognita $u(x, y)$ nei punti interni della reticolazione. La matrice dei coefficienti ha la seguente struttura:

$$A = \begin{pmatrix} T & -I & & & \\ -I & T & -I & & \\ & \ddots & \ddots & \ddots & \\ & & & -I & T & -I \\ & & & & -I & T \end{pmatrix},$$

dove I è la matrice identità di ordine m e T è la matrice tridiagonale di ordine m :

$$T = \begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & & -1 & 4 & -1 \\ & & & & -1 & 4 \end{pmatrix}.$$

La matrice A è quindi una T -matrice simmetrica, in cui le matrici diagonali, T , sono non singolari. Inoltre, posto:

$$t = \frac{\pi}{m+1} = \pi h, \quad (3.77)$$

è noto che gli autovalori della matrice A sono dati da:

$$\lambda_{i,j} = 4 \left(\sin^2 i \frac{t}{2} + \sin^2 j \frac{t}{2} \right), \quad i, j = 1, \dots, m.$$

Dalla conoscenza degli autovalori di A si determinano facilmente anche quelli della matrice di iterazione del metodo di Jacobi applicato al sistema (3.76), che nel caso specifico è data da $C_J = I - D^{-1}A = I - \frac{1}{4}A$. Gli autovalori di C_J sono quindi:

$$\mu_{i,j} = 1 - \frac{1}{4}\lambda_{i,j} = \frac{1}{2}(\cos it + \cos jt), \quad i, j = 1, \dots, m,$$

da cui, per il raggio spettrale di C_J si ha:

$$\rho(C_J) = \max_{i,j} |\mu_{i,j}| = \cos t < 1. \quad (3.78)$$

In conclusione, per il sistema (3.76) sono verificate tutte le ipotesi del Teorema 3.9, per cui il valore ottimale del parametro di rilassamento relativo al metodo SOR è:

$$\omega_{opt} = \frac{2}{1 + (1 - \cos^2 t)^{1/2}} = \frac{2}{1 + \sin t}. \quad (3.79)$$

Metodo	IT	x_1	x_2	x_3	x_4	ERR
Jacobi	20	0.666674	0.999999	0.999999	1.33332	$0.71 \cdot 10^{-6}$
G-S	12	0.666674	0.999999	0.999999	1.33332	$0.49 \cdot 10^{-6}$
SOR	8	0.666674	1.000000	1.000000	1.33333	$0.76 \cdot 10^{-6}$

Tabella 3.11: Schema riassuntivo relativo alla risoluzione del sistema (3.76), con A e f dati dalle (3.81), con i metodi di Jacobi, G-S e SOR ($\omega = 1.0718$).

Come esempio di applicazione del risultato ottenuto, si consideri, in particolare, il caso in cui, in (3.74), si abbia:

$$f(x, y) = 0; \quad g(x, y) = x + y. \quad (3.80)$$

Con facili calcoli si verifica che in tali ipotesi e per $m = 2$ la matrice A ed il vettore f del sistema (3.76) sono:

$$A = \begin{pmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{pmatrix}; \quad f = \begin{pmatrix} 2/3 \\ 2 \\ 2 \\ 10/3 \end{pmatrix}. \quad (3.81)$$

La soluzione di tale sistema è $(2/3, 1, 1, 4/3)$. Inoltre, dalla (3.79) con $t = \frac{\pi}{3}$ si ha che:

$$\omega_{opt} = 1.0718$$

Nella Tabella 3.11 sono riportati i risultati ottenuti risolvendo il sistema (3.76), con A e f dati dalle (3.81), rispettivamente con il metodo di Jacobi, il metodo di G-S ($\omega = 1$) e il metodo SOR con $\omega = \omega_{opt} = 1.0718$. In particolare sono state utilizzate le **Procedure 3.3** e **3.4**, l'ultima opportunamente modificata per introdurre il parametro di rilassamento. Inoltre, si è posto $\text{To1} = 10^{-6}$ e $x_i^{(0)} = 0$, $i = 1, \dots, 4$. Si osservi che, come si aspettava, la velocità di convergenza del metodo di G-S è quasi il doppio di quella del metodo di Jacobi. Inoltre, il numero di iterazioni (IT) richieste dal metodo SOR per soddisfare la tolleranza è inferiore rispetto a quelle richieste dal metodo di G-S.

È da osservare che il miglioramento in velocità di convergenza, usando $\omega = \omega_{opt}$ nel metodo SOR, aumenta al crescere di m e, quindi, della dimensione del sistema. Infatti, per m abbastanza grande, e quindi, dalla (3.77), per $t \ll 1$, si ha che una stima accurata del raggio spettrale della matrice di iterazione del metodo SOR è data da:

$$\rho(C_{\omega_{opt}}) = \omega_{opt} - 1 = \frac{2}{1 + \sin t} - 1 \simeq 1 - 2t.$$

Di conseguenza, una stima della velocità di convergenza del metodo SOR con $\omega = \omega_{opt}$ è:

$$R_{\omega_{opt}} \simeq -\ln(1 - 2t) \simeq 2t. \quad (3.82)$$

Per il metodo di G-S si ha:

$$\rho(C_G) = \rho^2(C_J) = \cos^2 t \simeq 1 - t^2,$$

da cui la velocità di convergenza del metodo di G-S è:

$$R_G \simeq -\ln(1 - t^2) \simeq t^2.$$

m	n	IT (Jacobi)	IT (G-S)	IT (SOR)
2	4	20	12	8
3	9	38	21	12
5	25	84	45	18
7	49	142	77	24
9	81	214	116	30
10	100	254	138	32

Tabella 3.12: Risultati relativi alla risoluzione del sistema (3.76) al variare di m , utilizzando i metodi di Jacobi, G-S e SOR con parametro ottimale.

Confrontando le velocità di convergenza dei due metodi, si ha che, per m abbastanza grande, il metodo SOR con valore ottimale del parametro di rilassamento richiede, rispetto al metodo di G-S, un numero di iterazioni minore di un fattore dato da:

$$\frac{R_{\omega_{opt}}}{R_G} \simeq \frac{2}{t} \simeq \frac{2m}{\pi}.$$

Tale fattore è proporzionale a m , cioè aumenta al crescere di m . Ad esempio, se $m = 5$ si ha:

$$\frac{R_{\omega_{opt}}}{R_G} \simeq 3,$$

mentre se $m = 10$ si ha:

$$\frac{R_{\omega_{opt}}}{R_G} \simeq 6.$$

In pratica, ciò significa che, se $m = 5$, il metodo SOR con parametro ottimale è circa 3 volte più veloce del metodo di G-S, mentre, se $m = 10$, circa 6. In Tabella 3.12 è riportato, al variare di m da 2 a 10, il numero di iterazioni richiesto dai metodi di Jacobi, G-S e SOR con parametro ottimale, per risolvere il sistema (3.76) derivante dal problema differenziale (3.74), con le funzioni f e g date dalle (3.80) ed utilizzando $\text{To1}=10^{-6}$. I risultati ottenuti confermano le stime teoriche sul guadagno, in termini di velocità di convergenza, che si ha utilizzando il metodo SOR con il valore ottimale del parametro di rilassamento.



È importante osservare ancora che l'utilizzazione pratica dell'espressione analitica del valore ottimale del parametro di rilassamento fornita dal Teorema 3.9, dipende, tuttavia, dalla possibilità di disporre di una stima del raggio spettrale della matrice di iterazione del metodo di Jacobi. Tale stima, inoltre, a parte qualche caso particolare come quello illustrato nell'esempio 3.26, non è nota, e spesso calcolarla è computazionalmente molto dispendioso¹⁵.

¹⁵Per tale motivo, la maggior parte delle principali librerie di software matematico, che contengono routine per la risoluzione di sistemi lineari con il metodo SOR, prevedono procedure *adattative* per il calcolo di ω_{opt} , cioè algoritmi numerici che determinino dinamicamente, ad ogni iterazione, una stima di ω_{opt} .

3.11.3 Il metodo SSOR

Si conclude il paragrafo con una breve descrizione di un altro metodo di rilassamento che si ottiene “simmetrizzando” il metodo SOR. Più precisamente, l’idea di base è quella di considerare il procedimento iterativo:

$$x_i^{(k+1,1)} = x_i^{(k)} + \omega r_i^{(k+1,1)}, \quad i = 1, \dots, n, \quad (3.83)$$

$$x_i^{(k+1)} = x_i^{(k+1,1)} + \omega r_i^{(k+1)}, \quad i = n, \dots, 1, \quad (3.84)$$

dove:

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

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

In tale procedimento iterativo, noto con il nome di **metodo Symmetric SOR (SSOR)**, si determinano i valori $x_i^{(k+1,1)}$ mediante il metodo SOR classico e successivamente si calcolano i valori $x_i^{(k+1)}$ utilizzando lo schema SOR all’indietro, cioè considerando le righe di A a partire dall’ultima.

Utilizzando la decomposizione (3.57) di A è possibile, con un procedimento analogo a quello illustrato per il metodo SOR, esprimere le (3.83) e (3.84) nella seguente forma matriciale:

$$x^{(k+1,1)} = (\omega^{-1}D + L)^{-1}[(\omega^{-1} - 1)D - U]x^{(k)} + (\omega^{-1}D + L)^{-1}b, \quad (3.85)$$

$$x^{(k+1)} = (\omega^{-1}D + U)^{-1}[(\omega^{-1} - 1)D - L]x^{(k+1,1)} + (\omega^{-1}D + U)^{-1}b. \quad (3.86)$$

Sostituendo la (3.85) nella (3.86) e raccogliendo a fattore comune, posto:

$$F_\omega = (\omega^{-1}D + L)^{-1}[(\omega^{-1} - 1)D - U];$$

$$B_\omega = (\omega^{-1}D + U)^{-1}[(\omega^{-1} - 1)D - L];$$

$$f_\omega = B_\omega(\omega^{-1}D + L)^{-1}b;$$

$$g_\omega = (\omega^{-1}D + U)^{-1}b;$$

combinando insieme la (3.85) e la (3.86) si ha:

METODO SSOR $x^{(k+1)} = C_{SSOR}x^{(k)} + d_{SSOR}$ $C_{SSOR} = B_\omega F_\omega = I - [(\omega/(2 - \omega))(\omega^{-1}D + L)D^{-1}(\omega^{-1}D + U)]^{-1}A$ $d_{SSOR} = f_\omega + g_\omega = \omega(2 - \omega)(I + D^{-1}\omega U)^{-1}(I + D^{-1}\omega L)^{-1}D^{-1}b$	(3.87)
--	--------

Si osservi che la matrice di splitting del metodo SSOR è

$$M_{SSOR} = \left[\frac{\omega}{2 - \omega} (\omega^{-1}D + L)D^{-1}(\omega^{-1}D + U) \right],$$

e questa, se la matrice A è non singolare, è evidentemente non singolare. Quindi, il metodo SSOR è consistente. Relativamente alle proprietà di convergenza del metodo SSOR, si osservi che i risultati stabiliti dal Teorema 3.7 e dal Corollario 3.2 valgono anche per tale metodo. Inoltre è stato verificato sperimentalmente che la velocità di convergenza del metodo SSOR è “poco” sensibile al valore del parametro ω . Pertanto, la determinazione del valore ottimale di ω non è di importanza cruciale come per il metodo SOR. Un risultato tra i più significativi a tale riguardo è il seguente:

Teorema 3.10. *Dato il sistema lineare $Ax = b$, considerata la decomposizione (3.57) di A e posto $E = D^{-1}L$ e $F = D^{-1}U$, se*

$$\rho(LU) < 1/4, \tag{3.88}$$

allora il valore ottimale del parametro di rilassamento per il metodo SSOR è:

$$\omega_{opt} = \frac{2}{1 + \sqrt{2(1 - \rho(C_J))}}. \tag{3.89}$$

Inoltre, per $\omega = \omega_{opt}$ si ha:

$$\rho(C_{SSOR}) \leq \left(1 - \sqrt{\frac{1 - \rho(C_J)}{2}} \right) / \left(1 + \sqrt{\frac{1 - \rho(C_J)}{2}} \right). \tag{3.90}$$

♣ **Esempio 3.27.** Si consideri il sistema lineare (3.76) derivante dal problema al contorno per l'equazione di Poisson, analizzato nell'esempio 3.26, dove le funzioni f e g sono date dalle (3.80). Per tale sistema si verifica che vale la condizione (3.88) e quindi si è nelle ipotesi del Teorema 3.10. Utilizzando la (3.89), si ottiene, dalle (3.78) e (3.90), che:

$$\rho(C_{SSOR}) \leq \frac{1 - \sin t/2}{1 + \sin t/2}$$

e di conseguenza, per valori di $t \ll 1$, si ha, per la velocità di convergenza del metodo SSOR:

$$R_{SSOR} \geq t \tag{3.91}$$

♣

Sulla base della relazione (3.91) dell'esempio precedente e della stima (3.82) della velocità di convergenza del metodo SOR calcolata nell'esempio 3.26, si evince che il metodo SSOR risulta competitivo con il metodo SOR, per la risoluzione della classe di sistemi lineari considerata nell'esempio. Tuttavia, se la condizione (3.88) non è verificata, e ciò accade nella maggior parte delle applicazioni, l'efficacia del metodo SSOR diminuisce. Inoltre, è importante osservare che, ad ogni passo, il metodo SSOR richiede un numero di operazioni che è circa il doppio di quello richiesto dagli altri metodi analizzati fino ad ora. Per tali motivi, esso non è utilizzato frequentemente per la risoluzione di sistemi lineari. Tuttavia, il suo uso è più diffuso come "precondizionatore", ed, in particolare, come strumento per accelerare la convergenza di altri metodi iterativi più efficienti.

3.11.4 Accelerazione polinomiale

In questo paragrafo si introduce un'altra tecnica generale per accelerare la convergenza dei metodi iterativi del tipo (3.51). Tale tecnica, nota con il nome di **procedura di accelerazione polinomiale**, consiste nel costruire una nuova sequenza di vettori mediante combinazioni lineari dei vettori ottenuti con un fissato metodo iterativo, che è definito **metodo iterativo base** della procedura.

Sia $\{x^{(k)}\}$ la successione di vettori generati, a partire da un vettore iniziale $x^{(0)}$, da un metodo base:

$$x^{(k+1)} = Cx^{(k)} + d, \quad k \geq 0. \quad (3.92)$$

Si consideri la nuova sequenza di vettori $\{y^{(k)}\}$ definiti dalle combinazioni lineari:

$$y^{(k)} = \sum_{j=0}^k \alpha_{jk} x^{(j)}, \quad \alpha_{jk} \in \mathbb{R}, \quad k \geq 0. \quad (3.93)$$

Si vogliono determinare i coefficienti α_{jk} in modo che $y^{(k)}$ sia un'approssimazione più accurata della soluzione del sistema rispetto a $x^{(k)}$.

Indicata con x^* la soluzione del sistema $Ax = b$, si osserva, innanzitutto, che se $x^{(0)} = x^{(1)} = \dots = x^{(k)} = x^*$, allora deve risultare $y^{(k)} = x^*$. Sui coefficienti α_{jk} si impone quindi il seguente vincolo:

$$\sum_{j=0}^k \alpha_{jk} = 1, \quad \forall k \geq 0. \quad (3.94)$$

Indicati con

$$e^{(k)} = x^{(k)} - x^* \quad \text{ed} \quad \tilde{e}^{(k)} = y^{(k)} - x^*,$$

gli errori associati rispettivamente a $x^{(k)}$ e ad $y^{(k)}$, per le (3.93) e (3.94) si ha:

$$\tilde{e}^{(k)} = \sum_{j=0}^k \alpha_{jk} x^{(j)} - x^* = \sum_{j=0}^k \alpha_{jk} (x^{(j)} - x^*) = \sum_{j=0}^k \alpha_{jk} e^{(j)}. \quad (3.95)$$

Ricordando che:

$$e^{(j)} = C^j e^{(0)}, \quad j \geq 0,$$

con $e^{(0)} = x^{(0)} - x^*$, dalla (3.95) si ha:

$$\tilde{e}^{(k)} = \left(\sum_{j=0}^k \alpha_{jk} C^j \right) e^{(0)}, \quad \forall k \geq 0, \quad (3.96)$$

dove $\tilde{e}^{(0)} = e^{(0)}$. Introdotta il polinomio di grado k nella variabile reale z :

$$p_k(z) = \alpha_{0,k} + \alpha_{1,k}z + \dots + \alpha_{k,k}z^k, \quad \text{con } p_k(1) = 1,$$

la (3.96) può essere riscritta nella forma:

$$\tilde{e}^{(k)} = p_k(C)\tilde{e}^{(0)}, \quad \forall k \geq 0, \quad (3.97)$$

dove $p_k(C) \in \mathbb{R}^{n \times n}$ ha la seguente espressione:

$$p_k(C) = \alpha_{0,k}I + \alpha_{1,k}C + \dots + \alpha_{k,k}C^k.$$

Il problema fondamentale legato all'utilizzo della procedura di accelerazione descritta è quello di trovare una successione di polinomi

$$\{p_k(z) : p_k(1) = 1\}$$

in modo da rendere minimo l'errore $\tilde{e}^{(k)}$, $\forall k \geq 0$. Considerando la norma euclidea, dalla (3.97) si ha:

$$\|\tilde{e}^{(k)}\|_2 \leq \|p_k(C)\|_2 \|\tilde{e}^{(0)}\|_2, \quad \forall k \geq 0,$$

e, quindi, per rendere piccola la norma dell'errore è necessario determinare una successione di polinomi $\{p_k(z) : p_k(1) = 1\}$ tali che $\|p_k(C)\|_2$ sia piccola. Ciò conduce al seguente problema:

$$\min_{p_k(1)=1} \|p_k(C)\|_2, \quad \forall k \geq 0. \quad (3.98)$$

Si analizza nel prossimo paragrafo una possibile soluzione di tale problema.

3.11.5 Accelerazione polinomiale di Chebyshev

Si assuma che la matrice di iterazione C del metodo base (3.92) della procedura sia simmetrica. I suoi autovalori $\{\lambda_i\}_{i=1}^n$ sono quindi reali e, nell'ipotesi che il metodo base converga ($\rho(C) < 1$), si ha:

$$-1 < a = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n = b < 1, \quad (3.99)$$

con a e b , rispettivamente, il più piccolo ed il più grande tra gli autovalori di C . Si osservi che $\{p_k(\lambda_i)\}_{i=1}^n$ sono gli autovalori della matrice $p_k(C)$ (vedi **Teorema B.18**

dell'Appendice B, della Parte 1). Inoltre, dal fatto che C è simmetrica discende che lo è anche $p_k(C)$. Si ha quindi:

$$\|p_k(C)\|_2 = \rho(p_k(C)) = \max_{1 \leq i \leq n} |p_k(\lambda_i)|.$$

Dalla (3.99) si ha:

$$\|p_k(C)\|_2 = \max_{1 \leq i \leq n} |p_k(\lambda_i)| \leq \max_{a \leq z \leq b} |p_k(z)|.$$

Di conseguenza, il problema (3.98) è ricondotto alla determinazione di una successione di polinomi $\{p_k(z) : p_k(1) = 1\}$ che risolvano il problema:

$$\min_{p_k(1)=1} \left\{ \max_{a \leq z \leq b} |p_k(z)| \right\}, \quad \forall k \geq 0. \quad (3.100)$$

La soluzione del problema è unica ed è definita in termini dei **polinomi di Chebyshev**, come mostra il Teorema che segue, la cui dimostrazione (come anche le principali proprietà dei polinomi di Chebyshev) è riportata in Appendice (Teorema B.3).

Teorema 3.11. *Sia $T_k(z)$ il polinomio di Chebyshev di prima specie di grado k . Si consideri la funzione $w : [a, b] \rightarrow [-1, 1]$ così definita:*

$$w(z) = \frac{2z - (b+a)}{b-a}.$$

Sia \mathcal{P}_k l'insieme dei polinomi $p_k(z)$ di grado k nella variabile reale z tali che $p_k(1) = 1$. Il polinomio $H_k(z)$ definito da:

$$H_k(z) = \frac{T_k(w(z))}{T_k(w(1))}, \quad \forall k \geq 0, \quad (3.101)$$

è l'unico polinomio $\in \mathcal{P}_k$ che risolve il problema (3.100).

Il metodo iterativo base (3.92) e le (3.93), dove i coefficienti α_{jk} sono quelli dei polinomi (3.101), rappresentano la **procedura di accelerazione di Chebyshev**. Affinchè tale procedura costituisca un metodo effettivamente utilizzabile, è necessario tuttavia determinare una formula più efficiente della (3.93) per calcolare i vettori $y^{(k)}$; in tale formula, infatti, $y^{(k)}$ si ottiene a partire dai $k+1$ vettori $x^{(j)}$, $j = 0, \dots, k$. Pertanto, l'uso della (3.93) comporta un'elevata complessità sia di tempo che di spazio quando k è grande. Sfruttando la nota relazione di ricorrenza a tre termini dei polinomi di Chebyshev (Teorema B.2 dell'Appendice) si può ottenere una formula più conveniente della (3.93) da un punto di vista computazionale. Si ha infatti il seguente Teorema, la cui dimostrazione è riportata in Appendice (Teorema B.4):

Teorema 3.12. *I polinomi definiti in (3.101) soddisfano la relazione di ricorrenza:*

$$\begin{cases} H_0(z) = 1, & H_1(z) = \gamma z + 1 - \gamma; \\ H_{k+1}(z) = \beta_{k+1}(\gamma z + 1 - \gamma)H_k(z) + (1 - \beta_{k+1})H_{k-1}(z), & k \geq 1, \end{cases} \quad (3.102)$$

dove:

$$\begin{cases} \gamma = 2/(2 - (b + a)); \\ \beta_{k+1} = 2w(1)T_{(k)}(w(1))/T_{(k+1)}(w(1)). \end{cases} \quad (3.103)$$

Dalla relazione (3.102) stabilita dal Teorema precedente segue che anche i valori $y^{(k)}$ della procedura di accelerazione basata sui polinomi $H_{(k)}(z)$ possono essere ottenuti mediante una formula di ricorrenza a tre termini. Si ha infatti il seguente:

Corollario 3.3. *I vettori $y^{(k)}$ della procedura di accelerazione di Chebyshev si possono calcolare, a partire da un arbitrario vettore $y^{(0)}$, mediante la formula:*

$$\begin{cases} y^{(1)} = \gamma(Cy^{(0)} + d) + (1 - \gamma)y^{(0)}; \\ y^{(k+1)} = \beta_{k+1}[\gamma(Cy^{(k)} + d) + (1 - \gamma)y^{(k)}] + (1 - \beta_{k+1})y^{(k-1)}. \end{cases} \quad (3.104)$$

Dimostrazione Ricordando la (3.93), $y^{(k)} = \sum_{j=0}^k \alpha_{jk}x^{(j)}$, si ricava che $y^{(0)} = x^{(0)}$. Inoltre, dalle (3.102) si ha $H_1(x(1)) = \gamma x^{(1)} + 1 - \gamma$. Ma, d'altra parte, $H_1(x(1)) = \alpha_{10} + \alpha_{11}x^{(1)}$, da cui si ha $\alpha_{11} = \gamma$ e $\alpha_{01} = 1 - \gamma$. Poichè $y^{(1)} = \alpha_{01}x^{(0)} + \alpha_{11}x^{(1)}$, e tenuto conto che $x^{(1)} = Cx^{(0)} + d$, si ha la prima uguaglianza delle (3.104).

Si consideri ora l'errore associato al vettore $y^{(k)}$ in (3.93). Dalla (3.97) e dalla (3.102) si ha:

$$\tilde{e}^{(k+1)}(z) = \{\beta_{k+1}[\gamma C + (1 - \gamma)I]H_k(C) + (1 - \beta_{k+1})H_{k-1}(C)\}\tilde{e}^{(0)},$$

da cui, utilizzando ancora la (3.97), si ha:

$$\tilde{e}^{(k+1)}(z) = \beta_{k+1}[\gamma C + (1 - \gamma)I]\tilde{e}^{(k)} + (1 - \beta_{k+1})\tilde{e}^{(k-1)}. \quad (3.105)$$

Aggiungendo x^* ad entrambi i termini della (3.105) si ottiene:

$$y^{(k+1)}(z) = \beta_{k+1}[\gamma C + (1 - \gamma)I]y^{(k)} + (1 - \beta_{k+1})y^{(k-1)} - \beta_{k+1}\gamma(C - I)x^*. \quad (3.106)$$

Poichè x^* è soluzione del sistema associato al metodo di base (3.92), cioè $x^* = Cx^* + d$, si ha $(C - I)x^* = -d$, per cui dalla (3.106) si ottiene la relazione (3.104). ■

Osservazione 3.4. *I coefficienti β_{k+1} della relazione di ricorrenza (3.104) sono determinati dalla formula (3.103). Si osserva, tuttavia, che tali coefficienti possono essere scritti nella forma seguente, più vantaggiosa da un punto di vista computazionale:*

$$\begin{aligned} \beta_1 &= 1, & \beta_2 &= (1 - \frac{1}{2}\sigma^2)^{-1} \\ \beta_{k+1} &= (1 - \frac{1}{4}\sigma^2\beta_k)^{-1}, & k &\geq 2, \end{aligned}$$

con:

$$\sigma = 1/(w(1)) = (b - a)/(2 - (b + a)).$$

La relazione (3.104) mostra che non è necessario utilizzare un vettore ausiliario per memorizzare i vettori $x^{(k)}$ del metodo base (3.92). Infatti, anche se l'idea di partenza della procedura di accelerazione è quella di ottenere i vettori $y^{(k)}$ come combinazioni dei vettori $x^{(k)}$, la (3.104) consente di determinare direttamente gli $y^{(k)}$ in un modo simile al calcolo dei vettori $x^{(k)}$ nel metodo base. Si nota, tuttavia, che, mentre $x^{(k+1)}$ dipende solo da $x^{(k)}$, il vettore $y^{(k+1)}$ è ottenuto utilizzando i vettori calcolati nelle ultime due iterazioni, $y^{(k)}$ e $y^{(k-1)}$. Per tale motivo la procedura descritta è un **metodo iterativo a due passi**. Per la presenza di coefficienti il cui valore non è costante, ma dipende dall'iterazione (β_k), la procedura, al contrario dei metodi analizzati fino ad ora, appartiene alla classe dei **metodi non stazionari**. Inoltre, essa richiede l'uso di un vettore in più rispetto al metodo base. Ciò può pesare in maniera sensibile quando si vogliono risolvere sistemi lineari di grandi dimensioni.

È da osservare infine che la procedura di accelerazione in esame richiede anche la conoscenza di a e b che, si ricorda, sono il più piccolo ed il più grande autovalore della matrice di iterazione del metodo base. Nella maggior parte dei casi tali quantità non sono note, ed il loro calcolo è, in generale, computazionalmente dispendioso. Per tale motivo, sono state sviluppate opportune versioni della procedura, nelle quali si utilizzano stime di a e di b .

3.12 Criteri di arresto

Nel paragrafo 3.3 si è analizzato un criterio per arrestare un metodo iterativo, il quale è basato su una stima dell'errore. Si ritorna su tale argomento con l'obiettivo di discutere dei concetti di base per la progettazione di criteri di arresto affidabili, cioè criteri il cui utilizzo consenta di ottenere una desiderata accuratezza.

In generale, l'effettivo utilizzo di un metodo iterativo richiede l'arresto del procedimento quando l'errore è divenuto sufficientemente piccolo. Poichè non è possibile calcolare l'errore in maniera diretta dal momento che la soluzione non è nota, per progettare un criterio di arresto soddisfacente occorre stimare l'errore ad ogni iterazione.

Si premette la seguente definizione:

Definizione 3.13. (Residuo)

Dato il sistema lineare $Ax = b$, sia \bar{x} un vettore di dimensione n . Si definisce **residuo** del sistema in \bar{x} il vettore:

$$r = b - A\bar{x}.$$

Si osservi che una norma del residuo è una misura di quanto il vettore \bar{x} soddisfa il sistema.

Uno dei criteri di arresto più utilizzato si basa su una stima di quanto il residuo all'iterazione generica si è ridotto rispetto al residuo iniziale, cioè:

$$\frac{\|r^{(k)}\|}{\|r^{(0)}\|} \leq \text{To1}, \quad (3.107)$$

dove $r^{(k)}$ è il residuo del sistema $Ax = b$ in $x^{(k)}$. Il Teorema che segue stabilisce la relazione che sussiste tra l'errore e il residuo alla generica iterazione.

Teorema 3.13. *Alla k -ma iterazione del metodo iterativo:*

$$x^{(k+1)} = Cx^{(k)} + d, \quad k \geq 0,$$

si ha:

$$\|x^{(k)} - x^*\| \leq \|A^{-1}\| \cdot \|r^{(k)}\|, \quad (3.108)$$

dove $r^{(k)}$ è il residuo in $x^{(k)}$. Inoltre, se $x^{(0)}$ è il vettore iniziale, si ha:

$$\frac{\|x^{(k)} - x^*\|}{\|x^{(0)} - x^*\|} \leq \mu(A) \cdot \frac{\|r^{(k)}\|}{\|r^{(0)}\|}, \quad (3.109)$$

dove $\mu(A) = \|A\| \cdot \|A^{-1}\|$ è l'indice di condizionamento della matrice A .

Dimostrazione Da

$$r^{(k)} = b - Ax^{(k)}, \quad (3.110)$$

si ha:

$$x^{(k)} - x^* = -A^{-1}r^{(k)},$$

da cui segue la (3.108). Inoltre, dalla (3.110), per $k = 0$ si ha:

$$r^{(0)} = -A(x^{(0)} - x^*),$$

da cui segue che:

$$\|r^{(0)}\| = \|A(x^{(0)} - x^*)\| \leq \|A\| \cdot \|x^{(0)} - x^*\|.$$

Dall'ultima relazione e dalla (3.108) si ricava che:

$$\frac{\|x^{(k)} - x^*\|}{\|x^{(0)} - x^*\|} \leq \|A\| \cdot \|A^{-1}\| \cdot \frac{\|r^{(k)}\|}{\|r^{(0)}\|}$$

e, quindi, la (3.109). ■

Si osservi, innanzitutto, che la (3.109) mette in relazione la riduzione dell'errore alla generica iterazione con quella del residuo. Inoltre, in base alla (3.108), se si arresta il metodo quando vale la (3.107), per l'errore assoluto si ha la seguente maggiorazione:

$$\|x^{(k)} - x^*\| \leq \|A^{-1}\| \cdot \|r^{(0)}\| \cdot \text{To1}. \quad (3.111)$$

In generale, l'utilizzo del criterio (3.107) richiede ad ogni iterazione il calcolo del prodotto tra una matrice ed un vettore, e quindi un costo computazionale per iterazione circa uguale a n^2 flops. Tuttavia, per alcuni metodi, il residuo ad ogni iterazione è disponibile senza costi aggiuntivi. Ad esempio, per il metodo di Jacobi, il residuo al passo k può essere ottenuto senza dover calcolare un prodotto matrice-vettore; infatti per la i -ma componente del residuo si ha:

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

Si può verificare che per il metodo di G-S il residuo assume un'analogha espressione.

Osservazione 3.5. È importante osservare che, anche se criteri di arresto basati sul residuo, come (3.107), sono utilizzati frequentemente, un residuo piccolo non comporta automaticamente un errore piccolo. Ciò è dovuto in generale al fatto che, come stabilito dal Teorema 3.13, la relazione tra l'errore e il residuo dipende dagli autovalori e dal condizionamento della matrice A .

♣ **Esempio 3.28.** Per illustrare che la grandezza del residuo, $\|r^{(k)}\|$, può non essere un affidabile indicatore della grandezza dell'errore, si consideri il sistema lineare:

$$\begin{cases} 5x_1 + 0.0001x_2 + 0.0001x_3 = 5.0002 \\ 0.0001x_1 + 2.25000001x_2 + 0.00000001x_3 = 2.25010002 \\ 0.0001x_1 + 0.00000001x_2 + 0.00000001x_3 = 0.00010002 \end{cases}$$

la cui soluzione è $x_i^* = 1$, $i = 1, 2, 3$. Nella Tabella che segue sono riportati i valori di $\|r^{(k)}\|$ e $\|x^{(k)} - x^*\|$ nelle prime iterazioni del metodo di G-S (in particolare si è utilizzata la norma euclidea) partendo da $x_i^0 = 0$, $i = 1, 2, 3$. Si evince che ad un valore piccolo di $\|r^{(k)}\|_2$ non necessariamente corrisponde un errore con un comparabile ordine di grandezza. Tale fenomeno è dovuto agli autovalori della matrice A ($0.8 \cdot 10^{-8}, 2.25, 5$), in particolare al più piccolo, il cui valore è $\lambda_1 \simeq 10^{-8}$. Poichè $\|A^{-1}\|_2 = \lambda_1^{-1}$, dalla (3.108) si ha, per l'errore, la maggiorazione $\|x^{(k)} - x^*\|_2 \leq 10^8 \cdot \|r^{(k)}\|_2$, che giustifica i risultati ottenuti.

k	$\ r^{(k)}\ _2$	$\ x^{(k)} - x^*\ _2$
1	$1.60 \cdot 10^{-4}$	$0.40 \cdot 10^0$
2	$3.20 \cdot 10^{-5}$	$8.00 \cdot 10^{-2}$
3	$6.40 \cdot 10^{-6}$	$1.60 \cdot 10^{-2}$
4	$1.28 \cdot 10^{-6}$	$3.20 \cdot 10^{-3}$
5	$2.56 \cdot 10^{-7}$	$6.40 \cdot 10^{-4}$
6	$5.12 \cdot 10^{-8}$	$1.28 \cdot 10^{-4}$
7	$1.02 \cdot 10^{-8}$	$2.56 \cdot 10^{-5}$
8	$2.04 \cdot 10^{-9}$	$5.12 \cdot 10^{-6}$
9	$4.09 \cdot 10^{-10}$	$1.02 \cdot 10^{-6}$
10	$8.19 \cdot 10^{-11}$	$2.04 \cdot 10^{-7}$



Si osservi, infine, che l'affidabilità del criterio di arresto (3.107) dipende anche dalla scelta di $x^{(0)}$. Infatti, se $x^{(0)}$ è “vicino” alla soluzione, la quantità $\|r^{(0)}\|$ può essere molto piccola e si corre il rischio di eseguire iterazioni inutili, cioè che non migliorano l'accuratezza del risultato; viceversa, se il punto iniziale è “distante” dalla soluzione, e quindi $\|r^{(0)}\|$ è un numero grande, il metodo può essere arrestato “troppo presto”, ottenendo un'approssimazione poco accurata della soluzione.

♣ **Esempio 3.29.** Si consideri il sistema di equazioni lineari:

$$\begin{cases} 10000x_1 + 1x_2 + 4x_3 = 10005 \\ 0.5x_1 + 1x_2 + 0.4x_3 = 1.9 \\ 0.2x_1 + 0.6x_2 + 0.3x_3 = 1.1 \end{cases}$$

la cui soluzione è $x_i^* = 1$, $i = 1, 2, 3$. Applicando a tale sistema il metodo di Jacobi con $x_i^{(0)} = 0$, $i = 1, 2, 3$ ed utilizzando come criterio di arresto la (3.107) con $\text{To1} = 10^{-6}$, il metodo si ferma dopo 71 iterazioni e gli ultimi valori calcolati, arrotondati alla sesta cifra decimale, sono: $x_1^{(71)} = 1.000000$, $x_2^{(71)} = 1.000380$, $x_3^{(71)} = 1.001114$. Si osserva che per le componenti x_2 e x_3 non si è ottenuta l'accuratezza desiderata. Poichè $\|r^{(0)}\|_2 \simeq 10005$ e $\|A^{-1}\|_2 \simeq 21.119$, dalla (3.111) si ha, per l'errore, la maggiorazione $\|x^{(71)} - x^*\|_2 \leq 0.211$. Si osservi che $\|x^{(71)} - x^*\|_2 = 0.001177$. ♣

3.13 Alcuni problemi da risolvere con il calcolatore

Esercizio 1 Sia $A \in \mathbb{R}^{8 \times 8}$:

$$A = \begin{bmatrix} 8 & -2 & 0 & 0 & \cdots \\ -2 & 8 & -2 & 0 & \cdots \\ 0 & -2 & 8 & -2 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & -2 & 8 \end{bmatrix}$$

la matrice dei coefficienti di un sistema lineare:

$$Ax = b.$$

1. Verificare che il metodo di Jacobi è convergente.
2. Verificare che il metodo di Gauss-Seidel è convergente.
3. Detto C_J la matrice di iterazione del metodo di Jacobi e C_{GS} quella del metodo di Gauss-Seidel, verificare che tra i raggi spettrali di tali matrici sussiste la relazione:

$$\rho(C_{GS}) = \rho^2(C_J).$$

Esercizio 2 Sia $A \in \mathbb{R}^{8 \times 8}$:

$$A = \begin{bmatrix} 7 & 1 & 0 & 0 & \cdots \\ 1 & 7 & 1 & 0 & \cdots \\ 0 & 1 & 7 & 1 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 1 & 7 \end{bmatrix}$$

la matrice dei coefficienti di un sistema lineare:

$$Ax = b.$$

Supponendo di utilizzare il metodo iterativo SOR:

1. verificare che il metodo è convergente;
2. trovare il valore ottimale del parametro di rilassamento ω_{opt} del metodo SOR;
3. calcolare la matrice di iterazione $C_{\omega_{opt}}$ del metodo SOR.

Esercizio 3 Sia $A \in \mathbb{R}^{3 \times 3}$:

$$A = \begin{bmatrix} 8 & -1 & 0 \\ -2 & 4 & -1 \\ 3 & -1 & 7 \end{bmatrix}$$

la matrice dei coefficienti di un sistema lineare:

$$Ax = b.$$

Supponendo di utilizzare il metodo iterativo di Jacobi:

1. verificare che il metodo è convergente;
2. costruire la matrice di iterazione C_J ;
3. determinare il numero k minimo di iterazioni affinché l'errore sulla soluzione sia minore di 10^{-5} .

Esercizio 4 Sia $A \in \mathbb{R}^{3 \times 3}$:

$$A = \begin{bmatrix} 10 & -3 & 1 \\ -3 & 10 & 4 \\ 1 & 4 & 10 \end{bmatrix}$$

la matrice dei coefficienti di un sistema lineare:

$$Ax = b.$$

Supponendo di utilizzare il metodo iterativo di Gauss-Seidel:

1. verificare che il metodo è convergente;
2. costruire la matrice di iterazione C_{GS} ;
3. determinare il numero k minimo di iterazioni affinché l'errore sulla soluzione sia minore di 10^{-5} .

Esercizio 5 Si consideri un sistema lineare:

$$Ax = b, \quad b \in \mathbb{R}^3$$

dove la matrice dei coefficienti è

$$A = \begin{bmatrix} -1 & 0 & a \\ f & -1 & 0 \\ 0 & c & -1 \end{bmatrix},$$

con a , f , e c numeri reali. Studiare la convergenza del metodo iterativo di Gauss-Seidel al variare dei parametri a , f , e c .

Esercizio 6 Si consideri un sistema lineare:

$$Ax = b$$

dove

$$A = \begin{bmatrix} 10 & -1 & 2 & 3 \\ 1 & -1 & -1 & 2 \\ 2 & 3 & 20 & -1 \\ 3 & 2 & 1 & 20 \end{bmatrix} \quad \text{e} \quad b = \begin{bmatrix} 14 \\ 12 \\ 24 \\ 26 \end{bmatrix}.$$

Dopo avere discusso la convergenza dei metodi iterativi di Jacobi e di Gauss-Seidel per ciascun metodo, sviluppare una function `MATLAB` per risolvere il sistema lineare. Inoltre,

1. determinare le velocità asintotiche di convergenza per ciascun metodo;
2. verificare i risultati ottenuti con le stime sperimentali dedotte applicando le function `MATLAB` elaborate.

Esercizio 7 Sviluppare un algoritmo basato sul metodo di Gauss-Seidel e risolvere i sistemi lineari

$$A_1 x = b_1,$$

dove

$$A_1 = \begin{bmatrix} 3 & 1 & 1 \\ 1 & 3 & -1 \\ 3 & 1 & -5 \end{bmatrix} \quad \text{e} \quad b_1 = \begin{bmatrix} 5 \\ 3 \\ -1 \end{bmatrix}$$

e

$$A_2 x = b_2,$$

dove

$$A_2 = \begin{bmatrix} 3 & 1 & 1 \\ 3 & 1 & -5 \\ 1 & 3 & -1 \end{bmatrix} \quad \text{e} \quad b_2 = \begin{bmatrix} 5 \\ -1 \\ 3 \end{bmatrix}.$$

Confrontare le soluzioni ottenute con quelle calcolate mediante l'algoritmo di Gauss senza pivoting. Commentare i risultati.

Esercizio 8 Utilizzare l'algoritmo sviluppato nell'esempio precedente per risolvere il sistema

$$Ax = b$$

dove

$$A_1 = \begin{bmatrix} 0.96326 & 0.81321 \\ 0.81321 & 0.68654 \end{bmatrix} \quad \text{e} \quad b = \begin{bmatrix} 0.88824 \\ 0.74988 \end{bmatrix},$$

utilizzando come punto iniziale $x_0 = (0.3316, 0.7)$; discutere cosa succede alla soluzione del sistema.

A. Murli

Bibliografia

- [1] Axelsson O. - *Iterative Solution Methods* - Cambridge University Press, 1994.
- [2] Barlett R. et al. - *Templates for the Solution of Linear Systems. Building Blocks for Iterative Methods* - SIAM, Philadelphia, 1993.
- [3] Cheney W., Kincaid D.R. - *Numerical Mathematics and Computing* - 3rd ed., Pacific Grove, CA, Brooks/Cole, 1994.
- [4] Greenbaum A. - *Iterative Methods for Solving Linear Systems* - SIAM, Philadelphia, 1997.
- [5] Hageman L.A., Young D.M. - *Applied Iterative Methods* - Academic Press, 1981.
- [6] Kelley C.T. - *Iterative Methods for Linear and Nonlinear Equations* - SIAM, Philadelphia, 1995.
- [7] Lancaster P., Tismenetsky M. - *The Theory of Matrices* - second edition, Academic Press, 1985.
- [8] Ortega J.M., Rheinboldt W.C. - *Iterative Solution of Nonlinear Equations in Several Variables* - Academic Press, 1970.
- [9] Ortega J.M. - *Matrix Theory* - Plenum Press, 1988.
- [10] Saad Y. - *Iterative Methods for Sparse Linear Systems* - PWS Publishing Co, Boston, 1996.
- [11] Stewart G.W. - *Afternotes on Numerical Analysis* - SIAM, Philadelphia, 1996.
- [12] Trefethen L.N., Bau D. - *Numerical Linear Algebra* - SIAM, Philadelphia, 1997.
- [13] Varga R.S. - *Matrix Iterative Analysis* - Prentice Hall, Englewood Cliffs, NJ, 1962.
- [14] Young D.M. - *Iterative Solution of Large Linear Systems* - Academic Press, NY, 1971.