Università
Ca'Foscari
Venezia

Corso di Laurea magistrale
in Statistica Per l'Impresa

Tesi di Laurea

Ca' Foscari
Dorsoduro 3246
30123 Venezia

Precondizionamento per sistemi lineari simmetrici a grande dimensione

Relatore
Prof. Giovanni Fasano

Laureando
Mirko Scavetta
Matricola 810940

Anno Accademico
2012 / 2013

Ringraziamenti

Desidero ringraziare il mio relatore, il prof. Giovanni Fasano, che mi ha seguito nella realizzazione di questa tesi.

Un grazie particolare alla mia famiglia, che mi ha sempre sostenuto sia moralmente che economicamente, per arrivare fino a questo punto della mia vita.

Indice

Prefazione

L'Algebra lineare è per definizione "*un campo della matematica che si occupa dello studio dei vettori, spazi vettoriali (o spazi lineari),trasformazioni lineari e sistemi di equazioni lineari* ".
Essa rappresenta sia un efficace codice comune con cui si possono formulare problemi di diversa natura, sia una vasta gamma di metodi di risoluzione per problemi che si incontrano in molte applicazioni. Infatti, ogni disciplina come la fisica, le scienze naturali e biologiche, l'ingegneria, l'economia e la finanza genera problemi che richiedono di essere risolti attraverso l'aiuto dell'algebra lineare.

Nonostante i suoi principi teorici siano consolidati ormai da molto tempo, gli aspetti computazionali di questa particolare disciplina sono molto più recenti e in continuo sviluppo. Il pensiero algebrico e l'evoluzione culturale umana si sono sviluppate nel tempo in maniera parallela e connessa.
Le prime manifestazioni documentate del pensiero algebrico risalgono addirittura attorno al 2000 a.C. ai tempi degli antichi egizi, i quali riuscirono a creare un sistema di numerazione basato sui geroglifici per risolvere diverse classi di problemi.
Mentre nel libro cinese, scritto intorno al 200 a.C., *Jiuzhang suansh* ("*Scritti sui numeri in nove capitoli")* vi sono descritti i primi procedimenti matematici per risolvere sistemi di equazioni lineari. In questo particolare testo vi erano descritti circa 246 problemi ripartiti a seconda degli algoritmi risolutivi, in cui i numeri venivano rappresentati attraverso delle semplici bacchette. Notevoli progressi avvennero con il matematico greco Diofanto (200 - 284 d.C.), considerato tuttora il padre dell'algebra, il quale si spinse fino alla risoluzione numerica delle equazioni di 2° grado. Dai suoi studi derivano molti dei termini usati tuttora nel linguaggio algebrico, come per esempio il termine *potenza*. Le opere di Diofanto hanno influenzato in maniera evidente l'algebra araba, infatti nel IX secolo il matematico arabo Al-Khuwarzimi, oltre ad introdurre il termine *àlàgabr* da cui deriva algebra, scrisse uno dei primi libri di algebra araba, che conteneva un' esposizione sulla teoria fondamentale delle equazioni.
Il primo ad introdurre il termine, di origine latina, *equazione* nel linguaggio matematico fu il matematico pisano Leonardo Fibonacci (1170-1240) nel suo testo *Liber Abaci* (1202). La sua opera è un ponderoso trattato di aritmetica e algebra con la quale, all'inizio del XIII secolo, introdusse in Europa il sistema numerico decimale indo-arabico e i principali metodi di calcolo ad esso relativi.

Nell'Europa del'500, e in particolare in Italia, si diffuse un forte interesse per l'algebra. In questo periodo molti matematici, tra cui Niccolò Tartaglia (1550-1557), entrarono in competizione per risolvere alcuni problemi, tra cui le equazioni di grado superiore al secondo. Un importante sviluppo dell' algebra del XVI avvenne con l'introduzione dei simboli per indicare le incognite, le potenze algebriche e le operazioni, com'è dimostrato nel III libro della *Géometrie* di Cartesio(1596-1650).
Il termine *matrice* comparve, nei primi secoli a.C., , per la prima volta nel trattato matematico cinese *Jiuzhang suansh,* in cui si descriveva come risolvere un problema sotto forma di un sistema tabellare, dove i coefficienti di ogni equazione sono disposti parallelamente in senso verticale (a differenza dalla notazione odierna) in modo da applicare successivamente una serie di operazioni per risolvere più facilmente il problema in esame. Successivamente, dopo quasi due millenni , nel 1683, il matematico giapponese S. Kowa (1637-1708) e il matematico tedesco Leibniz (1646-1716) migliorarono la tecnica cinese introducendo il concetto attualmente noto come *determinante*. Nella prima metà del XVIII secolo Cramer (1704-1752) diede un importante contribuito nella risoluzione di sistemi lineari creando un algoritmo per trovare il determinante per matrici quadrate di qualunque ordine, la cosiddetta *regola di Cramer*. In questo secolo fu scritto molto sul concetto di determinante, che diventò lo strumento più importante per risolvere i sistemi lineari.
L'algebra ricevette nei primi anni del XIX secolo un grande impulso grazie agli studi effettuati dal matematico tedesco Carl Friedrich Gauss (1777-1855). Egli riuscì a creare uno specifico algoritmo per determinare le soluzioni di un sistema di equazioni lineari, il ben noto *metodo di eliminazione di Gauss*. Nel 1848 solo grazie agli studi del matematico inglese Sylvester (1814-1897) e del suo collega A.Cayley (1821-1895) venne introdotto il termine vero e proprio di matrice, inteso come elemento distinto dal concetto di determinante, insieme alle operazioni algebriche fra matrici.
Infatti è il lavoro, del 1858, "*A memoir on the theory of matrices"* di Cayley, che segna la nascita della teoria delle matrici e dell'algebra lineare.

L'Algebra lineare è sempre più presente nella vita di tutti i giorni, infatti molti problemi reali possono essere risolti attraverso questa particolare disciplina matematica. Questa tesi si concentra su un particolare ramo dell'algebra lineare: la risoluzione di sistemi di equazioni lineari simmetrici. Molti fenomeni di natura fisica, economica e sociale possono essere definiti da modelli "lineari", in quanto non sono influenzati da distorsioni, turbolenze e fenomeni caotici in generale.
Dato che il sistema lineare è uno strumento di base della matematica applicata, vi è la necessità di avere a disposizione una vasta gamma di algoritmi dai quali poter scegliere il più adatto per risolvere un determinato problema.
L'algoritmo più efficiente dovrà essere scelto oltre che in base alla sua stabilità e suo costo computazionale, anche sulla base delle caratteristiche, al numero ed al tipo delle equazioni del sistema da risolvere.

Lo scopo della tesi sarà quello di andare a risolvere una determinata classe di sistemi lineari con determinate caratteristiche. Questi sistemi, del tipo $Ax = b$, avranno la matrice dei coefficienti $A \epsilon R^{nxn}$ simmetrica ($A = A^T$), definita positiva ($A \succ 0$) con n (numero degli elementi) molto grande. Verrà analizzata e studiata una particolare tecnica per risolvere più agevolmente questo tipo di sistemi la cosiddetta tecnica del *precondizionamento*.

La tesi è sviluppata in 6 capitoli. In particolare si avrà:

- Nel capitolo 1 verrà fatta una breve introduzione sui sistemi lineari in generale. Si partirà dalla definizione più semplice di una funzione lineare, per poi vedere che cos'è una equazione lineare, per finire a vedere come viene definito in generale un sistema lineare in tutti i suoi elementi, anche in forma matriciale.

- Nel capitolo 2 viene effettuata un'analisi dei metodi di risoluzione per sistemi lineari. Si andrà a vedere che questi metodi di calcolo si suddividono in due grandi categorie: i metodi diretti ed i metodi iterativi. I metodi diretti che si andranno ad analizzare sono il cosiddetto *metodo di eliminazione di Gauss* e la sua forma avanzata, ovvero la *fattorizzazione LU*. Invece dei metodi iterativi verranno descritti i tre metodi, basati sulla decomposizione della matrice dei coefficienti *A*, ovvero quello di *Jacobi*, quello di *Gauss-Seidel* e infine il metodo del *rilassamento*. Alla fine del capitolo vi è una tabella in cui viene riportato il costo computazionale di ogni metodo di risoluzione di un sistema lineare descritto precedentemente.

- Il capitolo 3 è completamente dedicato ad un particolare metodo iterativo: il metodo del *Gradiente Coniugato*. Questo metodo afferma che risolvere un sistema lineare in cui la matrice dei coefficienti è definita positiva equivale a minimizzare una forma quadratica ad essa relativa. Prima verrà fatta una breve introduzione sul metodo del gradiente per poi andare a vedere in maniera più approfondita, la sua forma perfezionata, ovvero il metodo del gradiente coniugato, e si andranno ad analizzare le relative proprietà.

- Nel capitolo 4 verrà descritta la tecnica del cosiddetto *precondizionamento*. Questa tecnica serve per far convergere i metodi iterativi più velocemente ad una soluzione del sistema lineare in esame. Per prima cosa verrà detto cosa è un precondizionatore in generale, a cosa serve e quali sono i suoi requisiti fondamentali. Successivamente vengono descritti i diversi precondizionatori che si possono trovare nella letteratura matematica; in particolar modo verrà data molto attenzione ad una classe di precondizionatori costruiti attraverso l'utilizzo del metodo del Gradiente Coniugato.
  I precondizionatori descritti saranno quelli proposti dagli studiosi Gratton, Sartenaer e Tshimanga, i *Limited Memory Preconditioners (LMP),* i quali sono

molto adatti a risolvere sistemi lineari simmetrici a grandi dimensioni. Inoltre sono gli unici in letteratura adatti a questo tipo di problemi. Alla fine verrà presentato il precondizionatore proposto da G. Fasano e M. Roma, per la risoluzione di sistemi lineari indefiniti di grandi dimensioni. La costruzione di questo precondizionatore avviene attraverso una decomposizione della matrice di sistema ottenuta attraverso un metodo di Krylov, il metodo del Gradiente Coniugato spiegato nel capitolo 3, avente determinati requisiti. Questo particolare precondizionatore è ottenuto memorizzando solo alcuni vettori, senza calcolare il prodotto di matrici o matrici inverse.

- Nel capitolo 5, vengono fatte delle analisi numeriche, attraverso il software matematico *Matlab*, su alcune sistemi lineari simmetrici, presi dal database di *Matrix Market* (*http://math.nist.gov/MatrixMarket/index.html*), per testare le caratteristiche e l'efficacia dei due precondizionatori descritti nel capitolo precedente.

- Nell'ultimo capitolo, ci sono considerazioni conclusive sullo svolgimento della tesi e sui risultati ottenuti. Nell'appendice vengono riportate infine alcune tabelle sul numero di condizionamento dei vari sistemi lineari testati precedentemente, e gli algoritmi del gradiente coniugato non precondizionato e precondizionato.

Cap.1 - Introduzione sui sistemi lineari

L'*Algebra Lineare* ha come obiettivo quello di fornire strumenti, metodi e tecniche per la risoluzione di problemi reali di analisi numerica. Molti problemi effettivi, che siano di natura economica aziendale o scientifica, sono rappresentati attraverso modelli matematici per essere risolti nella loro complessità.
Spesso i modelli matematici sono di tipo *lineare*, ma anche i modelli più complessi (non lineari), di solito più attinenti alla descrizione della realtà, sono spesso "linearizzati", nel senso che la loro risoluzione è ricondotta alla risoluzione di problemi lineari (Lipschutz,1992 [9]). Molti problemi concreti possono essere descritti mediante modelli basati su sistemi di equazioni lineari, per risolverli basterà scegliere l'algoritmo più adatto in termini di costi e tempo.
In questo capitolo si andrà a vedere,anche attraverso esempi, come viene definito un sistema lineare in tutti i suoi elementi. Per prima cosa verrà spiegato cosa si intende con il termine *lineare* rispetto ad una funzione e successivamente cosa è una vera e propria equazione lineare di primo grado.

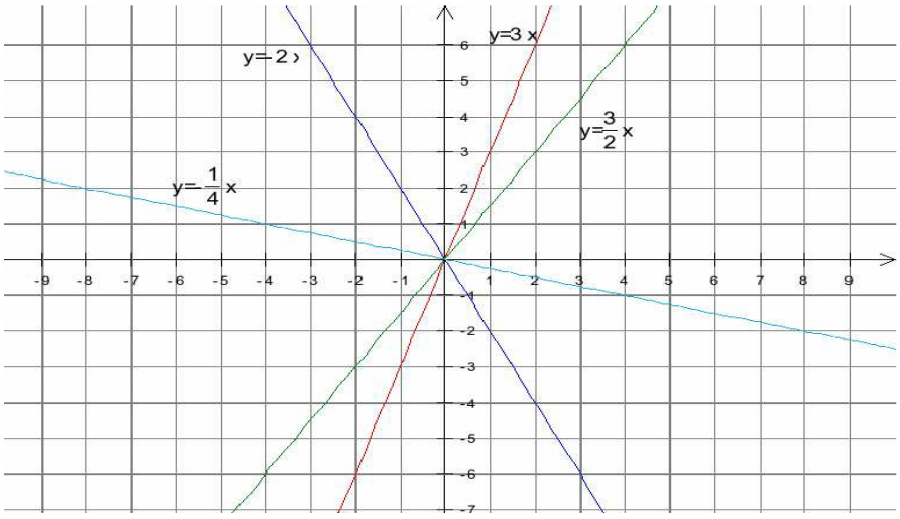
1.1. Funzione lineare

Una funzione $f : R^n \to R$ è *lineare* in R^n se soddisfa le seguenti relazioni

1) $f(x+y) = f(x) + f(y)$ $\qquad \forall\, x, y \in R^n$
(1.1.)

2) $f(\alpha x) = \alpha f(x)$ $\qquad \forall\, x \in R^n, \forall\, \alpha \in R$
(1.2.)

La sua rappresentazione grafica risulta essere una retta del tipo $y = a \cdot x$ con $a \in R$.
Nel seguente piano cartesiano sono rappresentate alcune funzioni lineari.



1.2. Equazione lineare

Per iniziare, un'equazione è semplicemente un uguaglianza tra due espressioni algebriche contenenti una o più variabili, dette incognite, verificata solo per determinati valori attribuiti a queste.
Un'*equazione lineare* è invece un'equazione algebrica di primo grado, dove il grado massimo delle incognite è uno. Ciascuna equazione lineare può essere riconducibile, tramite trasformazioni equivalenti, nella cosiddetta *forma normale*:

$$a \cdot x + b = 0 \qquad (1.3.)$$

Dove *a* e *b* sono numeri reali. Si osserva che :

- se $a \neq 0$, l'equazione ammette una e una solo soluzione,corrispondente a $x = -\frac{b}{a}$;
- se $a = 0$ e $b \neq 0$, l'equazione non ammette soluzioni;
- se $a = 0$ e $b = 0$, l'equazione è soddisfatta per ogni valore di x, ed è più propriamente un identità.

In generale un'equazione lineare in *n* incognite $x_1, x_2, \dots, x_n$ viene definita nel seguente modo :

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n + k = 0 \qquad (1.4.)$$

dove $a_1, a_2 \dots, a_n, k \in R$. In generale, nello spazio euclideo *n*-dimensionale, l'insieme delle soluzioni di un'equazione lineare in *n* incognite viene rappresentato graficamente da un iperpiano.

1.3. Sistemi lineari

Prendendo ad esempio le due equazioni lineari:

$$x + y = 3$$
$$x - y = 1$$

mettere a "sistema" significa chiedersi se esistono coppie ordinate $(x; y)$ che soddisfano contemporaneamente le due equazioni.
Un *sistema di equazioni* perciò sarà un insieme di equazioni in cui compaiono le stesse incognite, per le quali ci chiediamo quali siano le soluzioni comuni. Per indicare un sistema, si scrivono le equazioni in colonna, racchiuse da una parentesi graffa:

$$\begin{cases} 3x + 5y = 0 \\ 2x + 3y = 1 \end{cases}$$

Verranno presentati semplici esempi di un sistema di equazioni (Robbiano, 2006 [12]).

Esempio 1. La dieta

Si supponga di voler preparare una colazione con burro, pane e latte, in modo da ottenere 600 calorie, 15 grammi di proteine e 40 grammi di grassi. Data la seguente tabella che mostra il numero di calorie, le proteine (in grammi) e i grassi (in grammi), forniti da un grammo di ciascun alimento sopraindicato.

	burro	*pane*	*latte*
calorie	6.18	3.00	0.35
proteine	0.005	0.067	0.04
grassi	0.73	0.02	0.02

Se si indica con x_1, x_2, x_3 rispettivamente il numero dei grammi di burro, pane e latte, quella che si cerca non è altro che la soluzione del seguente sistema di equazioni:

$$\begin{cases} 6.18x_1 + 3x_2 + 0.35x_3 = 600 \\ 0.005x_1 + 0.067x_2 + 0.04x_3 = 15 \\ 0.73x_1 + 0.02x_2 + 0.02x_3 = 40 \end{cases}$$

Esempio 2. Il trasporto

Si supponga di avere due fabbriche F_1, F_2 che producono ogni giorno rispettivamente 130, 250 automobili. Supponiamo che le fabbriche debbano fornire le loro auto a due rivenditori R_1, R_2, che richiedono rispettivamente 74 e 306 auto. Se chiamiamo x_1, x_2 le quantità di auto che dalla fabbrica F_1 saranno trasportate ai rivenditori R_1, R_2 e con y_1, y_2 le quantità di auto che dalla fabbrica F_2 saranno trasportate ai rivenditori R_1, R_2, il nostro problema del trasporto viene tradotto nel seguente sistema di equazioni:

$$\begin{cases} x_1 + x_2 = 130 \\ y_1 + y_2 = 250 \\ x_1 + y_1 = 74 \\ x_2 + y_2 = 306 \end{cases}$$

Si può notare che in tutti e due i problemi vengono usati sistemi definiti lineari. Un sistema si dice *lineare* se tutte le equazioni che lo compongono sono lineari.

Un sistema di equazioni lineari generico composto da un numero *m* di equazioni lineari e da un numero *n* di incognite $x_1, x_2, \dots, x_n$ viene presentato nella seguente *forma standard* (Lipschutz,1992 [9]):

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \end{cases} \qquad (1.5.)$$

Si può osservare che gli elementi che contraddistinguono il sistema sono i dati del problema, rappresentati dai coefficienti a_{ij} e dai termini noti b_i con $i = 1 \dots m$ e $j = 1 \dots n$. Da notare che la notazione che si è usata per i coefficienti a_{ij} del sistema è chiamata a doppio pedice, in cui il primo indica l'equazione corrispondente, mentre il secondo a quale incognita si riferisce (ad esempio a_{12} è il nome del coefficiente dell'incognita x_2 nella prima equazione).

Come già detto per trovare la soluzione del seguente sistema a equazioni lineari basterà trovare le loro soluzioni comuni. In modo più preciso si dice che soluzione di detto sistema è un insieme di valori delle incognite $x_1 = k_1, x_2 = k_2, \dots, x_n = k_n$, ovvero una *n*-upla $u = (k_1, k_2, \dots, k_n)$ di costanti, soluzione di ogni equazione del sistema.
Risolvere un sistema significa determinare l'insieme di tutte le soluzioni: un sistema è *impossibile* se non ha soluzioni, è *determinat*o se ha un numero finito di soluzioni, è *indeterminato* se ha un numero infinito di soluzioni.

1.4. Formalismo Ax = b (sistemi lineari in forma matriciale)

Riesaminando un generico sistema lineare si può notare che può essere scomposto sostanzialmente in tre elementi: una matrice e due vettori. Infatti un sistema di equazioni lineari può essere rappresentato nella seguente forma matriciale:

$$Ax = b \tag{1.6.}$$

con

- $A \in R^{mxn}$, matrice dei coefficienti;
- $x \in R^n$, vettore delle incognite;
- $b \in R^m$, vettore dei termini noti,

potendo scrivere il sistema in modo esteso:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \tag{1.7.}$$

Come si può notare viene utilizzato il prodotto tra matrici, più precisamente viene calcolato il prodotto tra la matrice dei coefficienti $A \in R^{mxn}$ e il vettore delle incognite $x \in R^n$, che viene visto come una matrice *nx1*. Questo prodotto viene definito anche *prodotto righe per colonne* perché darà come risultato una nuova matrice in cui il suo elemento *i*-esimo è la somma dei prodotti della *i*-esima riga della matrice *A* per gli elementi della *k*-esima riga di *x*, che in questo caso è un semplice vettore.

Un sistema lineare del tipo $Ax = b$, in cui la matrice dei coefficienti A è una matrice quadrata (ovvero il numero di equazioni e pari al numero delle incognite), ammette *una e una sola* soluzione se e solo se è verificata una delle seguenti condizioni:

- il determinante della matrice A è diverso da zero (ovvero la matrice A è invertibile); la soluzione sarà data da $x = A^{-1}b$;
- il rango (ovvero il numero di righe o colonne linearmente indipendenti) di A è uguale a n;
- il sistema è omogeneo, ovvero il vettore dei termini noti b è un vettore nullo ($Ax = 0$), ed A ha rango pari ad n. In tal caso il sistema ammette la solo soluzione banale $x_i = 0$ con $i = 1, \dots, n$.

1.5. Sistemi sovradeterminati e sottodeterminati

Oltre al caso in cui il numero delle equazioni sia uguale al numero delle incognite (nel sistema $Ax = b$, la matrice dei coefficienti A è una matrice quadrata), si possono verificare i due seguenti casi:

1. se il numero delle equazioni *m* è maggiore del numero *n* delle incognite si parla di *sistemi sovradeterminati*;
2. se il numero delle equazioni *m* è minore del numero *n* delle incognite si parla di *sistemi sottodeterminati*.

Nel caso il problema sia sovradeterminato potrebbe non esistere la soluzione del problema, mentre nel caso in cui sia sottodeterminato la soluzione esisterà ma non sarà unica. In questi due casi si può parlare di *problema mal posto* e perciò per trovare una soluzione in senso classico si dovrà trasformarlo in un *problema ben posto*.

Se si considera il primo caso, non essendo possibile verificare contemporaneamente tutte le possibile equazioni si andrà per esempio a trovare il vettore delle variabili che minimizza lo scarto (o residuo) quadratico dal vettore dei termini noti; in pratica si andrà a trovare una soluzione del sistema nel senso dei minimi quadrati, ossia nel sistema $Ax = b$ quel vettore x^* che minimizza la norma Euclidea del residuo $r = b - Ax$:

$$min_{x \epsilon R^n} \|b - Ax\|_2^2 \, . \qquad (1.8.)$$

Cap.2 – Metodi di risoluzione di sistemi lineari

Come già è stato detto nel capitolo precedente la risoluzione dei sistemi di equazioni lineari ha un ruolo fondamentale nell'analisi matematica, visto che molti problemi che ci si trova ad affrontare nella vita di tutti i giorni possono essere risolti attraverso semplici modelli matematici.
In questo capitolo si andranno a studiare alcuni algoritmi che esistono per risolvere un sistema di equazioni lineari. Si vedrà che sarà molto importante la scelta dell'algoritmo più adatto ed efficiente in base alle caratteristiche del sistema, al numero ed al tipo delle equazioni.
Esistono sostanzialmente due diverse categorie di metodi di risoluzione di un sistema lineare (Quarteroni, 2008 [11]) e sono:

1. *Metodi diretti*
2. *Metodi iterativi*

Con i metodi diretti si arriva a trovare una soluzione esatta, tenendo conto di eventuali errori di arrotondamento, calcolando direttamente la soluzione finale (senza calcolarne approssimazione) in un numero finito di passi. Risultano i più adatti ed efficienti nel caso ci si trova a risolvere sistemi lineari con matrici dense (ovvero con pochi elementi nulli) e di non elevate dimensioni. In pratica spesso si trasforma il sistema lineare in un altro equivalente semplificato nella risoluzione.
Nel caso ci si trovi ad affrontare sistemi lineari con matrici di elevate dimensioni e sparse (ovvero con molti elementi nulli) risultano metodi inefficaci e troppo onerosi in termini di costo computazionale e tempo di calcolo. Per questo tipo di situazioni esistono metodi iterativi, che mirano a costruire la soluzione di un sistema lineare come limite di una successione di vettori che sotto opportune ipotesi convergerà alla soluzione del problema cercata.
Nei metodi iterativi possono comparire, oltre che a degli errori sperimentali come nei metodi diretti, degli errori di troncamento, dato che si va a cercare sostanzialmente un' approssimazione del limite di una successione di punti.
Un'altra differenza importante è che mentre un metodo diretto modifica la matrice di partenza del sistema, spesso alterandone la struttura e aumentando perciò il numero degli elementi non nulli, un metodo iterativo ha il vantaggio di preservare spesso le caratteristiche della matrice iniziale, sfruttandone cosi nei migliore dei modi la sua eventuale sparsità.
I metodi iterativi inoltre vengono molto utilizzati nei casi in cui si voglia perfezionare una soluzione approssimata ottenuta con altri algoritmi, o mediante informazioni a priori sul problema in esame.

3.1. Metodi diretti

Attraverso questi metodi si riesce ad ottenere direttamente (senza approssimazioni intermedie) la soluzione di un sistema lineare in un numero finito di passi (operazioni). Questi metodi identificano l'esatta soluzione *x* del problema purché tutte le operazioni aritmetiche siano eseguite in modo esatto. Questi metodi cercano di trasformare la matrice dei coefficienti *A* di partenza formando così un sistema equivalente a quello iniziale, di più facile risoluzione.

I metodi diretti che si andranno a vedere in questo paragrafo e più noti, sono i seguenti (Robbiano, 2006 [12]):

- Metodo di eliminazione di Gauss
- Fattorizzazione LU

3.1.1. Metodo di eliminazione di Gauss

Il metodo diretto più usato, semplice e conosciuto da chiunque si trovi a risolvere un sistema di equazioni lineari è sicuramente quello delle eliminazioni successive di Gauss.
Questo metodo consiste nel trasformare il sistema lineare iniziale $Ax = b$ in un sistema equivalente di forma triangolare superiore o inferiore.
Un sistema di equazioni è in forma triangolare se la matrice *A* dei coefficienti è triangolare inferiore o superiore. Perciò un sistema triangolare di equazioni lineari avrà la seguente forma:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1,n-1}x_{n-1} + a_{1n}x_n = b_1 \\ a_{22}x_2 + \cdots + a_{2,n-1}x_{n-1} + a_{2n}x_n = b_2 \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ a_{n-1,n-1}x_{n-1} + a_{n-1,n}x_n = b_{n-1} \\ a_{n,n}x_n = b_n \end{cases} \qquad (2.1.)$$

ove supponiamo per semplicità $a_{11} \neq 0, a_{22} \neq 0, \dots, a_{nn} \neq 0$.

Alla base di questo metodo vi è il presupposto secondo il quale risolvere un sistema che abbia forma triangolare superiore o inferiore sia molto semplice e immediato attraverso l'algoritmo di sostituzione all'indietro.

Infatti il suddetto sistema ha una soluzione semplice, che si può ottenere con il seguente procedimento di sostituzione a posteriori. Si risolve per prima l'ultima equazione rispetto all'ultima incognita x_n:

$$x_n = \frac{b_n}{a_{nn}}$$

In secondo luogo si sostituisce questo valore all'x_n della penultima equazione, risolvendola rispetto alla penultima incognita x_{n-1}:

$$x_{n-1} = \frac{b_n - a_{n-1,n}(b_n/a_{nn})}{a_{n-1,n-1}}$$

In terzo luogo si sostituiscono questi valori a x_n e a x_{n-1} nella terzultima equazione, risolvendola rispetto alla terzultima incognita x_{n-2}:

$$x_{n-2} = \frac{b_{n-2} - \left(a_{n-2,n-1}/a_{n-1,n-1}\right)\left[b_{n-1} - a_{n-1,n}(b_n/a_{nn})\right] - \left(a_{n-2,n}/a_{nn}\right)b_n}{a_{n-2,n-2}}$$

In generale, x_k si calcola sostituendo i valori $x_n, x_{n-1}, \dots, x_{k+1}$ nell'equazione *k*-ma:

$$x_k = \frac{\left(b_k - \sum_{m=k+1}^{n} a_{km}x_m\right)}{a_{kk}} \qquad (2.2.)$$

Il procedimento si fermerà quando si troverà la prima incognita x_1. La soluzione trovata è unica perché a ogni passaggio dell'algoritmo il valore di x_k è unicamente determinato.

Il metodo di Gauss è un algoritmo che in un numero finito di passi sostanzialmente trasforma il sistema dato in uno equivalente e della forma indicata precedentemente.
In generale se ci si trova con un sistema di ordine *n*, questo particolare metodo in *n* - 1 stadi riesce a mutarlo in un sistema lineare $\tilde{A}x = b$, cui la matrice dei coefficienti $\tilde{A}$ è una matrice triangolare superiore (ossia tale che tutti gli elementi sotto la diagonale principale sono nulli). Questa trasformazione avviene attraverso operazioni consentite, dette anche *mosse di Gauss*:

1) Scambiare tra loro due equazioni;
2) Moltiplicare un'equazione per una costante diversa da zero;
3) Sommare un'equazione ad un'altra equazione.

Per comodità viene presa in considerazione la matrice associata al sistema:

$$(A|b) = \left[\begin{array}{ccccc|c} a_{11} & a_{12} & a_{13} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} & b_2 \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} & b_3 \\ a_{41} & a_{42} & a_{43} & \dots & a_{4n} & b_4 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} & b_m \end{array}\right] \qquad (2.3.)$$

Questa procedura inizia considerando l'elemento a_{11}; si possono verificare uno dei due seguenti:

- se $a_{11} = 0$ allora il metodo non può essere applicato direttamente; in questo caso basterà speso scambiare questa riga con un'altra del sistema in esame;

- se $a_{11} \neq 0$ è possibile eliminare l'incognita x_1 da tutte le righe successive alla prima. Per ottenere ciò basta semplicemente sostituire la *k*-esima riga R_k*;* $k = 2, \dots, n$ con la somma della riga stessa e della prima riga R_1 moltiplicata per $-\frac{a_{11}}{a_{k1}}$; per continuare l'algoritmo basterà riapplicare lo stesso procedimento alle restanti *n-1* equazioni.

In questo modo attraverso le cosiddette mosse di Gauss si trasformerà la matrice associata $(A|b)$ in una matrice triangolare superiore del tipo:

$$\left[\begin{array}{ccccccc|c} a_{11} & a_{12} & a_{13} & \dots & \dots & a_{1n} & b_1 \\ 0 & a_{22} & a_{23} & \dots & \dots & a_{2n} & b_2 \\ 0 & 0 & a_{33} & \dots & \dots & a_{3n} & b_3 \\ 0 & 0 & 0 & a_{44} & \dots & a_{4n} & b_4 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & a_{mn} & b_m \end{array}\right]$$

Alla fine della procedura il nuovo sistema di equazione lineare verrà risolto attraverso l'algoritmo di sostituzione all'indietro.

2.1.2. Fattorizzazione LU

Un altro interessante metodo diretto è una forma di decomposizione di matrici quadrate in *forma LU* o *fattorizzazione LU*. Esso è praticamente una riformulazione del metodo precedente che comporta un vantaggio significativo dal punto di vista computazionale, dato che riesce a risolvere un sistema lineare con meno operazioni algebriche.
Prima di spiegare il procedimento è importante dire che condizione *necessaria e sufficiente* per poter applicare tale metodo è che la matrice *A* dei coefficienti sia invertibile, e ciò equivale a dire che il determinante delle matrici minori principali è diverso da zero.

In pratica questo metodo dato il solito sistema $Ax = b$, dove *A* rappresenta la matrice dei coefficienti e *b* il vettore termine noto, scompone la matrice *A* nel prodotto di due matrici:

$$A = LU \tag{2.4.}$$

con:

- $L \epsilon R^{nxn}$, matrice triangolare inferiore;
- $U \epsilon R^{nxn}$, matrice triangolare superiore.

Le notazioni *L* e *U* derivano dall'inglese *lower* e *upper*.
Si osserva che per trasformare la matrice *A* in una matrice triangolare *U* superiore basta pre-moltiplicare la matrice *A* per alcune matrici E_i, dette *elementari*, corrispondenti a quelle operazioni che si fanno nella prima parte del metodo di Gauss.
E' di facile osservazione che dato che le matrici elementari E_i corrispondono a moltiplicare una riga della matrice *A* per una costante, od a sommare una riga con una precedente moltiplicata per una costante, tali matrici sono triangolari inferiori.
Si ha perciò:

$$\begin{aligned}
&A_1 = E_1 A \\
&A_2 = E_2 A_1 = E_2 E_1 A \\
&A_3 = E_3 A_2 = E_3 E_2 E_1 A \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
&A_{n-1} = E_{n-1} A_{n-2} = E_{n-1} E_{n-2} \ldots E_2 E_1 A \\
&A_n = U = E_n E_{n-1} \ldots E_2 E_1 A
\end{aligned}$$

Alla fine risulta che:

$$A = {E_n}^{-1} {E_{n-1}}^{-1} \ldots E_2^{-1} {E_1}^{-1} U \tag{2.5.}$$

Dato che l'inversa di una matrice triangolare inferiore è una matrice triangolare inferiore e allo stesso tempo il prodotto di due matrici triangolari inferiori è sempre una matrice triangolare inferiore, si conclude che la matrice:

$$L = {E_1}^{-1} E_2^{-1} \dots {E_{n-1}}^{-1} {E_n}^{-1} \tag{2.6.}$$

è una matrice triangolare inferiore con $l_{ii} = 1, i = 1, \dots, n$. Questa fattorizzazione $A = LU$ realizzata mediante l'applicazione del metodo di Gauss alla matrice *A*, prende il nome di fattorizzazione di Doolittle.

Con questa fattorizzazione, la risoluzione del sistema lineare può essere ricondotta alla risoluzione in cascata di due sottosistemi triangolari:

$$\begin{cases} Ly = b \\ Ux = y \end{cases}$$

Infatti si nota che:

$$Ax = LUx = L(Ux) = Ly = b \tag{2.7.}$$

Prima si risolverà il primo sistema, tipo triangolare inferiore, $Ly = b$:

$$\begin{cases} l_{11} y_1 = b_1 \\ l_{21} y_2 + l_{22} y_2 = b_2 \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ l_{n-1,1} y_1 + l_{n-1,2} y_2 + \cdots + l_{n-1,n-1} y_{n-1} = b_{n-1} \\ l_{n,1} y_1 + l_{n,2} y_2 + \cdots + l_{n,n-1} y_{n-1} + l_{n,n} y_n = b_n \end{cases} \tag{2.8.}$$

attraverso il cosiddetto algoritmo di *eliminazione in avanti*:

$$y_1 = \frac{1}{l_{11}} b_1, \tag{2.9.}$$

$$y_i = \frac{1}{l_{11}} \left(b_i - \sum_{j=1}^{i-1} l_{ij} y_j \right) \quad i = 2,3, \dots, n. \tag{2.10.}$$

Una volta determinato $\boldsymbol{y}$, si risolve il sistema di tipo triangolare superiore, $Ux = y$:

$$\begin{cases} u_{11}x_1 + u_{12}x_2 + \cdots + u_{1,n-1}x_{n-1} + u_{1n}x_n = y_1 \\ u_{22}x_2 + \cdots + u_{2,n-1}x_{n-1} + u_{2n}x_n = y_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ u_{n-1,n-1}u_{n-1} + u_{n-1,n}x_n = y_{n-1} \\ u_{n,n}x_n = y_n \end{cases} \tag{2.11.}$$

attraverso l'algoritmo di *eliminazione all'indietro*:

$$x_n = \frac{1}{u_{nn}} y_n, \tag{2.12.}$$

$$x_i = \frac{1}{u_{ii}} \left(y_i - \sum_{j=i+1}^{n} u_{ij}x_j \right) \quad i = n-1, n-2, \ldots, 1. \tag{2.13}$$

Esistono altre fattorizzazioni LU, oltre quella di Doolittle (Lipschutz, 1992 [9]):

- *Fattorizzazione di Court*, in cui la matrice triangolare superiore *U* è caratterizzata dalla proprietà $u_{ii} = 1, i = 1, \ldots, n$;

- *Fattorizzazione di Cholesky*, in cui la matrice triangolare superiore *U* è uguale alla trasposta della matrice triangolare inferiore *L*.

Quest'ultima viene usata, quando nel sistema lineare $Ax = b$ preso in esame, la matrice *A* è:

1) *Simmetrica*, ovvero quando $A^T = A$;

2) *Definita positiva*, ovvero se per ogni valore non nullo del vettore x si ha $x^T Ax > 0$.

In pratica questo metodo pone *A* nella forma $A = LL^T$ in cui:

$$l_{ij} = \frac{1}{l_{jj}}\left(a_{ij} - \sum_{k=1}^{j-1} l_{ik}l_{jk} \right), j = 1,2, \dots, i-1 \qquad (2.14.)$$

$$l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2}, i = 1,2, \dots, n. \qquad (2.15.)$$

Calcolata la matrice,triangolare inferiore L si procede, come nella fattorizzazione precedente, attraverso la risoluzione in cascata dei due sistemi triangolari:

$$\begin{cases} Ly = b \\ L^T x = y \end{cases} \qquad (2.16.)$$

2.2. Metodi iterativi

In questo paragrafo si andranno a descrivere alcuni cosiddetti metodi iterativi, questi per risolvere un sistema lineare $Ax = b$, con $det(A) \neq 0$. Questi costruiscono a partire da un vettore iniziale $x^{(0)}$ una opportuna successione di vettori $\{x^{(k)}\}_{k\geq 0}$ che sotto specifiche condizioni convergerà alla soluzione cercata. Infatti un metodo iterativo si dice che è *globalmente convergente* se per ogni vettore $x^{(0)} \epsilon R^n$ si ha:

$$\lim_{k\to\infty} \left\| x^{(k)} - \bar{x} \right\| = 0 \qquad (2.17.)$$

dove $\bar{x}$ è una soluzione del sistema lineare (Quarteroni, 2008 [11]).

Come si può dedurre non si otterrà, anche in assenza di errori di arrotondamento, mai la soluzione esatta in quanto dopo un determinato tempo e un certo numero di iterazioni sarà necessario arrestare, o meglio troncare, la successione $\{x^{(k)}\}_{k\geq 0}$ ad un specifico valore di *k*.

Quando ci si trova ad affrontare sistemi sparsi e di grandi dimensioni$(n \approx 10^4 \to 10^8)$ i metodi diretti risultano inutilizzabili e troppi onerosi in termini di memorizzazione della matrice. Questo perché algoritmi come quello di Gauss possono creare il fenomeno di aumento del cosiddetto *fill-in*, ovvero l'aumento del numero degli elementi non nulli attraverso l'alterazione della struttura della matrice iniziale, richiedendo così un maggior spazio per memorizzare tutti gli elementi.
I metodi iterativi sono i più adatti ed efficienti in questo tipo di situazioni, in quanto non richiedono una modifica della matrice di partenza del sistema e nemmeno la sua effettiva memorizzazione, ma basterà solo accedere in qualche modo ai suoi elementi.

I metodi che si andranno ad approfondire in questa parte sono definiti *lineari, stazionari del primo ordine* ed assumono la seguente forma:

$$x^{(k+1)} = Bx^{(k)} + f \tag{2.18.}$$

Sono lineari perché lo è la relazione con cui si esprimono, stazionari perché la matrice B non dipende dalla variazione dell'indice di iterazione *k* e del primo ordine poiché il vettore $x^{(k+1)}$ dipende solo da quello precedente $x^{(k)}$.

Una delle tecniche più utilizzate e conosciute per la costruzione di questi algoritmi è il cosiddetto *splitting additivo* (Saad, 1996 [14]). In pratica viene considerata la decomposizione della matrice A del seguente tipo:

$$A = P - N \quad con \det(P) \neq 0 \tag{2.19.}$$

In questo modo si potrà scrivere un nuovo sistema del tutto equivalente a quello di partenza:

$$(P - N)x = b \tag{2.20.}$$

cioè:

$$Px = Nx + b \tag{2.21.}$$

e quindi moltiplicando entrambi in membri per P^{-1}, si otterrà:

$$x = P^{-1}Nx + P^{-1}b \tag{2.22.}$$

L'idea dei metodi iterativi, come già è stato detto, è quello di partire da un vettore $x^{(0)}$, più o meno prossimo alla soluzione del sistema lineare di partenza, e di costruire una successione di vettori $x^{(k)}$ mediante il seguente procedimento iterativo:

$$x^{(k+1)} = P^{-1}Nx^{(k)} + P^{-1}b. \tag{2.23.}$$

Si può osservare alla fine che il problema di determinare la soluzione del sistema lineare $Ax = b$ si può ricondurre nel problema di determinare il vettore *x* punto fisso dell'equazione:

$$x^{(k+1)} = Bx^{(k)} + f$$

indicando con $f = P^{-1}b$ e $B = P^{-1}N = P^{-1}(P - A) = I - P^{-1}A$.
B è definita *matrice di iterazione*; essa determina un particolare metodo e il suo studio è indispensabile per stabilire la convergenza e la rapidità di convergenza del relativo metodo.
Come si può vedere tramite questa strategia dello splitting ci si riconduce alla forma data in precedenza ai metodi iterativi lineari, stazionari del primo ordine.

Un metodo iterativo è detto convergente se, qualunque sia il vettore iniziale $x^{(0)}$, la successione $\{x^{(k)}\}$ converge. Più precisamente la condizione necessaria e sufficiente per cui il procedimento iterativo sia convergente è che:

$$\rho(B) < 1 \qquad (2.24.)$$

ossia il raggio spettrale (corrispondente all'autovalore di modulo massimo) della matrice di iterazione B deve essere minore di 1.

Come già è stato detto, con un metodo iterativo non è possibile (in generale) calcolare la soluzione in un numero finito di operazioni, per questo motivo sarà necessario individuare dei criteri di arresto (Rodriguez, 2003 [13]). Fissata una tolleranza *TOLL* arbitrariamente piccola si utilizzano come criteri di arresto:

- la norma della differenza dei vettori delle soluzioni, tra due iterazioni consecutive è inferiore alla precisione richiesta:

  $$\left\|x^{(k+1)} - x^{(k)}\right\| \leq TOLL$$

  oppure se $x^{(k+1)} \neq 0$,

  $$\frac{\left\|x^{(k+1)} - x^{(k)}\right\|}{\|x^{(k+1)}\|} \leq TOLL$$

- fissato a priori il numero massimo di iterazioni ITMAX:

  $$numero\ delle\ iterazioni \leq ITMAX,$$

- Criterio del residuo, ossia $\left\|Ax^{(k)} - b\right\|$ deve essere più piccolo possibile.

E' buona norma utilizzare tutti e tre i criteri.

I metodi iterativi che si andranno brevemente ad analizzare successivamente sono i seguenti:

- Il metodo di Jacobi;
- Il metodo di Gauss-Seidel;
- Il metodo di rilassamento;
- Il metodo del gradiente coniugato.

I primi tre metodi si basano sulla decomposizione della matrice dei coefficienti *A*, mentre l'ultimo vedremo che afferma che risolvere un sistema lineare con alcune specifiche caratteristiche della matrice A equivale a minimizzare una forma quadratica ad essa relativa.

2.2.1. Metodo di Jacobi

In questo metodo, detto anche metodo degli sostituzioni simultanee, la matrice dei coefficienti *A*, del relativo sistema lineare, viene decomposta nel seguente modo:

$$A = D - E - F \tag{2.25.}$$

dove D è la matrice diagonale estratta da *A*:

$$\begin{bmatrix} a_{11} & 0 & 0 & \dots & \dots & 0 \\ 0 & a_{22} & 0 & \dots & \dots & 0 \\ 0 & 0 & a_{33} & 0 & \dots & 0 \\ 0 & 0 & 0 & a_{44} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \dots & 0 & a_{nn} \end{bmatrix} \tag{2.26.}$$

$-E$ è la matrice triangolare inferiore corrispettiva:

$$\begin{bmatrix} 0 & 0 & 0 & \dots & \dots & 0 \\ a_{21} & 0 & 0 & \dots & \dots & 0 \\ a_{31} & a_{32} & 0 & 0 & \dots & 0 \\ \vdots & \vdots & & 0 & \dots & 0 \\ \vdots & \vdots & & & \ddots & 0 \\ a_{n1} & a_{n2} & \dots & \dots & a_{n-1,n} & 0 \end{bmatrix} \tag{2.27.}$$

mentre $-F$ è la matrice triangolare superiore:

$$\begin{bmatrix} 0 & a_{12} & a_{13} & \dots & \dots & a_{1n} \\ 0 & 0 & a_{23} & \dots & \dots & a_{2n} \\ 0 & 0 & 0 & & & \vdots \\ 0 & 0 & 0 & 0 & & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & a_{n-1,n} \\ 0 & 0 & 0 & \dots & 0 & 0 \end{bmatrix} \tag{2.28.}$$

In questo procedimento iterativo, nella decomposizione di $A = P - N$, significa porre che:

$$P = D \quad e \quad N = E + F. \tag{2.29.}$$

Per poter confermare la non singolarità di P è richiesto che $a_{ii} \neq 0, i = 1, \dots, n$.
Nel caso non si verificasse questa condizione e inoltre A fosse non singolare esiste una permutazione delle righe che fa sì che la diagonale sia priva di elementi nulli.
Con questa decomposizione il procedimento iterativo avrà la forma canonica seguente perciò:

$$x^{(k+1)} = D^{-1}(E + F)x^{(k)} + D^{-1}b \tag{2.30.}$$

In questo caso la matrice di iterazione sarà:

$$B = P^{-1}N = D^{-1}(E + F) = D^{-1}(D - D + E + F) = D^{-1}(D - A) = I - D^{-1}A \tag{2.31.}$$

la quale determina la convergenza del metodo iterativo, come visto precedentemente, se e solo se $\rho(B) < 1$.

In alternativa questo metodo si può scrivere componente per componente nella forma seguente:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j^k \right), i = 1, \dots, n. \tag{2.32.}$$

Questo metodo viene anche detto parallelizzabile in quanto ogni componete $x^{(k)}$ è indipendente dalle altre.

2.2.2. Metodo di Gauss-Seidel

Questo procedimento è una variante del precedente in quanto può essere costruito utilizzando le componenti delle soluzioni già aggiornate. Infatti per determinare $x^{(k+1)}$ al passo $k + 1$ si utilizzano i valori di $x^{(k+1)}$ già calcolati e disponibili.
Per determinare le componenti del vettore $x^{(k)}$ si utilizzano non solo le componenti di $x^{(k-1)}$ ma anche le componenti precedentemente calcolate di $x^{(k)}$ stesso, per questo motivo questo metodo è detto anche metodo delle sostituzioni successive.
Al generico passo k+1 il metodo di Gauss-Seidel calcola le componenti, tenendo conto che $\forall i$ si ha $a_{ii} \neq 0$, nel modo seguente:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij}x_j^k \right), i = 1, \dots, n \tag{2.33.}$$

Dal punto di vista matriciale, considerata la decomposizione di $A = D - E - F$ definita in precedenza nel metodo di Jacobi, si ha che:

$$P = D - E \quad e \quad N = F \quad (A = P - N). \tag{2.34.}$$

Ricordando la forma canonica che assume un generico metodo iterativo, con le opportune sostituzioni in questo metodo si ha la seguente iterazione:

$$x^{(k+1)} = (D - E)^{-1} F x^{(k)} + (D - E)^{-1} b \tag{2.35.}$$

La matrice di iterazione è perciò:

$$B = P^{-1}N = (D - E)^{-1}F = (D - E)^{-1}\big((D - E) - A\big) = I - (D - E)^{-1}A\ ; \tag{2.36.}$$

la quale determina la convergenza se e solo se $\rho(B) < 1$. Inoltre questo metodo risulta convergente se si verifica una delle seguenti condizioni:

1) la matrice *A* del sistema è a diagonale dominante in senso stretto ovvero
$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|\,;$$
questa condizione di convergenza vale anche per il metodo di Jacobi.

2) la matrice *A* è simmetrica ($A^T = A$) e definita positiva ($\forall x \neq 0\ , x^T A x > 0$).

Il metodo di Gauss-Seidel, a differenza di quello di Jacobi, non è detto parallelizzabile in quanto ogni nuova componente dell'iterazione *k*-esima dipende da tutte le nuove componenti della stessa iterazione che sono state già calcolate.
Un'altra differenza tra i due metodi è che se il metodo di Jacobi ha bisogno per essere implementato sostanzialmente di due vettori per il calcolo di una iterazione, uno per ospitare la vecchia iterata e uno per la nuova, mentre il metodo di Gauss-Seidel lavora con un solo vettore, in quanto è possibile costruire l'algoritmo facendo in modo che il nuovo valore calcolato sovrascriva quello precedente. Per questi motivi, molte volte il metodo di Gauss-Seidel dal punto di vista computazionale e di velocità di convergenza è preferito a quello di Jacobi, infatti si ha che $\rho\big(B_J\big) < \rho(B_{GS}) < 1$.
Si possono comunque verificare dei casi in cui:

- il metodo di Jacobi raggiunge la convergenza in modo più veloce rispetto al metodo di Gauss-Seidel;

- il metodo di Jacobi raggiunge la convergenza mentre il metodo di Gauss-Seidel no.

2.2.3. Il metodo del rilassamento

Questo metodo è nato dalla necessità di accelerare la convergenza di un metodo iterativo. Da come si è visto nei precedenti metodi la velocità di convergenza è regolata dal raggio spettrale della relativa matrice di iterazione, in questo metodo viene introdotto perciò un parametro che dipende da questa, che viene chiamato *parametro di rilassamento ω*, per poter accelerare la convergenza del metodo.
L'iterazione assume la seguente forma:

$$x^{(k+1)} = (1 - \omega)x^{(k)} + \omega\tilde{x}^{(k+1)} \qquad (2.37.)$$

dove $\tilde{x}^{(k+1)}$ è il vettore calcolato dal metodo di Jacobi o da quello di Gauss-Seidel (Saad, 1996 [14]).
Nel prima caso il metodo che si ottiene è detto metodo del *rilassamento simultaneo* (definito con l'acronimo *JOR*, *Jacobi over relaxation*), mentre nel secondo caso si parla del metodo del *rilassamento successivo* (definito con l'acronimo *SOR*, *successive over relaxation*).
Questo *parametro di rilassamento ω* deve essere scelto in modo tale che la matrice di iterazione del relativo metodo iterativo abbia minimo raggio spettrale. In particolare nella scelta di questo parametro esistono due teoremi fondamentali:

1) Se la matrice *A* è simmetrica definite positiva allora il metodo di JOR converge se $0 < \omega < 2/\rho(D^{-1}A)$;

2) Se la matrice *A* è simmetrica definita positiva il metodo SOR converge se e solo se $0 < \omega < 2$.

2.2.4. Il metodo del gradiente coniugato

Questo metodo, che verrà spiegato in modo più completo e approfondito nel capitolo successivo, è forse il più importante tra i metodi iterativi per risolvere sistemi lineari del tipo $Ax = b$ sparsi e di dimensioni elevate, con *A* definita positiva.
Questo metodo parte dal fatto che la risoluzione di un sistema lineare equivale ad minimizzare la relativa forma quadratica ad esso associata con l'importante assunto che la matrice dei coefficienti corrispondente *A* sia simmetrica e definita positiva (Shewchuk, 1994 [15])
Come si spiegherà successivamente questo procedimento parte da un punto iniziale per poi scegliere opportune *direzioni di discesa* per la forma quadratica, lungo le quali muoversi per avvicinarsi il più possibile alla soluzione del sistema lineare preso in considerazione.
Questo metodo viene usato soprattutto per la risoluzione approssimata di equazioni differenziali a derivate parziali, per *structural analysis* e *circuit analysis*.

2.3. Costo computazionale

Nella tabella seguente viene riportato il costo computazionale di ogni metodo di risoluzione di un sistema lineare descritto nei paragrafi precedenti. Il costo computazionale è un aspetto molto importante quando si va ad implementare un algoritmo perché esso descrive il numero di operazioni pesanti (moltiplicazioni o divisioni) necessarie per calcolare numericamente la soluzione. La regola vuole che un algoritmo sia più efficiente e veloce se quest'ultimo ha un minor costo computazionale rispetto ad un altro. Si ricorda che nei metodi iterativi il costo computazionale deve essere moltiplicato per il numero di iterazioni necessarie per trovare la soluzione.

Metodi	**Costo computazionale**
Diretti	
- Gauss	$O(n^3/3)$
- Fattorizzazione LU	*Doolittle* $O(n^3)$ *Cholesky* $O(n^3/6)$
Iterativi - Jacobi - Gauss-Seidel - Rilassamento - Gradiente coniugato	se A è *sparsa*: $O(m) \quad m \ll n^2$ se A è *piena*: $O(n^2)$

Tab. 1 Costo computazionale dei vari metodi di risoluzione per un sistema lineare del tipo Ax=b.

Cap.3 – Gradiente coniugato e proprietà

In questo capitolo si andrà a vedere in maniera più approfondita il metodo del gradiente coniugato e le sue relative proprietà. Questo metodo, proposto da Hestenes e Stiefel nel 1950, fa parte della famiglia dei metodi iterativi per la soluzione di sistemi lineari simmetrici, i quali generano, partendo da un vettore iniziale, una successione di vettori convergente alla soluzione del problema dato. Esso è nato come alternativa al metodo di eliminazione di Gauss in quanto risulta molto performante nella risoluzione di sistemi lineari di grandi dimensioni e sparsi, ma può essere anche adattato a risolvere problemi di ottimizzazione non lineare.

3.1. Il metodo del gradiente

Il metodo del gradiente coniugato è sostanzialmente una forma più avanzata del più semplice metodo del gradiente applicato alla minimizzazione di una funzione quadratica. L'unica differenza è soltanto una scelta più accurata delle cosiddette direzioni di discesa, permettendo così di raggiungere la convergenza in un numero di iterazioni molto inferiore alla dimensione del problema in esame.
Per comprendere il procedimento di questo particolare metodo sarà necessario andare a vedere da dove nasce e come funziona appunto il cosiddetto *metodo del gradiente*.

Questo metodo parte dall'ipotesi che risolvere un problema del tipo $Ax = b$,con $b, x \in R^n, c$ costante e l'importante condizione che la matrice dei coefficienti $A \in R^{nxn}$ sia *simmetrica* e *definita positiva*, equivale a minimizzare la forma quadratica ad esso associata:

$$f(x) = \frac{1}{2}x^T Ax - b^T x + c \qquad (3.1.)$$

detta anche *energia del sistema* (Shewchuk, 1994 [15]).
Questo risultato può essere dimostrato. Definito il gradiente (che è il vettore che ha come componenti le derivate parziali) della funzione quadratica nella seguente forma:

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{pmatrix} \qquad (3.2.)$$

la funzione quadratica $f(x)$ raggiunge il suo valore minimo nel punto in cui si annulla, $\nabla f(x) = 0$.

Infatti andando a vedere più in dettaglio, tramite la forma esplicita di $f(x)$:

$$f(x) = \frac{1}{2}x^T Ax - b^T x + c = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} x_i a_{ij} x_j - \sum_{i=1}^{n} b_i x_i + c \qquad (3.3.)$$

E se si deriva rispetto a x_k, si ottiene la seguente derivata parziale:

$$\frac{\partial f(x)}{\partial x_k} = \frac{1}{2}\sum_{j=1}^{n} a_{kj} x_j + \frac{1}{2}\sum_{i=1}^{n} a_{ik} x_i - b_k \qquad (3.4.)$$

da cui segue la forma matriciale:

$$\nabla f(x) = \frac{1}{2}A^T x + \frac{1}{2}Ax - b \qquad (3.5.)$$

ed essendo la matrice *A* simmetrica, ovvero $A^T = A$, si semplificherà in questa forma:

$$\nabla f(x) = Ax - b. \qquad (3.6.)$$

Ponendo come si è detto $\nabla f(x) = 0$ viene confermato che il problema della minimizzazione appena mostrato equivale a trovare la soluzione $\bar{x}$ del sistema lineare di partenza $Ax = b$.
Viceversa se si considera la relazione tra f calcolata in $\bar{x}$, soluzione del sistema, e in un punto arbitrario *y*, allora si ha:

$$f(y) = f(\bar{x} + y - \bar{x}) = f(\bar{x}) + \tfrac{1}{2}(y - \bar{x})^T A(y - \bar{x}) \qquad (3.7.)$$

dato che l'ultimo termine è positivo se la matrice A è definita positiva, ossia $x^T Ax > 0$, per ogni $y \neq x$ si conclude che:

$$f(y) > f(\bar{x}) \qquad (3.8.)$$

ed $\bar{x}$ è *punto di minimo* per la funzione *f*.

Questo particolare risultato può essere dimostrato anche dal punto di vista grafico.
Come si è visto nel primo capitolo risolvere un sistema di equazioni lineari equivale a trovare un punto di intersezione di queste equazioni nel piano cartesiano.
La forma quadratica di un sistema con $A \succ 0$, dal punto di vista grafico, corrisponde ad un *paraboloide* la cui forma è determinata dall'importante condizione che la matrice dei coefficienti sia definita positiva. I vari casi che si possono verificare sono rappresentati in **Fig.1 (**Shewchuk, 1994 [15]).

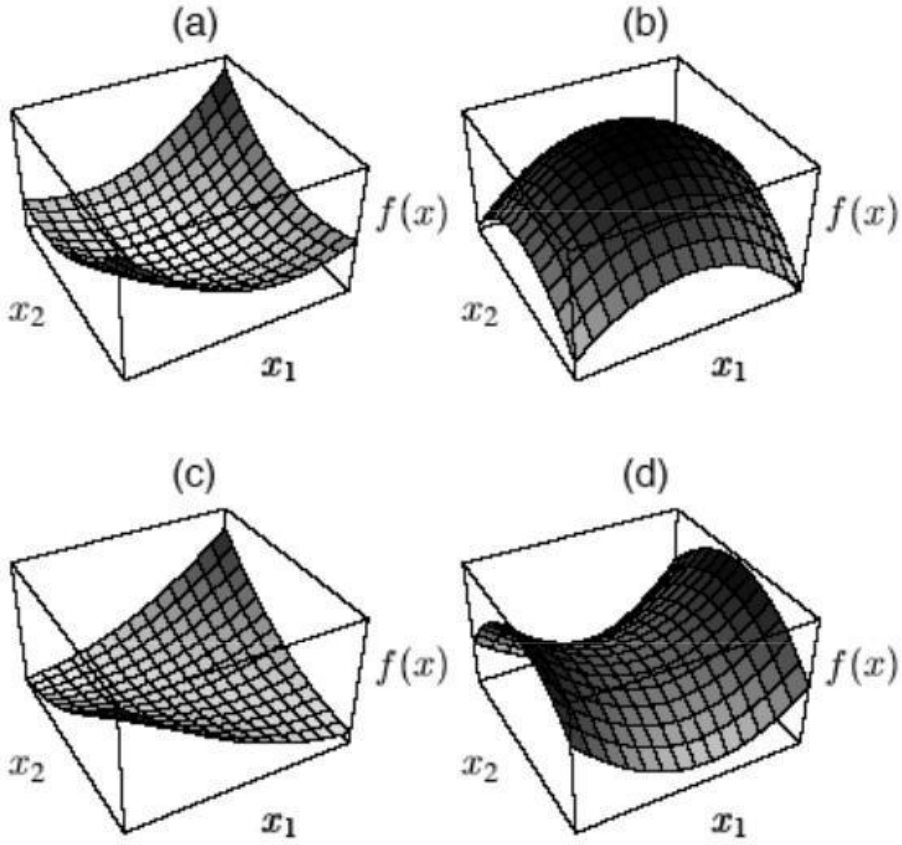


Fig. 1 Grafici della forma quadratica f(x). (a)Forma quadratica definita positiva; (b)Forma quadratica definita negativa; (c)Forma quadratica semidefinita positiva; (d)Forma quadratica non definita.

Se la matrice *A*, della forma quadratica $f(x)$, è *definita positiva* allora il minimo corrisponde alla soluzione del sistema $Ax = b$; mentre se *A* è *definita negativa*, in questo caso il paraboloide è rovesciato ed il suo massimo corrisponde alla soluzione del sistema; se invece *A* è *semidefinita positiva* allora vuol dire che almeno un suo autovalore è nullo e sarà perciò singolare, le soluzioni del sistema sono più di una e corrispondono all'insieme dei punti della linea sul fondo del paraboloide; infine se A è *non definita*, la soluzione è un cosiddetto punto di sella.

Gli altri parametri, presenti della forma quadratica *b* e *c* , definiscono dove si posiziona il punto di minimo del paraboloide, senza influirne sulla forma.

Dato che si è arrivati alla conclusione che risolvere un sistema lineare con matrice dei coefficienti simmetrica e definita positiva equivale a calcolare il minimo della sua relativa funzione quadratica, la difficoltà adesso è determinare con quali metodi risolvere questo problema di minimizzazione. Tra i metodi usati ci sono i cosiddetti *metodi di discesa*. Questi metodi per trovare il minimo della forma quadratica partono da un punto iniziale $x^{(0)} \epsilon R^n$ e successivamente scelgono opportune direzioni lungo le quali muoversi per avvicinarsi, il più velocemente possibile, alla soluzione $\bar{x}$.

Sono metodi iterativi non stazionari, in quanto i parametri che li caratterizzano variano ad ogni iterazione, infatti alla generica iterazione *k* si determina $x^{(k+1)}$ nella seguente forma:

$$x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)} \tag{3.9.}$$

in cui α_k è la lunghezza del passo lungo la direzione di ricerca $d^{(k)}$. La scelta che viene fatta di questi valori determina il metodo di discesa da utilizzare per minimizzare la forma quadratica relativa al sistema lineare da risolvere.
Nel metodo del gradiente viene scelta come direzione di ricerca quella di massima discesa (*steepest descent*):

$$d^{(k)} = -\nabla f\left(x^{(k)}\right) \tag{3.10.}$$

ovvero quella opposta alla direzione del gradiente della funzione quadratica f, detta anche *antigradiente* in $x^{(k)}$. Questa scelta viene motivata se si analizza che la direzione dell'antigradiente (normalizzata) è la direzione che minimizza la derivata direzionale in $x^{(k)}$ tra tutte le direzioni che hanno norma euclidea unitaria (Grippo e Sciandrone, 2011 [8]). In pratica è la soluzione del problema:

$$\min_{\|d\|_2=1} \nabla f\left(x^{(k)}\right)^T d \ , \tag{3.11.}$$

e per la *disuguaglianza di Schwarz*, si può scrivere:

$$\left|\nabla f\left(x^{(k)}\right)^T d\right| \leq \|d\|_2 \left\|\nabla f\left(x^{(k)}\right)\right\|_2 \ , \tag{3.12.}$$

in cui il segno di uguaglianza vale se e solo se $d = \lambda \nabla f\left(x^{(k)}\right)$ con $\lambda \epsilon R$. La soluzione a (3.11.) sarà data perciò da:

$$d^{(k)} = -\nabla f\left(x^{(k)}\right) / \left\|\nabla f\left(x^{(k)}\right)\right\|_2 \ , \tag{3.13.}$$

che definisce appunto la direzione dell'antigradiente in $x^{(k)}$. Per tale ragione si parla anche di *metodo della discesa più ripida*.

Si può osservare in questo particolare metodo che la direzione di decrescita di $f(x)$ coincide con il residuo che indica quanto siamo distanti dal corretto valore di *b* al passo *k*:

$$d^{(k)} = -\nabla f\left(x^{(k)}\right) = -\left(Ax^{(k)} - b\right) = r^{(k)}. \tag{3.14.}$$

Adesso, calcolata la direzione in cui muoversi si deve determinare con quale passo, ossia α_k. Questo particolare passo deve essere scelto, attraverso una *line search*, in modo tale che il valore della funzione quadratica $f(x^{(k)} + \alpha_k d^{(k)})$ sia minimo lungo la direzione scelta, per far sì che questo avvenga si deve imporre l'annullamento della

sua derivata rispetto al valore α, ossia $\frac{\partial f(x^{(k)}+\alpha_k d^{(k)})}{\partial \alpha} = 0$, in questo modo si ottiene il valore del passo con cui muoversi lungo la direzione scelta:

$$\alpha_k = \frac{\left(d^{(k)}\right)^T r^{(k)}}{(d^{(k)})^T A d^{(k)}} \,. \tag{3.15.}$$

Si fa notare che nel *metodo del gradiente*, sapendo che $d^{(k)} = r^{(k)}$, l'ampiezza del passo è:

$$\alpha_k = \frac{\left(r^{(k)}\right)^T r^{(k)}}{(r^{(k)})^T A r^{(k)}} \tag{3.16.}$$

e si conclude, dopo la scelta di questi due valori, che l'iterazione assume la seguente forma:

$$x^{(k+1)} = x^{(k)} + \alpha_k r^{(k)} \;. \tag{3.17.}$$

Questo algoritmo dovrà essere iterato fino a convergenza. Esiste un teorema che valuta con che velocità questo metodo converge alla soluzione cercata (Saad, 1996 [14]); esso afferma che se, in un generico sistema lineare $Ax = b$, la matrice A è simmetrica e definita positiva, il metodo del gradiente converge per ogni $x^{(0)}$ alla soluzione $\bar{x}$ e si ha:

$$\left\|x^{(k+1)} - \bar{x}\right\|_A \leq \left(\frac{\lambda_M - \lambda_m}{\lambda_M + \lambda_m}\right) \left\|x^{(k)} - \bar{x}\right\|_A \tag{3.18.}$$

dove per definizione $\|x\|_A = (x^T A x)^{1/2}$ e λ_M è l'autovalore massimo e λ_m è l'autovalore minimo della matrice *A*.
La rapidità con cui questo metodo converge alla soluzione a $\bar{x}$ dipende da questi due specifici autovalori. Inoltre, tanto più λ_M e λ_m sono vicini (ossia A è *ben condizionata*) tanto più la velocità di convergenza sarà veloce.

Come funziona dal punto di vista grafico il metodo del gradiente è mostrato in **Fig.2**. Data la funzione quadratica $f(x) = \frac{1}{2}x^T A x - b^T x + c$ nel piano cartesiano (cioè in due dimensioni) vengono mostrate le sue cosiddette curve di livello, ovvero l'insieme dei punti in cui la funzione è uguale ad una costante, attraverso ellissi il cui centro coincide con il minimo della funzione quadratica e rappresenta la soluzione $\bar{x}$ del sistema lineare in esame; dal punto di vista tridimensionale è come se andassimo a "tagliare" orizzontalmente a fette il paraboloide e lo guardassimo dall'alto.

L'algoritmo del gradiente inizia dal punto $x^{(0)}$ e sceglie come direzione di ricerca $d^{(0)}$, che in questo caso è l'antigradiente della funzione. Muovendosi lungo la direzione $d^{(0)}$(e proiettata quest'ultima sul piano delle variabili) va a cercare l'elisse tangente a questa e calcola il secondo punto come $x^{(1)} = x^{(0)} + \alpha_0 d^{(0)}$, poi da questo secondo punto $x^{(1)}$segue lo stesso procedimento fino ad avvicinarsi il più possibile alla soluzione *u*. Il percorso a "*zigzag*" è dovuto all'ortogonalità tra una direzione e la sua precedente.

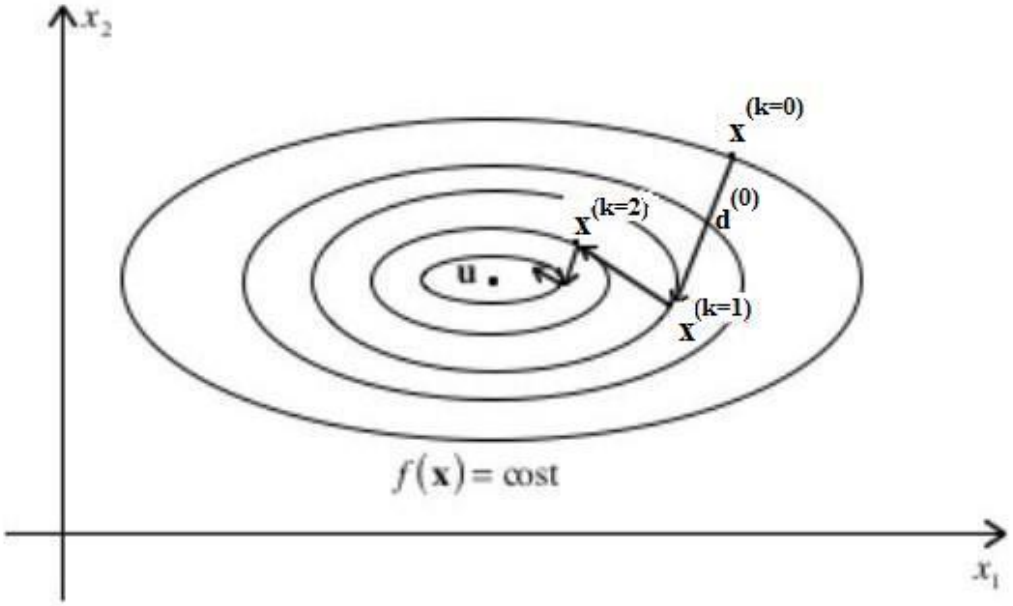


Fig. 2 Grafico relativo alle iterazioni del metodo del gradiente.

3.2. Il metodo del gradiente coniugato

Questo metodo è una forma di perfezionamento del precedente e fa parte sempre dei metodi di discesa per la funzione $f(x)$. Come già è stato detto precedentemente questo particolare metodo parte dal presupposto che risolvere un sistema lineare $Ax = b$, in cui A è simmetrica e definita positiva, equivale a minimizzare la relativa forma quadratica; la differenza con il metodo del gradiente e che in questo caso viene fatta una scelta più accurata delle cosiddette direzioni di discesa (Rodriguez, 2003 [13]).
Nel metodo del gradiente coniugato si parte, tenendo conto che si ha che fare sempre con la forma quadratica $f(x)$, dal presupposto che un vettore $x^{(k)}$ risulta ottimale rispetto ad una determinata direzione d se è verificata la seguente condizione:

$$f\left(x^{(k)}\right) \leq f\left(x^{(k)} + \lambda d\right) \quad , \forall\, \lambda \in R; \tag{3.19.}$$

inoltre il vettore $x^{(k)}$ si dice ottimale rispetto alla direzione d se e solo se questa è ortogonale al residuo $r^{(k)}$, ossia:

$$d^T r^{(k+1)} = 0. \tag{3.20.}$$

Questa collegamento tra ottimalità e ortogonalità può essere facilmente dimostrato; per prima cosa facendo opportune sostituzioni nella funzione quadratica $f(x)$, si ottiene la seguente forma equivalente:

$$f\left(x^{(k)} + \lambda d\right) = \frac{1}{2}\left(x^{(k)} + \lambda d\right)^T A(x^{(k)} + \lambda d) - b^T(x^{(k)} + \lambda d)\,, \tag{3.21.}$$

dopodiché si annulla la sua derivata rispetto a λ:

$$\frac{\partial f\left(x^{(k)} + \lambda d\right)}{\partial \lambda} = d^T A d\lambda - d^T r^{(k)} = 0 \tag{3.22.}$$

dato che $x^{(k)}$ è ottimale rispetto alla direzione d, la funzione $f\left(x^{(k)}\right)$ deve avere un minimo per $\lambda = 0$. Dall'altro lato se si impone $d^T r^{(k+1)} = 0$ si ha un minimo per $\lambda = 0$, confermando così il risultato enunciato prima.
Uno dei più grandi problemi del metodo del gradiente semplice è che questa condizione di ottimalità del vettore rispetto ad una certa direzione è confermata solamente al primo passo dell'algoritmo; per questo motivo viene in soccorso il metodo del gradiente coniugato, il quale riesce a far ereditare questa condizione di ottimalità del vettore *k*-esimo rispetto ad una certa direzione da tutti gli elementi della successione successivi al primo (Golub & Van Loan, 1996 [6]).

Supponendo che $x^{(k)}$ sia ottimale rispetto alla direzione *d*, e quindi $d^T r^{(k+1)} = 0$, si ponendo al passo successivo:

$$x^{(k+1)} = x^{(k)} + \tilde{d} \tag{3.23.}$$

per far sì che anche $x^{(k+1)}$ sia ottimale rispetto alla direzione *d* è necessario che:

$$0 = d^T r^{(k+1)} = d^T \left(r^{(k)} - A\tilde{d}\right) = -d^T A\tilde{d}\,, \tag{3.24.}$$

ovvero le direzioni d e $\tilde{d}$ devono essere *A-ortogonali* o *A-coniugate*.
Una delle proprietà fondamentali di questo algoritmo è la capacità appunto di generare un insieme di vettori cosiddetti coniugati (Nocedal & Wright, 1999 [10]).
Un insieme di vettori non nulli $\{v_0, v_1, \dots, v_n\}$ sono detti *coniugati* rispetto ad una matrice *A* simmetrica e definita positiva se:

$$v_i{}^T A v_j = 0\,, \quad \forall\, i \neq j. \tag{3.25.}$$

E' facile dimostrare che i vettori che possiedono questa particolare proprietà sono anche linearmente indipendenti.

Nello specifico, il vettore delle direzioni $d^{(k)}$ viene scelto in modo che al primo passo la direzione sia la stessa del metodo del gradiente, ovvero l'antigradiente (che come si è visto coincide con il residuo) e nei successivi passi viene scelta come segue:

$$d^{(k)} = \begin{cases} r^{(0)} & se\ k = 0 \\ r^{(k)} + \beta_k d^{(k-1)} & se\ k \geq 1 \end{cases}. \tag{3.26.}$$

Come si può notare il metodo del gradiente coniugato nella scelta della direzione di ricerca tiene conto delle direzioni precedentemente considerate.
Questo può avvenire solo se le direzioni godono della proprietà della coniugatezza, ovvero:

$$(d^{(k)})^T A d^{(k-1)} = 0\,, \tag{3.27.}$$

che si traduce nella condizione che:

$$\beta_{k+1} = -\frac{\left(d^{(k)}\right)^T r^{(k+1)}}{(d^{(k)})^T A d^{(k)}}\,. \tag{3.28.}$$

Dopo queste considerazioni si può scrivere il metodo del gradiente coniugato nel seguente modo:

$$x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)}\,, \qquad k \geq 0 \tag{3.29.}$$

con α_k calcolato come nel metodo del gradiente:

$$\alpha_k = \frac{\left(r^{(k)}\right)^T r^{(k)}}{(d^{(k)})^T A d^{(k)}}\ . \tag{3.30.}$$

Dato che $x^{(k+1)}$ è "ottimale" rispetto alle direzioni $d^{(i)}$ con $i = 1, \dots, k-1$ non si può far diminuire ulteriormente la funzione quadratica $f(x)$, ciò comprova che questo algoritmo converge alla soluzione esatta del sistema lineare in al più *n* iterazioni.

Oltre alla coniugatezza delle direzioni scelte vi sono altre due proprietà importanti in questo metodo:

1) i residui $r^{(k)}$ sono reciprocamente ortogonali;

2) ogni direzione di ricerca $d^{(k)}$ e ogni residuo $r^{(k)}$ è contenuto nel *sottospazio di Krylov* di ordine *k* del residuo $r^{(0)}$, definito nel seguente modo:

$$\mathcal{K}\left(r^{(0)}; k\right) \stackrel{\text{def}}{=} \text{span}\left\{r^{(0)}, Ar^{(0)}, \dots, A^k r^{(0)}\right\}. \tag{3.31.}$$

In pratica il metodo del gradiente coniugato minimizza la funzione quadratica,ovvero trova la soluzione corrente migliore del sistema lineare simmetrico su questo sottospazio di Krylov di dimensione crescente.
La soluzione più vicina alla soluzione finale sul sottospazio di Krylov $\mathcal{K}\left(r^{(0)}; k\right)$ è la migliore soluzione che il metodo riesce a trovare sul sottospazio spannato ottenuto dalle direzioni di ricerca che ha trovato (Conn, Gould & Toint, 2000 [3]).

Questo algoritmo termina dopo al più n iterazioni, ma vi possono nascere dei problemi dovuti a quanto sia grande questo *n*, perché in alcune situazioni ci si può trovare con matrici dei coefficienti $A \in R^{nxn}$ di elevate dimensioni. Per questo motivi esiste un teorema che valuta con che velocità questo metodo converge alla soluzione cercata (Nocedal & Wright, 1999 [10]). L'analisi della sua convergenza è importante per dimostrare come questo metodo sia molto utile per problemi che altri algoritmi non riuscirebbero a risolvere.

Questo teorema è basato sul numero di *condizionamento* di *A*:

$$\mathcal{K}(\mathrm{A}) = \|A\|_2 \|A^{-1}\|_2 = \lambda_M / \lambda_m \,, \tag{3.32.}$$

in cui λ_M è l'autovalore massimo mentre λ_m è l'autovalore minimo della matrice *A*. Esso afferma che se, in un generico sistema lineare $Ax = b$, la matrice A è simmetrica e definita positiva, il metodo del gradiente coniugato converge per ogni $x^{(0)}$ alla soluzione $\bar{x}$ e si ha:

$$\left\|x^{(k)} - \bar{x}\right\|_A \leq 2\left(\frac{\sqrt{\mathcal{K}(\mathrm{A})} - 1}{\sqrt{\mathcal{K}(\mathrm{A})} + 1}\right)^k \left\|x^{(0)} - \bar{x}\right\|_A \tag{3.33.}$$

Spesso questo limite dà una grande sovrastima dell'errore, ma può essere utile nei casi in cui le uniche informazioni che abbiamo sono le stime degli autovalori λ_i estremi della matrice dei coefficienti *A*; questo limite deve essere confrontato con quello di rapida discesa e più la matrice A ha un numero di condizionamento $\mathcal{K}(\mathrm{A}) \approx 1$ più il metodo del gradiente coniugato convergerà rapidamente alla soluzione cercata.

Come si vedrà nel seguente capitolo esiste una tecnica per accelerare la convergenza di questo metodo, questa è la cosiddetta tecnica del *precondizionamento* con la quale si riesce a trasformare un sistema lineare simmetrico definito positivo $Ax = b$ in un altro equivalente $A'x' = b'$ dalla cui soluzione x' è possibile ricavare la soluzione $\bar{x}$ del primo (Saad, 1996 [14]).

Cap.4 – Precondizionatori

4.1. Introduzione

Come già è stato detto nei capitoli precedenti la soluzione efficiente di sistemi di equazioni lineari è centrale per problemi di qualunque natura: scientifica, economica, aziendale e ingegneristica. Risolvere questi problemi può essere molto oneroso in termini di tempo e complessità computazionale.
I metodi diretti risultano più adatti ed efficienti nel caso di sistemi lineari con matrici dense e di non elevate dimensioni, grazie alla loro robustezza e affidabilità, mentre se si ha a che fare con problemi di elevate dimensioni sono preferibili i metodi iterativi, soprattutto quelli basati sui cosiddetti sottospazi di Krylov come il metodo del gradiente coniugato. Questi metodi di Krylov sono stati per lo più sviluppati tra gli anni '70 e '90, ma il loro interesse è in continua crescita in molte aree dell'ingegneria e del calcolo scientifico. Per diventare competitivi con i metodi diretti, i metodi iterativi basati sui sottospazi di Krylov hanno spesso bisogno di un apposito *precondizionamento* per poter convergere più velocemente ad una soluzione del sistema lineare in esame. Dato un sistema lineare simmetrico e definito positivo $Ax = b$, precondizionare tale sistema lineare equivale a voler condizionare la matrice dei coefficienti A prima di applicare qualsiasi metodo iterativo; per *condizionare* si intende migliorare il numero di condizionamento di questa matrice.
Il *numero di condizionamento* di una matrice simmetrica A è definito come segue:

$$\mathcal{K}(\mathrm{A}) = \|A\|_2 \|A^{-1}\|_2 = \lambda_M / \lambda_m \,, \tag{4.1.}$$

in cui λ_M è l'autovalore massimo mentre λ_m è l'autovalore minimo della matrice A. Una matrice per essere "ben condizionata" deve avere un numero di condizionamento $\mathcal{K}(\mathrm{A}) \approx 1$. La tecnica del precondizionamento serve per migliorare questo particolare numero e perciò la robustezza e l'efficienza computazionale dei metodi iterativi usati per risolvere il sistema lineare $Ax = b$. Come si vedrà nel corrente capitolo questa tecnica per accelerare la convergenza di un metodo iterativo trasforma il sistema lineare simmetrico definito positivo:

$$Ax = b \tag{4.2.}$$

in un altro equivalente $A'x' = b'$, dalla cui soluzione x' è possibile ricavare la soluzione $\bar{x}$ del primo (Saad, 1996 [14]).

In pratica il precondizionamento consiste nel prendere una matrice $M \epsilon R^{nxn}$, chiamata *matrice di precondizionamento* o *precondizionatore*, con le seguenti caratteristiche :

- non singolare, ovvero invertibile;
- simmetrica $M^T = M$;
- definita positiva $M \succ 0$;

e trasformare il sistema precedente nell'equivalente sistema precondizionato:

$$M^{-1}Ax = M^{-1}b \tag{4.3.}$$

che si può risolvere iterativamente più velocemente rispetto a quello iniziale, purché si abbia che $\mathcal{K}(M^{-1}\mathrm{A}) \ll \mathcal{K}(\mathrm{A})$. Dato un sistema $Ax = b$, sapendo che la matrice simmetrica A ha per definizione autovalori $\lambda \epsilon R$ (ponendo $Av = \lambda v$ dove λ sono gli autovalori e v sono gli autovettori di A), si può dimostrare che nonostante il precondizionamento del sistema, con il precondizionatore $M \succ 0$, gli autovalori di $M^{-1}\mathrm{A}$ rimangono sempre autovalori reali nonostante la matrice $M^{-1}\mathrm{A}$ non sia sempre simmetrica. Ciò può essere provato da semplici passaggi matematici. Se si indica con υ gli autovettori e μ gli autovalori di $M^{-1}\mathrm{A}$, si ha che:

$$M^{-1}\mathrm{A}\upsilon = \mu\upsilon, \tag{4.4.}$$

andando a moltiplicare a sinistra entrambi i membri dell'equazione per $M^{1/2}$e dopo alcuni semplici passaggi si ottiene che:

$$M^{1/2}M^{-1}\mathrm{A}\upsilon = \mu M^{1/2}\upsilon$$
$$M^{-1/2}A\upsilon = \mu M^{1/2}\upsilon$$
$$M^{-1/2}AM^{-1/2}(M^{1/2}\upsilon) = \mu(M^{1/2}\upsilon)$$

essendo $M^{-1/2}AM^{-1/2}$ simmetrica allora i suoi autovalori μ soddisfano la condizione $\mu \epsilon R$, confermando così questa proprietà degli autovalori di $M^{-1}\mathrm{A}$. Un'altra importante proprietà degli autovalori di $M^{-1}\mathrm{A}$ è che questi risultano meglio clusterizzati, ovvero più vicini tra loro,rispetto a quelli di A. Da ciò si deduce che il miglior precondizionatore possibile è quando $M^{-1} = A^{-1}$, perché in questo caso il *numero di condizionamento* è dato da:

$$\mathcal{K}(M^{-1}\mathrm{A}) = 1, \tag{4.5.}$$

ovvero gli autovalori di $M^{-1}\mathrm{A}$ sono tutti coincidenti e pari ad 1. Inoltre si nota che la forma quadratica $\frac{1}{2}x^T\frac{(M^{-1\circ}+AM^{-1})}{2}x + (M^{-1}b)^Tx$ è esattamente una sfera nel caso $M^{-1} = A^{-1}$ (si nota infatti che $M^{-1\circ} = A^{-1}A = I$), quindi la soluzione di (4.3.) viene trovata con soltanto un'unica iterazione. Questo perché come è stato spiegato nel

capitolo precedente la funzione quadratica $f(x) = \frac{1}{2}x^T Ax - b^T x + c$ nel piano cartesiano viene rappresentata dalle sue cosiddette curve di livello attraverso ellissoidi il cui centro coincide con il minimo della funzione quadratica e rappresenta la soluzione $\bar{x}$ del sistema lineare in esame; partendo da un punto $x^{(0)}$ l'algoritmo del gradiente sceglie come direzione di ricerca $d^{(0)}$, che coincide con l'antigradiente della funzione. Muovendosi lungo questa direzione $d^{(0)}$ va a cercare l'ellissoide tangente a questa e calcola il secondo punto come $x^{(1)} = x^{(0)} + \alpha_0 d^{(0)}$, poi da questo secondo punto $x^{(1)}$segue lo stesso procedimento fino ad avvicinarsi il più possibile alla soluzione $\bar{x}$. Dato che in questo caso gli ellissoidi saranno delle sfere vere e proprie basterà una sola iterazione per trovare la soluzione $\bar{x}$ al sistema in esame.

Quando un sistema lineare $Ax = b$ viene trasformato nel sistema precondizionato $M^{-1}Ax = M^{-1}b$ si parla di *precondizionamento a sinistra ma* si può utilizzare anche il *precondizionamento a destra*, ossia AM^{-1}. La scelta tra quale precondizionamento utilizzare spesso dipende dalle caratteristiche della matrice dei coefficienti A e dal metodo iterativo applicato. Le matrici $M^{-1\circ}$ e AM^{-1} non sono in generale simmetriche nonostante singolarmente M ed A lo siano. Per tale motivo esiste una strategia per preservare questa caratteristica di simmetria della matrice di partenza (Shewchuk, 1994 [15]). Questa strategia parte dal fatto che data una matrice M simmetrica definita positiva, esiste la matrice L tale che $LL^T = M$, calcolata attraverso la fattorizzazione di Cholesky. In questo modo il sistema lineare $Ax = b$ viene trasformato nel nuovo sistema simmetrico e definito positivo:

$$L^{-1}AL^{-T}\hat{x} = L^{-1}b \text{ con } \hat{x} = L^T x \qquad (4.6.)$$

che si risolve prima per $\hat{x}$ e poi x.

Si può notare che le matrici $M^{-1}A$ e $L^{-1}AL^{-T}$ hanno gli stessi autovalori, questo perché se v è un autovettore di $M^{-1\circ}$ con autovalore λ, allora $L^T v$ è un autovettore di $L^{-1}AL^{-T}$ con autovalore λ:

$$(L^{-1}AL^{-T})L^T v = (L^T L^{-T})L^{-1}Av = L^T \mathrm{M}^{-1}Av = \lambda L^T v. \qquad (4.7.)$$

In generale, vi sono tre requisiti fondamentali per un buon precondizionatore (Benzi, 2002 [1]):

1) gli autovalori della matrice precondizionata M^{-1}A devono essere meglio raggruppati della matrice dei coefficienti di partenza A, in modo che siano disposti nell'asse reale il più vicino ad 1; infatti, il precondizionamento cerca di "allungare" la forma quadratica per farla apparire più simile ad una sfera in modo tale che gli autovalori siano il più vicini tra loro.

2) il sistema precondizionato deve essere più facile da risolvere di quello di partenza;
3) il precondizionatore deve essere economico dal punto di vista computazionale e applicativo.

Il primo requisito è stato spiegato precedentemente, con il secondo si intende che l'iterazione "precondizionata" del metodo usato per risolvere il sistema lineare in esame deve convergere il più rapidamente possibile alla soluzione; mentre con la terza si richiede che l'iterazione non sia troppo costosa dal punto di vista computazionale. Si può notare che questi due ultimi requisiti sono spesso in concorrenza tra loro ed è per questo motivo è necessario nella costruzione di un buon precondizionatore trovare un equilibrio tra queste due esigenze. Con un buon precondizionatore il tempo di calcolo dell'iterazioni "precondizionata" deve essere significativamente inferiore a rispetto quello della normale iterazione non precondizionata.

In molti casi infatti ci si può trovare ad affrontare una sequenza di k sistemi lineari del tipo $Ax = b$ con stessa (o quasi) matrice dei coefficienti A e diverso vettore dei termini noti b. Allora in questo caso se si volesse calcolare un precondizionatore per ogni sistema lineare si avrebbe k volte il tempo di calcolo del precondizionatore. Per evitare questo, come si vedrà successivamente, nell'articolo di Gratton, Sartenaer e Tshimanga [7], vi è una classe di precondizionatori che risolve questo tipo di problema. Questo accade spesso nella risoluzione di problemi non lineari quando vengono applicati metodi di tipo Newton.

In generale esistono due approcci nella costruzione di un precondizionatore (Benzi, 2002 [1]):

1) *approccio problem-specific*: è uno dei più popolari e consiste nel progettare algoritmi specifici che siano efficaci per una classe ristretta di problemi. Questo approccio può essere molto efficace ma può richiedere la conoscenza completa del problema a portata di mano.

2) *approccio con tecniche di precondizionamento*: dato che il primo approccio non può essere sempre fattibile data la difficoltà nel trovare e utilizzare informazioni riguardanti il problema in esame, sono nate delle tecniche di precondizionamento universalmente applicabili. Queste sono puramente tecniche algebriche che utilizzano solo le informazioni contenute nella matrice dei coefficienti A. Inoltre sono molto facili da utilizzare e sviluppare.

4.2. I diversi precondizionatori

Esistono in letteratura diversi tipi di precondizionatore, i più conosciuti e utilizzati sono i seguenti (Trefethen e Bau, 1997 [16]):

1) *Precondizionatori diagonali o di Jacobi*: si definisce la matrice M come la matrice diagonale con gli elementi di A sulla diagonale, purché questa sia non singolare:

$$M = diag(A). \tag{4.8.}$$

In questo modo il precondizionatore fa una "scalatura" della matrice di partenza *A*. Questo precondizionatore, nonostante sia molto facile invertire una matrice diagonale, non porta sempre a buoni risultati.

2) *Precondizionatore incompleto di Cholesky*: esso parte dal presupposto che una matrice simmetrica A può essere decomposta, attraverso la fattorizzazione di Cholesky, nella seguente forma LL^T, dove L è una matrice triangolare inferiore. Quando si ha che fare con la risoluzione di sistemi lineari con matrice dei coefficienti A sparsa, ovvero con molti elementi nulli, metodi come quello di Gauss o di Cholesky "distruggono" questi elementi nulli in modo che se $A = LL^T$ solitamente la matrice L non risulta sparsa come la matrice di partenza *A*. In questo modo il sistema di partenza $Ax = b$ diventa $LL^T x = b$ e per trovare la soluzione basta risolve i due più semplici sistemi triangolari $L^T x = b$ e $L\hat{x} = b$. La matrice A, anche se è sparsa, è tale che quando si va a calcolare la fattorizzazione di Cholesky la matrice L può risultare densa. Per "sparsificare" L esiste la *fattorizzazione incompleta* di Cholesky dove vi è il cosiddetto numero di *fill-in* (ovvero il rapporto tra gli elementi non nulli e tutti gli elementi di L), il quale se è alto indica che L è più densa mentre se è basso L è più sparsa. La fattorizzazione incompleta di Cholesky è perciò una variante della fattorizzazione più semplice in cui si cerca di ridurre al minimo questo particolare numero. In pratica la matrice A viene approssimata con il prodotto $\tilde{L}\tilde{L}^T$, con la restrizione che la matrice $\tilde{L}$ abbia lo stesso numero di elementi non nulli della matrice di A. Per usare $\tilde{L}\tilde{L}^T$ come precondizionatore, la soluzione del nuovo sistema $\tilde{L}\tilde{L}^T w = z$ viene calcolata attraverso il processo di sostituzione all'indietro, i.e. l'inversa di $\tilde{L}\tilde{L}^T$ non viene mai calcolata esplicitamente. Questo precondizionatore è molto efficace per una vasta gamma di problemi ma purtroppo non è sempre numericamente stabile.

4.3. Dai metodi di quasi-Newton ad una particolare classe di precondizionatori

In questo paragrafo verrà descritta una classe particolare di precondizionatori (Gratton, 2009 [7]), chiamati *Limited Memory Preconditioners (LMP),* per la risoluzione di una sequenza di sistemi lineari $Ax = b_i$ con le seguenti caratteristiche:

- Elevate dimensioni;
- Simmetrici;
- Definiti positivi;
- Con matrice dei coefficienti sempre uguale ma con diverso vettore dei termini noti.

Come già è stato detto nei capitoli precedenti risolvere, un sistema lineare $Ax = b$ con l'importante condizione che la matrice dei coefficienti corrispondente $A \epsilon R^{nxn}$sia simmetrica e definita positiva equivale ad minimizzare la relativa forma quadratica ad esso associata

$$f(x) = \frac{1}{2}x^T Ax - b^T x + c. \qquad (4.9.)$$

Il metodo che a tal fine è il cosiddetto metodo del *gradiente coniugato di* M. Hestenes and E. Stiefel. Questo procedimento parte da un punto iniziale $x^{(0)}$ (con corrispondente residuo $r^{(0)} = -\nabla f\left(x^{(0)}\right) = b - Ax^{(0)}$) per poi generare una serie di iterazioni $x^{(k)}$ che minimizzano la forma quadratica f sul sottospazio di Krylov :

$$x^{(0)} + \text{span}\left\{r^{(0)}, Ar^{(0)}, \dots, A^{k-1}r^{(0)}\right\}. \qquad (4.10.)$$

È ben noto che, per ottenere una rapida convergenza alla soluzione del sistema lineare in esame, il metodo del gradiente coniugato spesso necessita di un buon *precondizionatore*, che può essere considerato come una approssimazione di A^{-1}.

Gli studiosi Gratton, Sartenaer e Tshimanga nel loro articolo "*On a class of limited memory preconditioners for large scale linear systems with multiple right-hand sides*"[7] partono dall'assunto che A^{-1} corrisponde all'inversa della matrice Hessiana, ovvero matrice quadrata $n \times n$ delle derivate parziali seconde, della funzione quadratica. Da questa considerazione deducono che qualsiasi approssimazione dell'inversa della matrice Hessiana ottenuta attraverso un metodo di *quasi-Newton* (si veda anche Grippo e Sciandrone, 2011 [8]) ad un costo computazionale accettabile, è un buon candidato ad essere usata come precondizionatore in caso di sistemi lineari a grandi dimensioni, usando il metodo del gradiente coniugato.

I metodi *quasi-Newton* sono una variante del metodo più semplice di Newton e si basano sulla conoscenza delle sole derivate prime della $f(x)$. Il metodo di Newton è definito dalla seguente iterazione:

$$x^{(k+1)} = x^{(k)} - \left(\nabla^2 f\left(x^{(k)}\right)\right)^{-1} \nabla f\left(x^{(k)}\right); \qquad (4.11.)$$

la differenza con il metodo di *quasi-Newton* è che al posto di invertire la matrice Hessiana della funzione quadratica f si cerca di estrarre delle informazioni relative alle derivate seconde dalle variazioni di $\nabla f(x)$ conservando una rapidità di convergenza superlineare. I metodi di tipo quasi-Newton vengono definiti dal seguente algoritmo:

$$x^{(k+1)} = x^{(k)} - \alpha_k {B_k}^{-1} \nabla f\left(x^{(k)}\right), \qquad (4.12.)$$

dove α_k è il passo lungo la direzione di ricerca e la matrice simmetrica B_k, che viene aggiornata ad ogni iterazione, approssima la matrice Hessiana $\nabla^2 f\left(x^{(k)}\right)$.
La matrice B_{k+1} viene definita in modo da soddisfare la seguente equazione:

$$\nabla f\left(x^{(k+1)}\right) - \nabla f\left(x^{(k)}\right) = B_{k+1}(x^{(k+1)} - x^{(k)}) \qquad (4.13.)$$

chiamata anche *equazione quasi-Newton*. Se si pone che:

- la variazione dei punti : $s^{(k)} = x^{(k)} - x^{(k-1)}$, (4.14.)
- la variazione dei gradienti : $y^{(k)} = \nabla f\left(x^{(k)}\right) - \nabla f\left(x^{(k-1)}\right)$, (4.15.)

la matrice B_{k+1} può essere definita, aggiornando B_k, con la seguente formula:

$$B_{k+1} = B_k + \Delta B_k \quad \text{con} \quad y^{(k)} = (B_k + \Delta B_k)s^{(k)}. \qquad (4.16.)$$

Un altro modo equivalente per definire questo metodo particolare è quello di generare una successione $\{H_k\}$ di matrici simmetriche e definite positive ognuna vista come un'approssimazione dell'inversa della matrice Hessiana; l'iterazione sarà calcolata nel seguente modo:

$$x^{(k+1)} = x^{(k)} + \alpha_k d_k \text{ con } d_k = -H_k \nabla f\left(x^{(k)}\right) \qquad (4.17.)$$

dove la matrice H_{k+1} può essere definita, aggiornando H_k, con la seguente formula:

$$H_{k+1} = H_k + \Delta H_k \quad \text{con} \quad s^{(k)} = (H_k + \Delta H_k)y^{(k)}. \qquad (4.18.)$$

Come già detto in precedenza questi metodi sono molto adatti per risolvere sistemi lineari di grandi dimensioni, ma possono portare dei problemi perché in generale la

matrice H_k, che è densa, può essere molto costosa dal punto di vista computazionale e di memorizzazione.

Per questo motivo sono stati proposti i *metodi quasi-Newton a memoria limitata* in cui una versione modificata di H_k è memorizzata implicitamente utilizzando un numero fissato *m* di coppie $\{ s^{(k)}, y^{(k)}\}$ che soddisfano la seguente condizione:

$$(s^{(k)})^T y^{(k)} > 0. \qquad (4.19.)$$

Questi metodi riescono a fornire un accettabile tasso di convergenza nonostante come detto abbiano dei requisiti di memorizzazione modesti.
L'algoritmo specifico di quasi-Newton, con cui Gratton, Sartenaer e Tshimanga cercano di costruire una nuova classe di precondizionatori, è quello conosciuto con il nome *L-BFGS*, che come suggerisce il nome è basato sulla formula di aggiornamento, della matrice H_k , di *BFGS* (*Broyden-Fletcher-Goldfarb-Shanno*), che viene aggiornata ad ogni iterazione dalla seguente formula (Nocedal Wright, 1999 [10]):

$$H_k = (V_k)^T H_{k-1} V_k + d_k s^{(k)} \left(s^{(k)}\right)^T, \qquad (4.20)$$

dove:

$$d_k = 1/\left(y^{(k)}\right)^T s^{(k)} \qquad V_k = I_n - d_k y^{(k)} \left(s^{(k)}\right)^T$$

e I_n rappresenta la matrice di identità di ordine *n* e $s^{(k)}, y^{(k)}$ sono stati definiti, precedentemente, rispettivamente come variazione tra un'iterata e l'altra e la variazione dei gradienti. La matrice H_{k+1} è ottenuta aggiornando la matrice precedente H_k, utilizzando la coppia di correzione $\{ s^{(k)}, y^{(k)}\}$. La matrice H_k può essere semplificata nella seguente formula:

$$H_k = \left(I_n - \frac{y^{(k)} \left(s^{(k)}\right)^T}{(y^{(k)})^T s^{(k)}} \right) H_{k-1} \left(I_n - \frac{y^{(k)} \left(s^{(k)}\right)^T}{(y^{(k)})^T s^{(k)}} \right) + \frac{s^{(k)} \left(s^{(k)}\right)^T}{(y^{(k)})^T s^{(k)}} \ . \qquad (4.21.)$$

Purtroppo non esiste una formula per scegliere H_0 che funzioni bene in qualunque caso. Un esempio potrebbe essere quello di usare delle specifiche informazioni sul problema per impostare che H_0 sia uguale all'approssimazione della matrice Hessiana calcolata attraverso il metodo delle differenze finite in $x^{(0)}$.In caso contrario si può impostare che H_0 coincide semplicemente con la matrice identità o un suo multiplo (il multiplo deve riflettere la scala delle variabili). Come già accennato in precedenza la matrice H_k, approssimazione dell'inversa della matrice Hessiana, generata dal metodo *BFGS* sarà generalmente densa diventando così troppo costosa dal punto di vista della memorizzazione e della manipolazione. Per ovviare a questo problema esistono i *metodi quasi-Newton a memoria* limitata i quali utilizzano una

rappresentazione alternativa del processo di aggiornamento delle matrici H_k. In pratica si inizia scegliendo una matrice iniziale H_k^0, la quale viene aggiornata *m* volte, attraverso la formula di BFGS (4.20.), utilizzando *m* di coppie $\{s^{(i)}, y^{(i)}\}$ con $i = k-m, \dots, k-1$.

Si arriva a concludere che nei metodi L-BFGS la matrice H_k è data dalla seguente formula:

$$\begin{aligned} H_k = {} & (V_{k-1}^T \dots V_{k-m}^T) H_k^0 (V_{k-m} \dots V_{k-1}) \\ & + d_{k-m} (V_{k-1}^T \dots V_{k-m+1}^T) s^{(k-m)} \left(s^{(k-m)}\right)^T (V_{k-m+1} \dots V_{k-1}) \\ & + d_{k-m+1} (V_{k-1}^T \dots V_{k-m+2}^T) s^{(k-m+1)} \left(s^{(k-m+1)}\right)^T (V_{k-m+2} \dots V_{k-1}) \\ & + \cdots \\ & + d_{k-1} s^{(k-1)} \left(s^{(k-1)}\right)^T. \end{aligned}$$

Dato che si ha a che fare con una minimizzazione di una forma quadratica definita positiva le iterazioni generate dal metodo *L-BFGS* sono identiche a quelle generate dal metodo del gradiente coniugato.
Ricordando che viene preso come step $s^{(k)} = \alpha_k d_k$ ottenuto attraverso un metodo *line search* lungo la direzione $d_k = -H_k \nabla f\left(x^{(k)}\right)$, la variazione dei gradienti è data dalla seguente formula:

$$y^{(k)} = r^{(k)} - r^{(k-1)} = \alpha_k A d_k = A\, s^{(k)} \qquad (4.22.)$$

dove $r^{(k)} = Ax^{(k)} - b$ rappresenta il residuo all'iterazione $x^{(k)}$. In questo modo possiamo scrivere la matrice V_k , definita prima, come:

$$V_k = \left(I_n - A \frac{s^{(k)} \left(s^{(k)}\right)^T}{(s^{(k)})^T A s^{(k)}} \right), \qquad (4.23.)$$

che è equivalente a:

$$V_k^T H_{k-1} V_k = V_k^T (V_{k-1}^T H_{k-2} V_{k-1}) V_k + V_k^T \frac{s^{(k-1)} \left(s^{(k-1)}\right)^T}{(s^{(k-1)})^T A s^{(k-1)}} V_k\,, \qquad (4.24.)$$

dove H_{k-1} è stata calcolata con la formula (4.21.).

Dal fatto che i vettori $s^{(i)}$ sono A - coniugati (i.e. $s^{(i)} A s^{(j)} = \delta_{ij}$ dove δ è il simbolo di Kronecker che è uguale a 1 se $i = j$ e 0 se $i \neq j$), segue che $V_k^T s^{(k-1)} = s^{(k-1)}$, la qualcosa a sua volta implica che:

$$V_k^T H_{k-1} V_k = V_k^T (V_{k-1}^T H_{k-2} V_{k-1}) V_k + \frac{s^{(k-1)} \left(s^{(k-1)}\right)^T}{(s^{(k-1)})^T A s^{(k-1)}}$$
$$= \cdots = V_k^T V_{k-1}^T \dots V_1^T H_0 V_1 \dots V_{k-1} V_k + \sum_{i=1}^{k-1} \frac{s^{(i)} \left(s^{(i)}\right)^T}{(s^{(i)})^T A s^{(i)}} .$$

Dopo queste considerazioni, facendo alcune sostituzioni, si ottiene che:

$$H_k = V_k^T V_{k-1}^T \dots V_1^T H_0 V_1 \dots V_{k-1} V_k + \sum_{i=1}^{k} \frac{s^{(i)} \left(s^{(i)}\right)^T}{(s^{(i)})^T A s^{(i)}} ; \qquad (4.25.)$$

ricordando la *A-coniugatezza* dei vettori $s^{(i)}$, si osserva che $V_1 \dots V_{k-1} V_k = I_n - \sum_{i=1}^{k} A \frac{s^{(i)} \left(s^{(i)}\right)^T}{\left(s^{(i)}\right)^T A s^{(i)}}$, e si arriva alla conclusione che:

$$H_k = \left(I_n - \sum_{i=1}^{k} \frac{s^{(i)} \left(s^{(i)}\right)^T}{(s^{(i)})^T A s^{(i)}} A \right) H_0 \left(I_n - \sum_{i=1}^{k} A \frac{s^{(i)} \left(s^{(i)}\right)^T}{(s^{(i)})^T A s^{(i)}} \right) + \sum_{i=1}^{k} \frac{s^{(i)} \left(s^{(i)}\right)^T}{(s^{(i)})^T A s^{(i)}} . \quad (4.26.)$$

Nell'articolo di Gratton, Sartenaer e Tshimanga si afferma che dato un sistema lineare con matrice dei coefficienti *A* simmetrica e definita positiva di ordine *n*, posto $S \epsilon R^{n \times k} = \left[s^{(1)}, \dots, s^{(k)} \right]$, e ponendo $(S^T A S)^{-1} = diag\left(\left(\left(s^{(i)}\right)^T A s^{(i)} \right)^{-1} \right)$ con $M = H_0$, si osserva che:

$$H_k = [I_n - S(S^T A S)^{-1} S^T A] M [I_n - A S (S^T A S)^{-1} S^T] + S(S^T A S)^{-1} S^T . \quad (4.27.)$$

La matrice simmetrica H_k di ordine *n* può essere usata come precondizionatore per risolvere una sequenza di sistemi lineari del tipo $Ax = b_i$. M è chiamato *precondizionatore di primo livello* ed è calcolato per esempio con le tecniche descritte in precedenza mentre la matrice simmetrica H_k è un *precondizionatore di secondo livello* ed è chiamato *Limited Memory Preconditioner (LMP).*

In pratica per affrontare una sequenza di sistemi lineari del tipo $Ax = b_i$, si risolve il primo dei sistemi lineari usando *m* passi del metodo del gradiente coniugato "non precondizionato", successivamente si salvano le m coppie di correzione $\left\{ s^{(i)}, y^{(i)} \right\}$ generate durante le iterazioni prodotte da questo, per poi usare le m iterazioni per costruire il precondizionatore. Il resto dei sistemi verrà risolto usando il metodo del gradiente coniugato, questa volta precondizionato (con il precondizionatore costruito nella prima fase).

4.4. Precondizionatori costruiti con metodi di Krylov

In questo paragrafo viene presentato un precondizionatore, proposto da G. Fasano e M. Roma [5], per la risoluzione di sistemi lineari indefiniti di grandi dimensioni. In pratica attraverso un metodo iterativo di tipo Krylov, il quale deve soddisfare determinate condizioni, costruiscono un'approssimazione della matrice inversa della matrice di partenza A di un sistema lineare del tipo $Ax = b$. Si vedrà che il loro precondizionatore ha qualche somiglianza con il *Limited Memory Preconditioner*, di S. Gratton&Co, descritto prima, con la differenza che è meno costoso dal punto di vista computazionale. Come già è stato detto nei capitoli precedenti i metodi di Krylov sono di solito utilizzati per risolvere iterativamente sistemi lineari di grandi dimensioni. Esempi di questi metodi sono il metodo del *gradiente coniugato* (spiegato nel capitolo 3) per sistemi simmetrici definiti positivi, il processo di *Lanczos* per sistemi simmetrici indefiniti, il metodo *GMRES* (Generalized Minimal Residual) e il metodo del *gradiente biconiugato stabilizzato* per sistemi non simmetrici. Nel loro articolo G. Fasano e M. Roma considerano sistemi lineari simmetrici indefiniti e si concentrano sulla costruzione di precondizionatori costruiti utilizzando una decomposizione della matrice di sistema ottenuta attraverso un metodo di Krylov. Questo particolare metodo di Krylov, per risolvere questo tipo di problemi, dovrà soddisfare determinati requisiti. L'idea del loro approccio è quella di applicare un metodo di Krylov, in questo caso quello del gradiente coniugato, per generare un'approssimazione definita positiva della matrice inversa del sistema in esame, quest'ultima verrà usata per il costruire il loro precondizionatore. Questo particolare precondizionatore è ottenuto memorizzando solo alcuni vettori, senza calcolare il prodotto di matrici o matrici inverse. I due autori assumono che gli elementi della matrice A del sistema non siano noti e le uniche sue informazioni disponibili sono ottenute per mezzo di una routine che calcola il prodotto della matrice A per un vettore.

Come già è stato detto in precedenza il problema affrontato è quello di trovare la soluzione di un sistema lineare simmetrico a grandi dimensioni, attraverso l'utilizzo di un metodo di Krylov. Il metodo di Krylov scelto deve soddisfare delle determinate condizioni al fine di ottenere in maniera iterativa una decomposizione della matrice di sistema per costruire il precondizionatore in esame.
A questo scopo si considera il sistema lineare *indefinito*:

$$Ax = b, \tag{4.28.}$$

dove la matrice dei coefficienti $A \in R^{nxn}$ è simmetrica e non singolare, n è molto grande e $b \in R^n$ è il vettore dei termini noti. Viene supposto che qualunque metodo di Krylov possa essere utilizzato per risolvere tale sistema, per esempio il *processo di Lanczos* o *il metodo del gradiente coniugato*. Questi due metodi sono equivalenti fintanto che la matrice dei coefficienti A sia definita positiva, tenendo conto che il metodo del gradiente coniugato, nonostante sia il più economico tra i due, non può

affrontare il caso indefinito. In questa tesi viene considerato solo il metodo del gradiente coniugato, il quale è spiegato nel Capitolo 3.
Considerato qualsiasi metodo di Krylov per risolvere il sistema lineare (4.28.) si suppone che questo viene eseguito per un certo numero finito di iterazioni $m \ll n$.

All'iterazione $m \geq 1$ del metodo di Krylov, in questo caso il metodo del gradiente coniugato, le matrici $R_m \epsilon R^{nxm}$, $T_m \epsilon R^{nxm}$, $L_m \epsilon R^{mxm}$ e $B_m \epsilon R^{mxm}$ sono definite come segue:

$$AR_m = R_m T_m + \rho_{m+1} u_{m+1} e_m^T \ , \qquad (4.29.)$$
$$T_m = L_m B_m L_m^T \ ,$$

dove, si indica il residuo all'iterazione *k* con $r^{(k)} = b - Ax^{(k)}$ e:

- la matrice $R_m = \left(u^{(1)} \dots u^{(m)}\right)$, tenendo conto che $\left(u^{(i)}\right)^T u^{(j)} = 0$ e $\left\|u^{(i)}\right\| = 1$, $1 \leq i \neq j \leq h + 1$;

- la matrice T_m è:
  - *simmetrica tridiagonale* (ovvero al di fuori della diagonale principale e delle linee immediatamente al di sopra e al di sotto di essa , ha solo valori nulli),
  - *irriducibile,*
  - *non singolare;*

- la matrice L_m è *triangolare inferiore unitaria* (ovvero ha nulli tutti gli elementi al di sopra della diagonale principale e i numeri su tale diagonale sono tutti uguali ad 1);

- la matrice B_m è:
  - *diagonale a blocchi* (ossia è scritta in modo da raggrupparne gli elementi in blocchi diagonali quadrati, ossia sottomatrici della matrice stessa, sulla diagonale e i cui altri blocchi contengono solo zeri),
  - *non singolare*, la cui inversa è "facile da calcolare".

Sulla base di queste assunzioni, G. Fasano e M. Roma sono riusciti a definire il loro precondizionatore e a descriverne le proprietà.
Sia $Z \epsilon R^{nxn}$ una matrice simmetrica non singolare tale che $Z = WDW^T$, dove $W \epsilon R^{nxn}$ è non singolare e $D \epsilon R^{nxn}$ è diagonale a blocchi con *simple pattern*, si definisce:

$$|Z| = W|D|W^T$$

dove $|D|$ è una matrice diagonale a blocchi definiti positivi. In particolare quando D è diagonale, $|D|$ è il valore assoluto degli elementi diagonali di D.

Se viene considerato qualunque metodo di Krylov per risolvere il sistema lineare (4.4.1), e si suppone che vengano effettuate $m \leq n$ iterazioni tenendo conto delle posizioni (4.4.2), il precondizionatore costruito è definito dalla seguente matrice:

$$M_m = (I_n - R_m R_m^T) + R_m |T_m| R_m^T \,, \qquad (4.30.)$$

dove R_m e T_m soddisfano la relazione (4.4.2) e $|T_m| = L_m |B_m| L_m^T$. La matrice inversa M_m^{-1} è data dalla seguente formula:

$$M_m^{-1} = (I_n - R_m R_m^T) + R_m |T_m|^{-1} R_m^T \,, \qquad (4.31.)$$

Inoltre se $m = n$, allora $M_n^{-1} = R_n |T_n|^{-1} R_n^T = |A^{-1}|$.

Questa matrice M_m possiede le seguenti proprietà:

1) M_m è una matrice simmetrica e non-singolare;

2) M_m è una matrice definita positiva e il suo spettro (ovvero l'insieme degli autovalori) $\Lambda(M_m)$ è dato dalla seguente formula:

$$\Lambda(M_m) = \Lambda(|T_m|) \cup \Lambda(I_{n-m}) \,, \qquad m = 1, \ldots, n-1;$$

3) $\Lambda(M_m^{-1} A) \equiv \Lambda(A M_m^{-1})$ e contiene almeno $m - 2$ autovalori nell'insieme $\{-1, +1\}$, $m = 2, \ldots, n-1$;

4) se la matrice A è definita positiva, allora $\Lambda(M_m^{-1} A)$ contiene almeno $m - 1$ autovalori pari a +1, $m = 1, \ldots, n-1$;

5) se $m = n$, allora $\Lambda(M_m) = \Lambda(|T_m|)$ e $\Lambda(M_m^{-1} A) = \Lambda(A M_m^{-1}) \subseteq \{-1, +1\}$.

Dopo queste considerazioni G. Fasano e M. Roma concludono che la matrice M_m^{-1}, quando $m \ll n$, può essere utilizzato come precondizionatore per risolvere il sistema lineare definito in (4.28.)

Per costruire iterativamente la matrice di precondizionamento in questo caso come metodo di Krylov è stato utilizzato il metodo del gradiente coniugato.
Per chiarezza si consiglia di rivedere l'algoritmo di questo particolare metodo nell'**Appendice A.1.**.

Dopo $m \leq n$ iterazioni del metodo del gradiente coniugato, le matrici definite nel caso generale (4.29.) saranno le seguenti:

$$R_m = \left(\frac{r^{(1)}}{\|r^{(1)}\|} \dots \frac{r^{(m)}}{\|r^{(m)}\|}\right) \epsilon\, R^{nxm}, \quad P_m = \left(\frac{p^{(1)}}{\|r^{(1)}\|} \dots \frac{p^{(m)}}{\|r^{(m)}\|}\right) \epsilon\, R^{nxm},$$

con le seguenti relazioni:

$$\begin{aligned} R_m^T R_m &= I_m \\ AR_m &= R_m\, T_m + \rho_{m+1} r_{m+1} e_m^T\,, \quad con\ \ \rho_{m+1} \neq 0, \end{aligned} \tag{4.32.}$$

dove $e_m^T = (0 \dots 01)\, \epsilon\, R^m$, la matrice T_m è tridiagonale ed è data da:

$$T_m = L_m D_m L_m^T\,, \tag{4.33.}$$

dove

$$L_m = \begin{pmatrix} 1 & & & \\ -\sqrt{\beta_1} & 1 & & \\ & -\sqrt{\beta_2} & \ddots & \\ & & \ddots & \\ & & -\sqrt{\beta_{m-1}} & 1 \end{pmatrix}$$

e

$$D_m = diag\left\{\frac{1}{\alpha_1}, \frac{1}{\alpha_2}, \dots, \frac{1}{\alpha_m}\right\}.$$

Se $r^{(i)} \neq 0, i = 1, \dots, m$ e $r^{(m+1)} = 0$ allora $x^{(m+1)}$ risolve il sistema lineare (4.28.), i.e. $Ax^{(m+1)} = b$ in modo che non sia necessario costruire un precondizionatore. Altrimenti quando $r^{(m+1)} \neq 0$ allora da (4.32.) si ottiene che:

$$R_m^T A R_m = T_m. \tag{4.34.}$$

Inoltre dato la matrice A è non singolare anche la matrice T_m è non singolare e irriducibile. Inoltre si ha che:

$$P_m L_m^T = R_m\,. \tag{4.35.}$$

Di conseguenza mediante l'algoritmo del gradiente coniugato le matrici R_m, T_m e L_m, le quali devono soddisfare le assunzioni definite nel caso generale, possono essere costruite iterativamente. In questo il precondizionatore M_n^{-1}, ottenuto utilizzando le espressioni (4.35.) e (4.33.) nella formula (4.31.), può essere riscritto come:

$$M_m^{-1} = (I_n - P_m L_m^T L_m P_m^T) + P_m L_m^T |T_m|^{-1} L_m P_m^T$$

$$= (I_n - P_m L_m^T L_m P_m^T) + P_m |D_m|^{-1} P_m^T , \qquad (4.36.)$$

dove:

$$|T_m| = L_m |D_m| L_m^T \quad \text{e} \quad |D_m| = diag\left\{\frac{1}{|\alpha_1|}, \frac{1}{|\alpha_2|}, \dots, \frac{1}{|\alpha_m|}\right\}.$$

Si può notare che nella costruzione del precondizionatore attraverso il metodo del gradiente coniugato non è richiesto calcolare alcuna inversa della matrice (a parte la matrice $|D_m|^{-1}$ che è una matrice diagonale e quindi facilmente invertibile). In particolar modo la memorizzazione delle sole matrici P_m e L_m per calcolare il precondizionatore M_m^{-1}. Come ultima osservazione si può osservare, sebbene la matrice P_m debba essere completamente memorizzata, la matrice L_m è sparsa con una struttura molto semplice.
Prima di andare a vedere con l'introduzione di questo precondizionatore M_m^{-1} come cambia il costo computazionale viene proposto l'algoritmo del *gradiente coniugato precondizionato* (si veda **Appendice A.1**.). Dopo aver calcolato un certo numero di iterazioni del gradiente coniugato si calcola, attraverso a queste informazioni, il precondizionatore, il quale viene riutilizzato nel gradiente coniugato per ottenere metodo del gradiente coniugato precondizionato.

Come si può vedere dall'algoritmo nessun suo coefficiente dipende esplicitamente dal precondizionatore M^{-1}, l'unico effetto del precondizionatore è attraverso il prodotto $M^{-1}r$, dove r è un vettore di dimensione n. Se si considera questo algoritmo con l'utilizzo del precondizionatore (4.4.4.), costruito G. Fasano e M. Roma, si ha che:

$$M_m^{-1} r = [(I_n - R_m R_m^T) + R_m |T_m|^{-1} R_m^T] r = r + R_m (|T_m|^{-1} - I_n) R_m^T r, \quad (4.37.)$$

in modo che il costo complessivo computazionale $\mathcal{C}(M_m^{-1} r)$ per calcolare il prodotto $M_m^{-1} r$ è dato da:

$$\mathcal{C}(M_m^{-1} r) = mn + \mathcal{C}(|T_m|^{-1}) + nm = 2mn + \mathcal{C}(|T_m|^{-1}) \leq O(m^3), \quad (4.38.)$$

dove $\mathcal{C}(|T_m|^{-1})$ include il costo computazionale di $|T_m|^{-1}$ e il costo per calcolare il prodotto $|T_m|^{-1} v$ con $v \in R^m$. Per stimare più facilmente $\mathcal{C}(|T_m|^{-1})$ si può vedere:

$$|T_m|^{-1} = L_m^{-T} |D_m|^{-1} L_m^{-1} \qquad (4.39.)$$

dove la matrice L_m^{-1} è facile da calcolare data la sparsità di L_m. In questo caso il costo complessivo $\mathcal{C}(|T_m|^{-1})$ è dato dalla somma dei seguenti costi:

- il costo computazionale di L_m^{-1} che corrisponde ad:

$$(m-1)+(m-2)+\cdots+1=\frac{m}{2}(m-1),$$

- il costo per calcolare $|T_m|^{-1}v$ con $v\,\epsilon\,R^m$, che corrisponde ad:

$$2[(m-1)+(m-2)+\cdots+1]+m=m^2.$$

Pertanto il costo complessivo $\mathcal{C}(M_m^{-1}r)$ per calcolare $M_m^{-1}r$ coincide con:

$$\mathcal{C}(M_m^{-1}r)=2mn+\mathcal{C}(|T_m|^{-1})=2mn+\frac{3}{2}m\left(m-\frac{1}{3}\right), \qquad (4.40.)$$

dove è $\approx 2mn$ quando $m \ll n$.

Capitolo 5 Analisi numeriche

In questo capitolo vengono riportati i risultati di un'analisi numerica su vari sistemi lineari $Ax = b$ simmetrici e definiti positivi di grandi dimensioni. Si andranno a confrontare le prestazioni dei due precondizionatori descritti nel precedente capitolo:

- *Limited Memory Preconditioners (LMP)* di Gratton, Sartenaer e Tshimanga; dove, come spiegato nel Capitolo 4 (paragrafo 4.3.), un'approssimazione dell'inversa della matrice Hessiana ottenuta attraverso un metodo di *quasi-Newton* è un buon candidato ad essere usato come precondizionatore in caso di sistemi lineari a grandi dimensioni, usando il metodo del gradiente coniugato;

- Precondizionatore *FaRo*, proposto da G.Fasano e M.Roma; in questo caso il precondizionatore, come spiegato nel capitolo 4 (paragrafo 4.4.), è costruito utilizzando una decomposizione della matrice di sistema ottenuta attraverso un metodo di Krylov con determinati requisiti (eventualmente anche il gradiente coniugato).

In pratica la strategia usata per costruire i due precondizionatori, impiegati nella risoluzione di sistemi lineari simmetrici e definiti positivi, è divisa in due fasi:

1) prima vengono eseguite $m \leq n$ iterazioni del metodo del gradiente coniugato "non precondizionato";

2) da queste m iterazioni del metodo del gradiente coniugato si costruisce il precondizionatore e le matrici che lo caratterizzano.

Come si può osservare nella costruzione dei precondizionatori viene utilizzato il metodo di Krylov del gradiente coniugato, il quale risulta molto efficace nella risoluzione di sistemi lineari con matrici di elevate dimensioni e sparse, ed inoltre risulta meno oneroso in termini di costo computazionale e tempo di calcolo rispetto ad altri metodi iterativi.
La differenza tra i due precondizionatori testati è che quello proposto da G.Fasano e M.Roma è ottenuto memorizzando un numero inferiore di vettori, senza calcolare il prodotto di matrici o matrici inverse, e le uniche informazioni disponibili sulla matrice del sistema sono ottenute per mezzo di una routine che calcola il prodotto della matrice per un vettore. In questo caso calcolare il prodotto di una matrice per un vettore è meno costoso dal punto di vista computazionale che calcolare il prodotto di matrici o matrici inverse.

Il software, utilizzato per implementare i vari sistemi lineari, il metodo del gradiente coniugato e i due precondizionatori, è stato *Matlab* versione *R2012b*.
Il confronto delle prestazioni dei due precondizionatori viene portato avanti in funzione delle dimensioni della matrice, della sua sparsità e del suo condizionamento.
Sono stati presi in considerazione vari sistemi lineari simmetrici, del tipo $Ax = b$ in cui:

- le varie matrici dei coefficienti $A \epsilon R^{nxn}$ analizzate vengono prese dal sito "*Matrix Market*" (*http://math.nist.gov/MatrixMarket/index.html*) in base alle loro caratteristiche e le loro dimensioni variano da circa $n = 100$ a $n = 2000$ variabili;

- i vari vettori dei termini noti $b \epsilon R^n$ vengono creati randomicamente utilizzando il comando *rand* di Matlab;

- Si ricorda che metodi iterativi come il metodo del gradiente coniugato, per risolvere un sistema lineare $Ax = b$, con $det(A) \neq 0$, costruiscono a partire da un vettore iniziale una opportuna successione di vettori $\{x^{(k)}\}_{k\geq 0}$ che sotto specifiche condizioni convergerà alla soluzione cercata. In questo caso il vettore iniziale $x^{(0)}$ è il vettore formato da tutti elementi nulli.

Come criterio di arresto del metodo del gradiente coniugato nel calcolare la soluzione di un sistema lineare simmetrico $Ax = b$ viene utilizzato il duplice criterio:

$$r^{(k)} = \left\|Ax^{(k)} - b\right\| \leq toll \qquad \& \qquad n° \, delle \, iterazioni \leq ITMAX, \tag{5.1.}$$

ovvero il residuo ad ogni iterazione del metodo di Krylov deve essere sufficientemente piccolo (nel nostro caso di una quantità $toll = 10^{-10}$) e si deve fissare a priori il numero massimo di iterazioni che nel nostro caso è la dimensione n della matrice dei coefficienti $A \epsilon R^{nxn}$ in esame.

Come primo caso viene considerato un sistema lineare $Ax = b$, di dimensione $n = 100$, in cui gli elementi della matrice dei coefficienti $A \epsilon R^{100x100}$ sono creati randomicamente nel seguente modo:

$$A = \{a_{ij} \epsilon U[-10, +10] \; con \; i, j = 1, \dots, 100\}, \tag{5.2.}$$

dove $a_{ij} = a_{ji}$, dato che si parla sempre di sistemi lineari simmetrici, sono creati randomicamente da una *distribuzione continua Uniforme* definita tra -10 e $+10$.
Con lo stesso tipo di distribuzione è stato creato randomicamente il vettore dei termini noti $b \epsilon R^{100}$.

Dopo aver creato randomicamente la matrice dei coefficienti A viene applicato il metodo del gradiente coniugato per un certo numero di iterazioni, che in questo caso viene fatto variare nel seguente range:

$$m \in \{4,8,12,16,20,40\}. \qquad (5.3.)$$

In questo modo dalle informazioni tratte da queste iterazioni del metodo di Krylov del gradiente coniugato verranno costruiti i due precondizionatori da testare.
Precondizionare un sistema lineare simmetrico e definito positivo $Ax = b$, come già detto nei capitoli precedenti, equivale a voler condizionare la matrice simmetrica dei coefficienti A. Si ricorda che per *condizionare* si intende

1) migliorare il numero di condizionamento di questa matrice, il quale è definito come segue:

$$\mathcal{K}(\mathrm{A}) = \frac{max_i|\lambda_i|}{min_i|\lambda_i|},$$

2) trasformare e clusterizzare lo spettro della matrice A rendendo unitari alcuni autovalori della matrice A.

ovvero il rapporto tra l'autovalore massimo e l'autovalore minimo della matrice A.
Per confrontare i due diversi precondizionatori si è andati così a calcolare il numero di condizionamento della matrice precondizionata per ogni valore m (numero di iterazioni di GC) definito in (5.3.).
Come si può vedere dalla tabella **Tab.1**, in questo sistema lineare considerato, la matrice dei coefficienti non è "ben condizionata", infatti il numero di condizionamento è moderatamente alto rispetto al caso migliore che si possa trovare, ovvero quando $\mathcal{K}(\mathrm{A}) \approx 1$.

	m = 4	**m = 8**	**m = 12**	**m = 16**	**m = 20**	**m = 40**
A	322.6274					
LMP*A	314.6023	313.0455	311.2992	309.7489	304.0434	790.2141
FaRo*A	334.1568	339.3447	351.5484	349.8533	355.4598	271.7388

Tab. 2 Numero di condizionamento ad ogni range di iterazione dei tre casi (Matrice di partenza, Matrice precondizionata con il precondizionatore LMP e matrice precondizionata con il precondizionatore costruito da Fasano e Roma).

Si ricorda che nel caso migliore per ogni valore di m, la matrice precondizionata "dovrebbe" avere un numero di condizionamento minore rispetto alla matrice non precondizionata.
Andando ad osservare la figura **Fig.1** si può determinare in modo migliore e rapidamente se la tecnica del precondizionamento è servita per migliorare il numero

di condizionamento della matrice simmetrica di partenza $A \epsilon R^{100x100}$, formata da elementi creati randomicamente.
Come si può vedere, naturalmente, il numero di condizionamento nel caso non precondizionato è costante al variare delle iterazioni del metodo del gradiente coniugato. Invece si può notare che i due precondizionatori hanno effetti opposti. Da un lato il precondizionatore *LMP*, proposto da Gratton, Sartenaer e Tshimanga, sembra inizialmente migliore rispetto a quello proposto da Fasano e Roma, per valori bassi di m $(m \leq 20)$, ciò è confermato anche dai valori in **Tab.1**. Mentre dall'altro lato il precondizionatore di Fasano e Roma è preferibile per valori più alti di m $(m > 20)$.

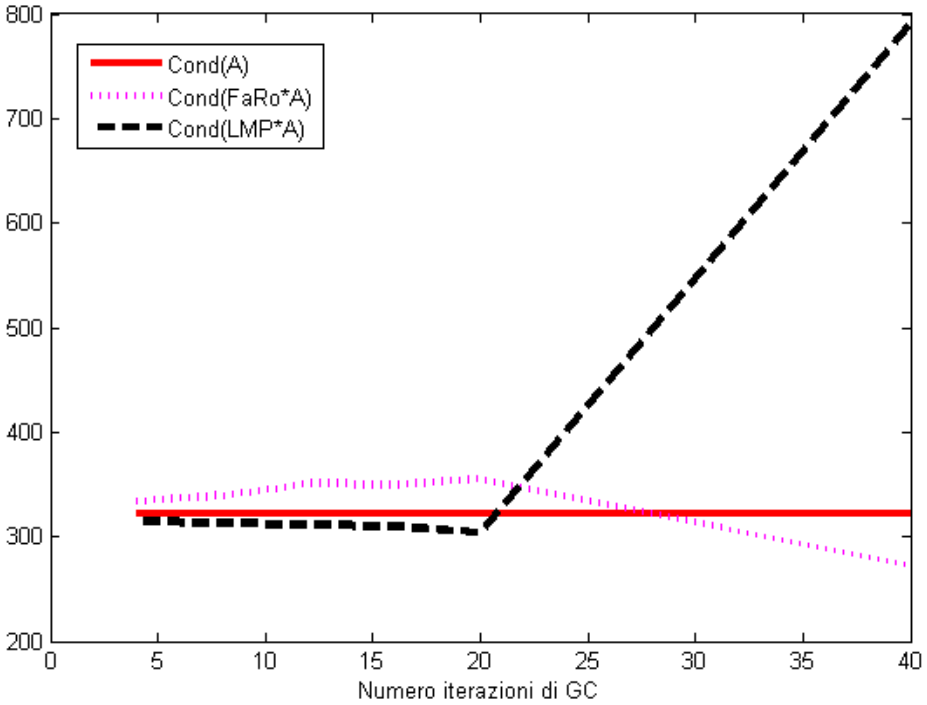


Fig. 3 Grafico del numero di condizionamento (Caso matrice A creata randomicamente)

Oltre al numero di condizionamento, dato un sistema lineare simmetrico e definito positivo $Ax = b$, per confrontare i due precondizionatori, si possono valutare anche gli autovalori delle rispettive matrici precondizionate.
Si ricorda che dato un sistema $Ax = b$, sapendo che la matrice simmetrica A ha per definizione autovalori $\lambda \epsilon R$, gli autovalori delle matrici precondizionate rimangono sempre autovalori reali.
Un'altra importante caratteristica degli autovalori di una matrice precondizionata è che questi risultano meglio clusterizzati della matrice non precondizionata, in modo che siano disposti nell'asse reale il più vicino possibile ad 1.
Per prima cosa, per andare a vedere graficamente in modo migliore come si comportano gli autovalori dopo la tecnica del precondizionamento, si è andati ad ordinare questi dal più piccolo al più grande, in modo che nel grafico si abbia una curva che sale sempre e non una dispersione di punti.

L'ideale è che gli autovalori delle matrici precondizionate siano tutti uguali al valore ±1 perché in questo modo il numero di condizionamento, che è il rapporto in modulo tra l'autovalore massimo e l'autovalore minimo della matrice in esame, è uguale ad 1, che come si è detto nel precedente capitolo è il miglior caso che si possa trovare per un precondizionatore. Si può vedere dalle figure seguenti (**Fig.2 e Fig.3**) che all'aumentare del numero di iterazioni i due precondizionatori riescono a far "schiantare" alcuni degli autovalori della matrice di partenza non precondizionata esattamente al valore +1 o -1. Da ciò si può dedurre che meno iterazioni del metodo del gradiente coniugato vengono fatte più poveri di informazioni saranno i precondizionatori usati.

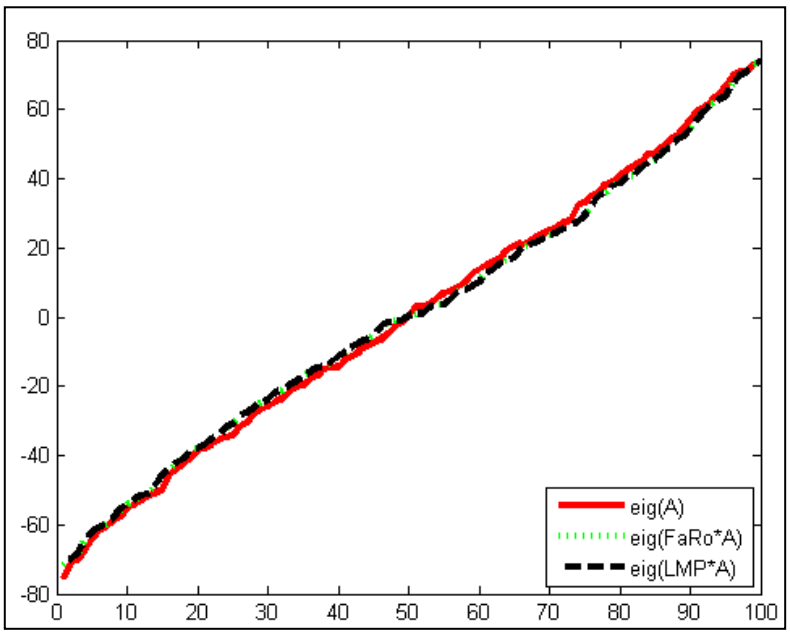


Fig. 4 Grafico degli autovalori, m = 4 iterazioni del GC (Caso matrice A creata randomicamente)

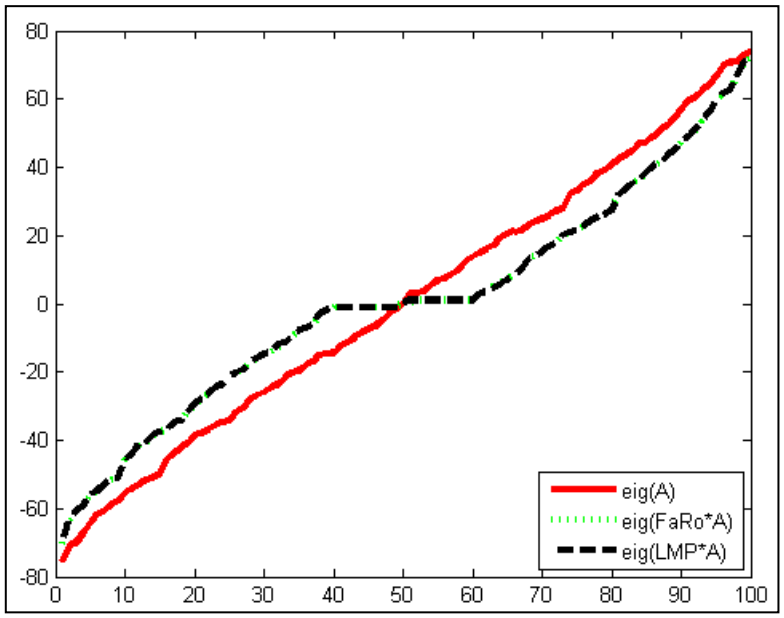


Fig. 5 Grafico degli autovalori, m = 20 iterazioni del GC (Caso matrice A creata randomicamente)

Dopo aver testato i due precondizionatori nel caso in cui, in un sistema lineare simmetrico $Ax = b$, gli elementi della matrice dei coefficienti $A \epsilon R^{nxn}$ siano creati randomicamente, si è voluto testare questi su particolari matrici presenti nel database del sito "*Matrix Market*". In questo sito sono presenti varie matrici di diverse dimensioni e divise in base alle loro caratteristiche.
In questa tesi si è scelto di selezionare due tipi di matrici: *sparse* e *bandate (banded)*.
Una matrice sparsa, come già detto nei capitoli precedenti, è una matrice avente la caratteristica di contenere al suo interno un numero significativo di elementi nulli, mentre una matrice *bandata* è sempre una matrice sparsa dove gli elementi diversi da zero sono tutti posti in *bande* diagonali che comprendono la diagonale principale e, opzionalmente, una o più diagonali alla sua destra od alla sua sinistra.
Si è scelto questo tipo di matrici perché sono molto frequenti nella realtà. Spesso lavorare con matrici sparse o a banda è sempre meno oneroso dal punto di vista computazionale che farlo con matrici quadrate dense (ovvero con pochi elementi nulli) di pari dimensioni; questo perché non avendo molti elementi non nulli si riduce in modo notevole l'impiego di risorse per svolgere operazioni quali le moltiplicazioni, portando così a grandi risparmi in termini di tempo e di complessità di calcolo.
Le matrici scelte sono descritte in **Tab.2**. e **Tab.3.**, con la loro dimensione, descrizione del problema da cui nascono e il loro numero di condizionamento.

Matrici Sparse			
Nome	**Dimensione**	**Descrizione**	$\mathcal{K}(\mathbf{A})$
NOS4	100x100	*Lanczos with partial reorthogonalization Finite element approximation to a beam structure*	$2{,}7 * 10^3$
NOS5	468x468	*Lanczos with partial reorthogonalization 3 story building with attached tower.*	$2{,}9 * 10^4$
BCCSSTK27	1224x1224	*BCS Structural Engineering Matrices (eigenvalue problems and linear equations) Buckling analysis, symmetric half of engine inlet from Boeing jetliner*	$7.7 * 10^4$

Tab. 3 Matrici sparse, con relativa dimensione, descrizione e numero di condizionamento usate per testare i due precondizionatori.

Matrici Bandate			
Nome	**Dimensione**	**Descrizione**	$\mathcal{K}(\mathbf{A})$
NOS1	237x237	*Lanczos with partial reorthogonalization Finite element approximation to biharmonic operator on a beam with one end free and one end fixed. 80 elements with 3 DOF per node.*	$2{,}5 * 10^5$
NOS6	675x675	*Lanczos with partial reorthogonalization Finite difference approximation to Poisson's equation in an L-shaped region, mixed boundary conditions.*	$8 * 10^6$
BCCSSTK14	1806x1806	*BCS Structural Engineering Matrices (linear equations) Roof of the Omni Coliseum, Atlanta*	$1.3 * 10^{10}$

Tab. 3 Matrici bandate (banded), con relativa dimensione, descrizione e numero di condizionamento usate per testare i due precondizionatori.

Come nel primo caso, per ogni matrice scelta viene applicato il metodo del gradiente coniugato, con il vettore dei termini noti $b \epsilon R^n$ creato randomicamente utilizzando il comando *rand* di *Matlab*, per un certo numero di iterazioni. Poi tramite le informazioni ottenute si andranno anche qui a costruire i due diversi precondizionatori. L'obiettivo della tesi è quello di testare l'efficacia di questi due particolari precondizionatori e per questo si è andati ad osservare se per ogni matrice scelta il numero di condizionamento nel caso "precondizionato" (sia con il precondizionatore proposto da Fasano e Roma e sia con il precondizionatore LMP di Gratton, Sartenaer e Tshimanga) migliorava rispetto al caso non precondizionato.
Come si può vedere dal test effettuato, mostrato in **Tab.4.**, sulla matrice quadrata sparsa NOS4 di dimensione $n = 100$, il numero di condizionamento diminuisce solo con 40 iterazioni di Gradiente Coniugato, questo non può essere un buon risultato in quanto è buona norma che quando si va a risolvere un sistema lineare attraverso un metodo iterativo di Krylov il numero delle iterazioni m debba essere al più il 10-20% della dimensione n del sistema lineare in esame.

	m = 4	**m = 8**	**m = 12**	**m = 16**	**m = 20**	**m = 40**
A	$1.5785 * 10^3$					
FaRo*A	$2.3752 * 10^3$	$2.4168 * 10^3$	$2.1975 * 10^3$	$2.2426 * 10^3$	$2.1693 * 10^3$	$0.3217 * 10^3$
LMP*A	$1.8566 * 10^3$	$1.8524 * 10^3$	$1.8470 * 10^3$	$1.8350 * 10^3$	$1.7868 * 10^3$	$0.2425 * 10^3$

Tab. 4 Matrice sparsa NOS4 100x100 - Numero di condizionamento ad ogni range di iterazione dei tre casi (Matrice di partenza, Matrice precondizionata con il precondizionatore LMP e matrice precondizionata con il precondizionatore costruito da Fasano e Roma).

Tutto ciò può essere osservato in maniera più evidente tramite il grafico in **Fig.4**., infatti le curve che rappresentano il numero di condizionamento sia nel caso in cui si usa il precondizionatore LMP di Gratton, Sartenaer e Tshimanga che il precondizionatore proposto da Fasano e Roma, fino a circa 20 iterazioni di Gradiente Coniugato sono al di sopra della curva nel caso non precondizionato per poi andare al di sotto di questa.

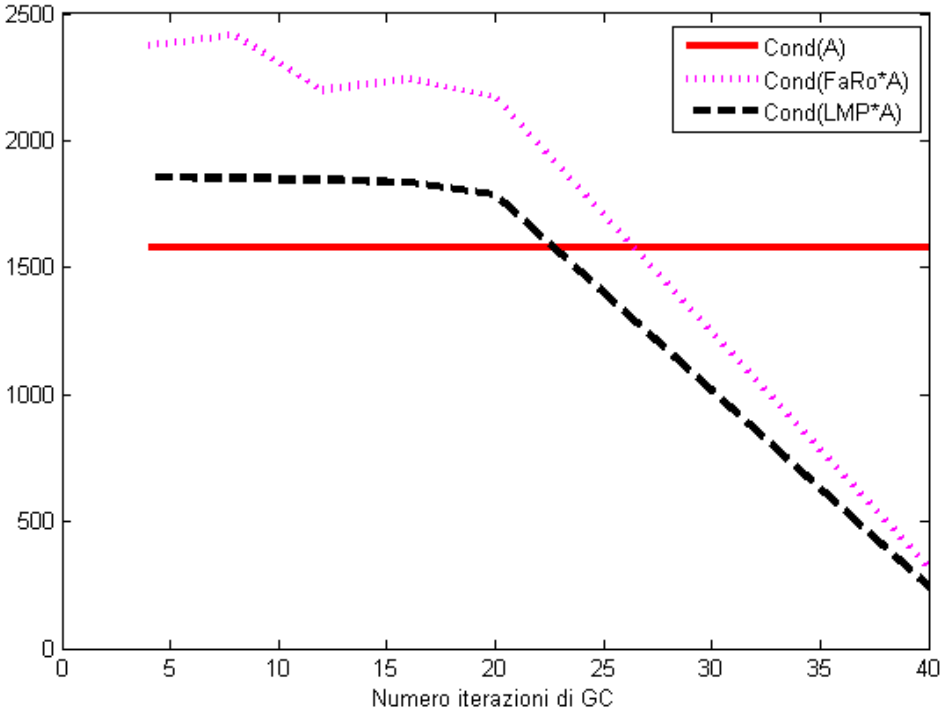


Fig. 6 Grafico del numero di condizionamento (Caso matrice sparsa NOS4 100x100)

Questo miglioramento del numero di condizionamento purtroppo non viene confermato negli altri casi effettuati, sia che la matrice sia sparsa che a banda. Ciò può essere visto nelle tabelle in **Appendice A.2**. Tra i due precondizionatori quello che risulta meno peggiore da questo punto di vista, in tutti i test effettuati, è quello LMP proposto da Gratton, Sartenaer e Tshimanga.

Se si va a vedere come si raggruppano, nei due diversi casi, gli autovalori della matrice precondizionata, si ottengono risultati più soddisfacenti rispetto al confronto dal punto di vista del numero di condizionamento. Un buon precondizionatore è definito tale anche se riesce a clusterizzare il maggior numero di autovalori della matrice precondizionata ad 1.

Andando a prendere come esempio la matrice sparsa NOS4 di dimensione $n = 100$ si può vedere dai grafici in **Fig.5** e **Fig.6,** che i due precondizionatori si comportano allo stesso modo nel clusterizzare gli autovalori nell'asse reale, il più vicino possibile ad 1.

Come si può vedere dalle figure sottostanti con poche iterazioni di Gradiente Coniugato non sono molti gli autovalori della matrice precondizionata uguali ad 1, ma con l'aumentare del numero di iterazioni questi aumentano sempre di più. Questo comportamento degli autovalori è stato riscontrato anche con le altre matrici descritte precedentemente in **Tab.2** e **Tab.3**.

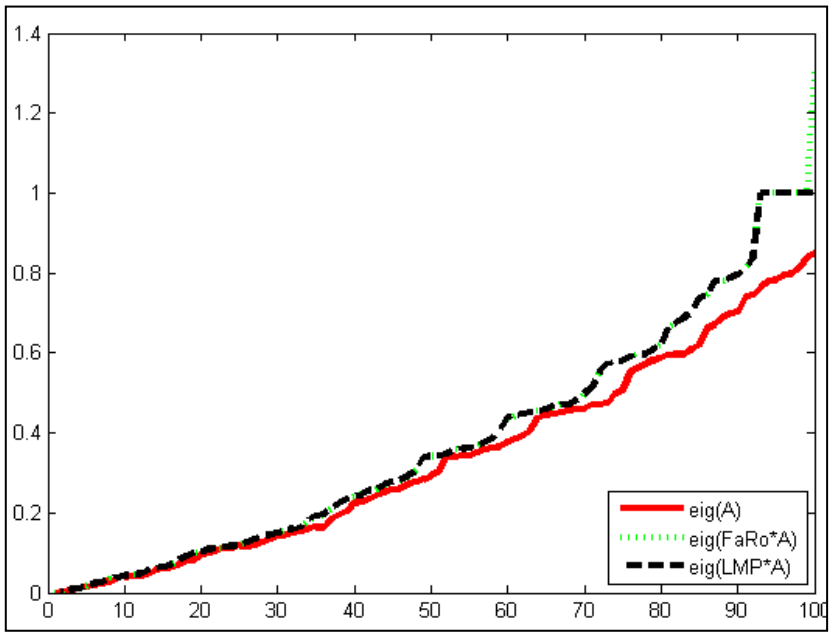


Fig. 7 Grafico degli autovalori, m = 8 iterazioni del GC (Caso matrice sparsa NOS4 100x100)

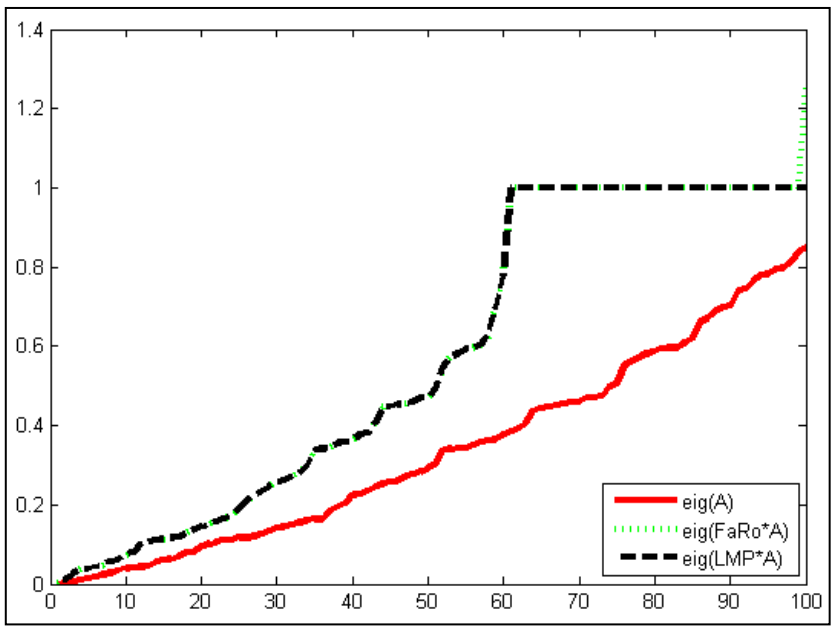


Fig. 8 Grafico degli autovalori, m = 40 iterazioni del GC (Caso matrice sparsa NOS4 100x100)

Questa differenza tra i risultati ottenuti dal punto di vista del numero di condizionamento e dal punto di vista degli autovalori è spiegata dal fatto che questi due precondizionatori considerati (LMP e FaRo) invece di mirare esplicitamente a diminuire il numero di condizionamento della matrice di partenza, mirano più che altro a far sì che un certo numero di autovalori della matrice precondizionata sia uguale al valore 1. E' importante infatti avere un buon numero di autovalori clusterizzati, perché con questo requisito il Gradiente Coniugato, applicato alla soluzione di un sistema lineare *precondizionato,* converge più velocemente alla soluzione, riducendo così i tempi di calcolo.

L'importanza di questi due tipi di precondizionatori è che, oltre ad essere in letteratura gli unici precondizionatori utilizzabili per sistemi a grandi dimensioni, sono molto adatti per risolvere una sequenza di sistemi lineari del tipo $A_k x = b_k$ in cui molto spesso $A_k \approx A_{k+1}$. In questo caso se si volesse calcolare un precondizionatore per ogni sistema lineare si avrebbe *k* volte il tempo di calcolo del precondizionatore. Sia il precondizionatore proposto da Gratton, Sartenaer e Tshimanga che quello di Fasano e Roma riescono a risolvere questo tipo di problema. In pratica per affrontare una sequenza di sistemi lineari del tipo $A_k x = b_k$, si applica il metodo del gradiente coniugato al primo sistema lineare non precondizionato $A_1 x = b_1$ e si costruirà dalle informazioni ricavate il precondizionatore, il quale verrà poi usato per risolvere il resto dei sistemi riducendo così il tempo e costo computazionale in generale.

Cap.6 - Conclusioni

L'obiettivo della tesi è stato quello di analizzare e studiare una particolare tecnica di risoluzione di sistemi lineari simmetrici e definiti positivi. Questa tecnica è la cosiddetta tecnica del *precondizionamento*, la più adatta per sistemi di grandi dimensioni e sparsi.

Come è detto nei capitoli precedenti qualunque problema che ci si trovi ad affrontare nella vita di tutti i giorni che sia di natura scientifica, economica, aziendale o ingegneristica può essere spesso rappresentato e risolto attraverso modelli matematici basati su semplici sistemi di equazioni lineari. L'unica difficoltà che ci si troverà ad affrontare con questi sistemi è quella della scelta dell'algoritmo più adatto per risolverli il più velocemente possibile in termini di tempo e costo computazionale.
Esiste una vasta gamma di questi metodi di risoluzione. Sostanzialmente questi si suddividono in due grandi rami: i metodi diretti ed i metodi iterativi.
Si è imparato nell'analisi di questi particolari algoritmi che per scegliere l'algoritmo più adatto ed efficiente è molto importante conoscere le caratteristiche di base del sistema come sparsità, numero e tipo delle equazioni.

I metodi diretti che si è andati ad analizzare sono il cosiddetto *metodo di eliminazione di Gauss* e la sua forma avanzata, ovvero la *fattorizzazione LU*. L'obiettivo di questi metodi è quello di ottenere direttamente (senza approssimazioni intermedie) la soluzione di un sistema lineare in un numero finito di passi (operazioni). Tutti e due i metodi diretti visti nella tesi, in pratica cercano di trasformare il sistema lineare di partenza $Ax = b$ in un sistema equivalente $A'x = b'$ di più facile risoluzione. Dato un sistema lineare $Ax = b$, il presupposto è quello di trasformare la matrice dei coefficienti $A \epsilon R^{nxn}$ in una matrice triangolare inferiore (o superiore) , ossia tale che tutti gli elementi sopra (sotto) la diagonale principale siano nulli. In questo modo verranno ridotte drasticamente le operazioni da eseguire per risolvere il sistema lineare in esame. E' emerso dallo studio di questi metodi che siano molto efficienti nel caso ci si trova a risolvere sistemi lineari con matrici dense (ovvero con pochi elementi nulli) e di non elevate dimensioni. Il loro svantaggio più significativo è che purtroppo non riescono a mantenere inalterate le caratteristiche di sparsità della matrice di partenza alterandone in maniera evidente il contenuto.

I metodi iterativi hanno invece un comportamento diverso, infatti il loro procedimento per risolvere un sistema lineare $Ax = b$, costruisce a partire da un vettore iniziale una successione di vettori che sotto opportune condizioni convergerà alla soluzione cercata. Dato che non si otterrà in questi casi (in genere) la soluzione esatta in un numero finito di operazioni, si è visto che è necessario individuare criteri di arresto per la successione di vettori che dovrà convergere alla soluzione. I più usati nella letteratura matematica sono il criterio sul residuo del sistema lineare e sul numero massimo di iterazioni.

Questi metodi sono molto adatti nel caso ci si trovi ad affrontare sistemi sparsi (ovvero con molti elementi nulli) e di grandi dimensioni ($n \approx 10^4 \to 10^8$). Il motivo è dovuto al fatto che questi algoritmi cercano a differenza dei precedenti di non alterare il contenuto della matrice iniziale del sistema da risolvere. Inoltre questi metodi, sfruttando il fatto che lavorano con matrici contenenti molti elementi nulli, non richiedono la memorizzazione effettiva della matrice di partenza. Sono anche meno onerosi dal punto di vista computazionale dato che il loro costo computazionale per iterazione è $O(n^2)$, che è molto inferiore (supponendo il numero di iterazioni $\ll n$) a quello dei metodi diretti per risolvere un sistema lineare, che è $O(n^3)$.

I metodi iterativi descritti nella tesi nel capitolo 2 sono basati sostanzialmente sulla decomposizione della matrice dei coefficienti *A*, ovvero quello di Jacobi, quello di Gauss-Seidel e infine il metodo del rilassamento.
E' stato dedicato un intero capitolo, il terzo, ad un particolare metodo iterativo: il metodo del Gradiente Coniugato proposto da Hestenes e Stiefel nel 1950. Questo metodo afferma che risolvere un sistema lineare in cui la matrice dei coefficienti è *definita positiva* equivale a minimizzare una forma quadratica ad esso relativa. Come è stato spiegato nella tesi questo procedimento parte da un punto iniziale per poi scegliere opportune *direzioni di discesa* per la forma quadratica, lungo le quali muoversi per avvicinarsi il più possibile alla soluzione del sistema lineare preso in considerazione. La prima direzione di discesa scelta per definizione coincide con l'antigradiente, che a sua volta è coincidente con il residuo iniziale. Le successive direzioni di discesa devono possedere un'importante proprietà: la coniugatezza (Equazione 3.27.).

Il metodo del gradiente coniugato inoltre fa parte dei cosiddetti *metodi di Krylov*, in quanto ad ogni iterazione k del metodo ogni direzione di ricerca $d^{(k)}$ e ogni residuo $r^{(k)}$è contenuto nel *sottospazio di Krylov* $\mathcal{K}\left(r^{(0)}; k\right) \stackrel{\text{def}}{=} \text{span}\left\{r^{(0)}, Ar^{(0)}, \dots, A^k r^{(0)}\right\}$, dove andrà a cercare la soluzione corrente migliore del sistema lineare simmetrico che si intende risolvere.
Nella risoluzione di un generico sistema lineare simmetrico $Ax = b$, tramite questo specifico algoritmo iterativo, si è osservato che è molto importante l'analisi di un particolare numero: il *numero di condizionamento* $\mathcal{K}(\text{A})$ della matrice dei coefficienti *A*. Il numero di condizionamento, che corrisponde al rapporto tra l'autovalore massimo e l'autovalore minimo della matrice *A*, serve per capire con che velocità il metodo del gradiente coniugato, ma anche qualsiasi altro metodo iterativo, converge alla soluzione del sistema lineare $Ax = b$ in esame.

Per far sì che i metodi iterativi basati sui sottospazi di Krylov (come il Gradiente Coniugato) trovino il più velocemente possibile la soluzione del sistema lineare in esame esiste la cosiddetta tecnica del *precondizionamento.* Questa tecnica consiste nel prendere una matrice $M \in R^{nxn}$, chiamata *matrice di precondizionamento* o *precondizionatore*, avente determinati requisiti e trasformare un sistema lineare

simmetrico $Ax = b$ nell'equivalente sistema precondizionato $M^{-1}Ax = M^{-1}b$ di più facile risoluzione.

Inoltre un'altra importante proprietà di questa tecnica è che fa sì che gli autovalori della matrice precondizionata $M^{-1}A$ siano meglio clusterizzati, ovvero il più vicino tra loro, della matrice dei coefficienti di partenza A, in modo che $\Lambda(M^{-1}A) = \{(1, \dots, 1)\}$.

I precondizionatori che si è voluto studiare e testare in questa tesi sono gli unici in letteratura adatti a risolvere sistemi lineari simmetrici a grandi dimensioni e sono *Il Limited Memory Preconditioners (LMP)* di Gratton, Sartenaer e Tshimanga e il precondizionatore proposto da G.Fasano e M.Roma.
Si è notato che la strategia che è stata usata per la loro creazione è la medesima. Dato un sistema lineare $Ax = b$, di dimensione n molto grande, per prima cosa vengono eseguite un certo numero $m \ll n$ di iterazioni di Gradiente Coniugato "non precondizionato". Successivamente da queste iterazioni viene costruito il precondizionatore e le matrici che lo compongono.
I test per confrontare i due precondizionatori sono stati effettuati su una serie di matrici di diversa dimensione e caratteristiche prese dal sito *Matrix Market* tramite il software matematico *Matlab*. Sapendo che un buon precondizionatore è definito tale se riesce a migliorare il più possibile il numero di condizionamento della matrice dei coefficienti iniziale, si è notato dai risultati ottenuti nel capitolo 5 che purtroppo questo non avveniva in maniera evidente. Tra i due precondizionatori quello che risulta meno peggiore da questo punto di vista, nei test effettuati, è quello LMP proposto da Gratton, Sartenaer e Tshimanga.

Andando a vedere invece come si clusterizzano gli autovalori della matrice precondizionata, si sono ottenuti risultati più soddisfacenti rispetto al confronto dal punto di vista del numero di condizionamento. Un buon precondizionatore è definito tale se riesce anche a clusterizzare il maggior numero di suoi autovalori ad 1. E' importante avere il maggior numero di autovalori clusterizzati perché in questo modo quando si andrà ad applicare il metodo del Gradiente Coniugato al sistema lineare precondizionato, con il precondizionatore scelto si troverà in maniera più veloce la soluzione. Questo è dovuto al fatto che le direzioni di discesa create dal metodo del Gradiente Coniugato sono direzioni che a volte si avvicinano agli autovettori associati agli autovalori della matrice precondizionata. In pratica il Gradiente Coniugato riesce a trovare una buona approssimazione della soluzione di un sistema lineare se riesce a generare il sottospazio di Krylov nel sottospazio degli autovettori associati a tutti gli autovalori uguali ad 1.
Questa difformità tra i risultati ottenuti dal punto di vista del numero di condizionamento e dal punto di vista degli autovalori è spiegata dal fatto che questi due precondizionatori considerati (LMP e FaRo) invece di mirare esplicitamente a diminuire il numero di condizionamento della matrice di partenza, mirano più che

altro a stabilire che un certo numero di autovalori della matrice precondizionata siano uguali al valore 1.
I due precondizionatori da questo punto di vista sembrano comportarsi nello stesso modo. Tutti e due sono ottenuti memorizzando solo alcuni vettori, senza calcolare il prodotto di matrici o matrici inverse. Quello proposto da Fasano e Roma è meno costoso dal punto di vista computazionale ed inoltre può essere usato, a differenza del LMP (proposto da Gratton, Sartenaer e Tshimanga), per sistemi lineari indefiniti.

La clusterizzazione comunque non può essere solo sufficiente se non si riesce a diminuire il numero di condizionamento. Si è capito per questo motivo che questi precondizionatori non possono essere usati in qualunque sistema lineare che ci si trovi ad affrontare. Dato che questi precondizionatori sfruttano al massimo le informazioni che hanno a loro disposizione, risultano molto adatti nella risoluzione di *classi di problemi* che abbiano determinate caratteristiche.
Infatti il precondizionatore proposto da Fasano e Roma studiato e testato in questa tesi può essere esteso ad una classe di precondizionatori più grande, sempre proposta dagli stessi, descritta nel loro articolo "*AINVK: a Class of Approximate Inverse Preconditioners based on Krylov-subspace methods, for Large Indefinite Linear Systems*"[4]. Questa classe di precondizionatori è stata usata per risolvere problemi di grandi dimensioni, di ottimizzazione non convessa, all'interno dei quali compaiono sequenze di sistemi lineari indefiniti.

Appendice

A.1. Algoritmi

Algoritmo del Gradiente Coniugato

Input:
$x^{(0)}, A, b, maxiter\ (n°\ massimo\ di\ iterazioni), tol(tolleranza\ sulla\ norma\ del\ residuo)$

$r^{(0)} = b - Ax^{(0)}$
$p^{(0)} = r^{(0)}$
$k = 0$
WHILE $\left(\left\|r^{(k)}\right\| > tol\right)$ AND $(k < maxiter)$ DO

1. $s = Ap^{(k)}$
2. $u = \left(p^{(k)}\right)^T s$
3. $\alpha_k = \frac{\left(p^{(k)}\right)^T r^{(k)}}{u}$
4. $x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)}$
5. $r^{(k+1)} = r^{(k)} - \alpha_k s$
6. $\beta_k = \frac{\left(r^{(k+1)}\right)^T r^{(k+1)}}{\left(r^{(k)}\right)^T r^{(k)}}$
7. $p^{(k+1)} = r^{(k+1)} + \beta_k p^{(k)}$
8. $k = k + 1$

END

Algoritmo del Gradiente Coniugato Precondizionato

Input: $x^{(0)}, A, b, M, maxiter, tol$

$r^{(0)} = b - Ax^{(0)}$
$z^{(0)} = M^{-1}r^{(0)}$
$p^{(0)} = z^{(0)}$
$k = 0$
WHILE $\left(\left\|r^{(k)}\right\| > tol\right)$ AND $(k < maxiter)$ DO

1. $s = Ap^{(k)}$
2. $u = \left(p^{(k)}\right)^T s$
3. $\alpha_k = \frac{\left(p^{(k)}\right)^T r^{(k)}}{u}$
4. $x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)}$
5. $r^{(k+1)} = r^{(k)} - \alpha_k s$
6. $z^{(k+1)} = M^{-1}r^{(k+1)}$
7. $\beta_k = \frac{\left(r^{(k+1)}\right)^T z^{(k+1)}}{\left(r^{(k)}\right)^T z^{(k)}}$
8. $p^{(k+1)} = z^{(k+1)} + \beta_k p^{(k)}$
9. $k = k + 1$

END

A.2. Tabelle numero condizionamento delle matrici testate nel cap. 5

Matrice NOS5 468x468

	m = 5	**m = 10**	**m = 15**	**m = 20**	**m = 30**	**m = 40**
A	$1.1003*10^4$					
FaRo*A	$2.8226*10^7$	$2.3342*10^7$	$3.2682*10^7$	$1.8306*10^7$	$2.3759*10^7$	$3.3333*10^7$
LMP*A	$5.8172*10^5$	$5.7535*10^5$	$5.2787*10^5$	$5.0904*10^5$	$5.0862*10^5$	$4.9321*10^5$

Tab. 1 Matrice sparsa NOS5 468x468 - Numero di condizionamento ad ogni range di iterazione dei tre casi (Matrice di partenza, Matrice precondizionata con il precondizionatore LMP e matrice precondizionata con il precondizionatore costruito da Fasano e Roma).

Matrice BCCSSTK27 1224X1224

	m = 20	**m = 40**	**m = 80**	**m = 160**	**m = 320**	**m = 640**
A	$2.4136*10^{4}$					
FaRo*A	$1.5336*10^{8}$	$1.0567*10^{8}$	$0.8476*10^{8}$	$0.9284*10^{8}$	$2.4270*10^{8}$	$9.9438*10^{8}$
LMP*A	$0.3097*10^{7}$	$1.4982*10^{7}$	$0.3598*10^{7}$	$1.5645*10^{7}$	$4.8248*10^{7}$	$6.9483*10^{7}$

Tab. 2 Matrice sparsa BCCSSTK27 1224X1224 - Numero di condizionamento ad ogni range di iterazione dei tre casi (Matrice di partenza, Matrice precondizionata con il precondizionatore LMP e matrice precondizionata con il precondizionatore costruito da Fasano e Roma).

Matrice NOS1 237X237

	m = 4	**m = 8**	**m = 12**	**m = 16**	**m = 20**	**m = 40**
A	$1.9915*10^{7}$					
FaRo*A	$0.5185*10^{13}$	$0.2268*10^{13}$	$0.6230*10^{13}$	$1.2069*10^{13}$	$1.1193*10^{13}$	$0.3266*10^{13}$
LMP*A	$2.4551*10^{9}$	$2.4550*10^{9}$	$2.4549*10^{9}$	$2.4547*10^{9}$	$2.4543*10^{9}$	$2.4470*10^{9}$

Tab. 3 Matrice banded NOS1 237x237 - Numero di condizionamento ad ogni range di iterazione dei tre casi (Matrice di partenza, Matrice precondizionata con il precondizionatore LMP e matrice precondizionata con il precondizionatore costruito da Fasano e Roma).

Matrice NOS6 675x675

	m = 5	**m = 10**	**m = 15**	**m = 20**	**m = 30**	**m = 40**
A	$7.6505*10^{6}$					
FaRo*A	$6.8984*10^{10}$	$1.2178*10^{10}$	$0.3416*10^{10}$	$0.1859*10^{10}$	$0.0809*10^{10}$	$0.6238*10^{10}$
LMP*A	$7.6482*10^{6}$	$7.6138*10^{6}$	$7.4806*10^{6}$	$7.4664*10^{6}$	$6.8976*10^{6}$	$6.7054*10^{6}$

Tab. 4 Matrice banded NOS6 675x675 - Numero di condizionamento ad ogni range di iterazione dei tre casi (Matrice di partenza, Matrice precondizionata con il precondizionatore LMP e matrice precondizionata con il precondizionatore costruito da Fasano e Roma).

Matrice BCCSSTK14 1806X1806

	m = 20	**m = 40**	**m = 80**	**m = 160**	**m = 320**	**m = 640**
A	$1.1923 *10^{10}$					
FaRo*A	$2.3833*10^{16}$	$1.7038*10^{16}$	$0.7380*10^{16}$	$2.3059*10^{16}$	$2.3272*10^{16}$	$3.6991*10^{16}$
LMP*A	$0.0120*10^{12}$	$0.0091*10^{12}$	$0.0148*10^{12}$	$0.0627*10^{12}$	$1.2695*10^{12}$	$0.4356*10^{12}$

Tab. 5 Matrice banded BCCSSTK14 1806x1806 - Numero di condizionamento ad ogni range di iterazione dei tre casi (Matrice di partenza, Matrice precondizionata con il precondizionatore LMP e matrice precondizionata con il precondizionatore costruito da Fasano e Roma).

Bibliografia

[1] Benzi M. (2002), *Preconditioning techniques for large linear systems: a survey*, Journal of Computational Physics, 182(2), 418-477.

[2] Bevilacqua R. (1987), *Introduzione alla matematica computazionale*, Zanichelli.

[3] Conn A.R., Gould N. & Toint P. (2000), *Trust Region Methods*, MPS/SIAM Series on Optimization.

[4] G.Fasano & M.Roma, *AINVK: a Class of Approximate Inverse Preconditioners based on Krylov-subspace methods, for Large Indefinite Linear Systems*, Optimization Online.

[5] G.Fasano & M.Roma (2013), *Preconditioning Newton–Krylov methods in nonconvex large scale optimization*, accettato per la pubblicazione su Computational Optimization and Applications.

[6] Golub, G. H., & Van Loan, C. F. (1996), *Matrix computations (Vol. 3)*, Johns Hopkins University Press.

[7] S. Gratton, A. Sartenaer, & J. Tshimanga (2009), *On a class of limited memory preconditioners for large scale linear systems with multiple right-hand sides*, SIAM Journal on Optimization, 21:912–935.

[8] Grippo L. & Sciandrone M. (2011), *Metodi di ottimizzazione non vincolata*, Springer.

[9] Lipschutz S. (1992), *Algebra lineare: Seconda edizione*, Etas Libri.

[10] Nocedal, J., & Wright, S. J. (1999), *Numerical optimization*, Springer Verlag.

[11] Quarteroni A., Sacco R., & Saleri F. (2008), *Matematica numerica*, Springer Verlag.

[12] Robbiano L. (2006), *Algebra lineare per tutti*, Springer.

[13] Rodriguez G. (2003), *Dispense (parziali) del corso Metodi iterativi per la risoluzione di modelli differenziali lineari e debolmente lineari*, Dipartimento di Matematica, Università degli studi di Cagliari.

[14] Saad Y. (1996), *Iterative methods for sparse linear system*, PWS Publishing Company.

[15] Shewchuk J. R. (1994), *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*, School of Compute Science, Carnegie Mellon University, Pittsburgh.

[16] Trefethen L. N. & Bau III D. (1997), *Numerical linear algebra* (No. 50), SIAM Society for Industrial and Applied Mathematics, 313-318.