UNIVERSITÀ DEGLI STUDI DI CAGLIARI

Facoltà di Scienze

Corso di Laurea Magistrale in Matematica

Metodi di proiezione per la risoluzione numerica di sistemi lineari

Relatore
Prof. Giuseppe Rodriguez

Tesi di Laurea di
Andrea Corongiu

Anno Accademico 2019-2020

Indice

Capitolo 1

Problema ai minimi quadrati

1.1 Formulazione del problema

In moltissimi casi si incontrano sistemi lineari dove il numero delle equazioni differisce dal numero delle incognite, che tradotto in termini matematici prende la forma di

$$Ax = b \tag{1.1}$$

con $A \in \mathbb{R}^{m \times n}$ con $m \neq n$, $b \in \mathbb{R}^m$ e $x \in \mathbb{R}^n$. Partendo dal presupposto che la matrice A sia a rango pieno, possiamo avere due casi : $m > n$ dove il problema potrebbe non avere soluzione e in questo caso il sistema viene denominato *sovradeterminato*; oppure $m < n$, qui si inciamperebbe in infinite soluzioni e viene nominato *sottodeterminato*.
Questi tipi di problemi sono detti *mal posti*[1] e non hanno quindi una soluzione nel senso classico. Prendiamo in esame il caso $m > n$ questo significa che $m - n$ equazioni sono linearmente dipendenti dalle altre, e tipicamente non ci è dato sapere quali. Nel caso in cui i dati siano affetti da errori, potrebbe capitare lo spiacevole inconveniente che le equazioni non siano verificate esattamente.
Visto che il problema è mal posto, ci piacerebbe riformularlo per arrivare ad uno equivalente ben posto. Se tutte le equazioni non possono essere verificate contemporaneamente, è ragionevole richiedere che la varianza tra il primo e il secondo membro sia minima. Si arriva così ad un *problema ai minimi quadrati* che possiamo esprimere nella forma

$$\min_x \|Ax - b\|_2 \tag{1.2}$$

[1] la definizione si vedrà meglio nel paragrafo sui problemi inversi, per ora ci basti sapere che un problema viene detto *ben posto* se ha una sola soluzione che dipende con continuità dai dati. Mal posto risulta il caso contrario.

Se il minimo risulta essere zero allora vuol dire che il sistema avrà una soluzione in senso classico, altrimenti otterremo la nostra soluzione ai minimi quadrati.
Dal momento che si vuole minimizzare una quantità non negativa, possiamo considerare il quadrato della norma del residuo e riscrivere la formula in questo modo

$$\|Ax - b\|_2^2 = x^T A^T Ax - 2x^T A^T b + b^T b.$$

Per minimizzare questa quantità dobbiamo richiedere l'annullarsi del gradiente

$$\frac{1}{2}\nabla\big(\|Ax - b\|_2^2\big) = A^T Ax - A^T b = 0$$

arrivando così al *sistema normale*

$$A^T Ax = A^T b \tag{1.3}$$

Se $\text{rank}(A) = n$ allora la matrice $A^T A \in \mathbb{R}^{n\times n}$ è invertibile e la soluzione del sistema è unica e della forma

$$x_{LS} = (A^T A)^{-1} A^T b \tag{1.4}$$

Se $\text{rank}(A) < n$ la matrice $A^T A$ risulta singolare ma il problema risulta comunque consistente per via del fatto che il vettore $A^T b$ appartiene all'immagine di A^T, che risulta coincidente con quella di $A^T A$. In questo caso comunque non rimaniamo spaesati perché tra gli infiniti vettori che soddisfano il sistema in genere si prende come soluzione x_{LS} quella di norma minima.

1.2 SVD

In questa sezione approfondiremo una decomposizione matriciale particolare che viene utilizzata a partire dal paragrafo sui problemi inversi. La Singular Value Decomposition (SVD) è una fattorizzazione matriciale che permette di rappresentare qualunque matrice come il prodotto di due matrici ortogonali e una diagonale.

Teorema 1.1. *Sia $A \in \mathbb{R}^{m\times n}$ con $m \geq n$. Allora la SVD di A è la decomposizione*

$$A = U\Sigma V^T \tag{1.5}$$

dove $U = (u_1, \ldots, u_m) \in \mathbb{R}^{m\times m}$ e $V = (v_1, \ldots, v_n) \in \mathbb{R}^{n\times n}$ sono matrici con colonne ortonormali, $U^T U = I_m$ e $V^T V = I_n$, mentre Σ è la matrice

diag$(\sigma_1, \ldots, \sigma_n) \in \mathbb{R}^{m\times n}$*, che ha come entrate elementi non negativi tali che*

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$$

I valori σ_i *vengono denominati valori singolari di* A*.*
I vettori u_i, v_i *sono chiamati rispettivamente vettori singolari sinistri e destri.*

$$A = \big(u_1, \ldots, u_m\big) \begin{pmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ . & . & . & . \\ & & & \sigma_n \\ 0 & . & . & 0 \end{pmatrix} \begin{pmatrix} v_1^T \\ . \\ . \\ v_n^T \end{pmatrix} \longleftrightarrow A = \sum_{i=1}^{n} \sigma_i u_i v_i^T$$

Osservazione 1.1.1. Si può notare che la sommatoria si ferma n e non compare m, questo per via del fatto che $m \geq n$ quindi abbiamo $m - n$ righe di zeri nella matrice Σ.

Osservazione 1.1.2. La SVD è definita per ogni m e n, se $m < n$ si applica la (1.5) a A^T e si interscambiano U, V.

Dal punto di vista dell'algebra lineare si può intuire che la fattorizzazione SVD di A fornisce due basi ortonormali (le colonne di U e V) tali che A diventi diagonale quando viene trasformata attraverso queste due basi.
Di seguito elenchiamo alcune proprietà che si possono dedurre per la decomposizione SVD:

- $AV = U\Sigma \longrightarrow Av_i = \sigma_i u_i$;
- $A^T U = V\Sigma^T \longrightarrow A^T u_i = \sigma_i v_i$
- $\kappa_2(A) = \frac{\sigma_1}{\sigma_n}$ (Assumendo che $\text{rank}(A) = n$)
- $\|A\|_2 = \sigma_1$
- Sia $b \in \mathbb{R}^m$ e $x \in \mathbb{R}^n$ t.c. $b = Ax \Rightarrow b = \sum_{i=1}^{n} \sigma_i (v_i^T x) u_i$
- Se $A \in \mathbb{R}^{n\times n}$ è non singolare $\Rightarrow A^{-1} = V\Sigma^{-1}U^T = \sum_{i=1}^{n} \sigma_i^{-1} v_i u_i^T$
- Se $A \in \mathbb{R}^{n\times n}$ e $b \in \mathbb{R}^n \Rightarrow A^{-1}b = \sum_{i=1}^{n} \sigma_i^{-1} (u_i^T x) v_i$
- se $\text{rank}(A) = r < n$ abbiamo:
  - $\text{rank}(A) = r \Longleftrightarrow \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_n = 0$
  - $\mathcal{N}(A) = \text{span}(v_{r+1}, \ldots, v_n)$ e $\mathcal{R}(A) = \text{span}(u_1, \ldots, u_r)$

$$- \; A = \sum_{i=1}^{r} \sigma_i u_i v_i^T \quad \longleftrightarrow \quad A = U_r \Sigma_r V_r^T$$

I valori singolari σ_i sono sempre ben condizionati rispetto alle perturbazioni della matrice A, infatti se A è perturbata da una certa matrice E, allora la norma $\|E\|_2$ è un limite superiore per la perturbazione assoluta di ogni valore singolare.
Dalle relazioni $A^TA = V\Sigma^2V^T$ e $AA^T = U\Sigma^2U^T$ si vede che SVD è connessa alla decomposizione agli autovalori di una matrice simmetrica e semidefinita A^TA e AA^T. Questo dimostra che la SVD è unica per una data matrice A, a meno di un cambio di segno della coppia (u_i, v_i), eccetto per i vettori singolari associati con valori singolari multipli.
Relativamente ai problemi discreti mal posti si riscontrano spesso due caratteristiche. La prima è che i valori singolari decadono gradualmente a 0. Con l'aumentare della dimensione della matrice incrementerà il numero di valori singolari piccoli. La seconda caratteristica è che i vettori singolari sinistro e destro, u_i v_i, tendono ad avere più cambi di segno nei loro elementi all'aumentare dell'indice i.

Teorema 1.2. *Sia $A \in \mathbb{R}^{m\times n}$ tale che rank$(A) = r$ e sia $A = U\Sigma V^T$. Assumiamo $k \le r$ e definiamo la SVD troncata*

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

Allora $A_k = \underset{rank(B)=k}{\operatorname{argmin}} \|B - A\|_2$. Quindi $\|A - A_k\|_2 = \sigma_{k+1}$

1.2.1 Applicazione al problema dei minimi quadrati

Un risvolto interessante della decomposizione SVD di una matrice A è sicuramente legato al problema dei minimi quadrati

$$\min_x \|Ax - b\|_2^2. \tag{1.6}$$

Sia $A \in \mathbb{R}^{m\times n}$ con $m \ge n$ =rank(A) e utilizziamo la sua SVD

$$\min_x \|Ax - b\|_2^2 = \min_x \|U\Sigma V^T x - b\|_2^2 = \min_x \|\Sigma V^T x - U^T b\|_2^2 = \min_x \|\Sigma y - U^T b\|_2^2$$

Ponendo $y = V^Tx$ e supponendo che rank$(A) = n$ abbiamo

$$\min_x \|Ax - b\|_2^2 = \min_x \|\Sigma y - U^T b\|_2^2 = \sum_{i=1}^{n} (\sigma_i y_i - u_i^T b)^2 + \sum_{i=n+1}^{m} (u_i^T b)^2$$

Dal fatto che A sia a rango pieno abbiamo tutti i $\sigma_i > 0$ e

$$y_i = \frac{u_i^T b}{\sigma_i} \qquad \forall 1 \le i \le n \tag{1.7}$$

che fornisce la soluzione $x = Vy$. Se $\sigma_i = 0$ allora y_i è indeterminata e perciò la soluzione ai minimi quadrati non è unica.
Supponendo che il rango di A sia $r < n$, allora abbiamo che $\sigma_j = 0$ per $j \in \{r+1, \dots, n\}$, perciò la (1.7) vale per $i = 1, \dots, r$ mentre per $i > r$ possiamo scegliere y_i arbitrari. Optando per la scelta migliore, $y_i = 0$ per $i > r$, otteniamo che la soluzione di minima norma del problema ai minimi quadrati è

$$x_{LS} = \underset{x}{\operatorname{argmin}} \|Ax - b\|_2^2 = \sum_{i=1}^{r} \frac{u_i^T b}{\sigma_i} v_i \tag{1.8}$$

Utilizzando la notazione matriciale possiamo scriverla come $x_{LS} = A^\dagger b$, dove con $A^\dagger \in \mathbb{R}^{n \times m}$ si intende la *matrice pseudoinversa o di Moore-Penrose* della forma $A^\dagger = V\Sigma^\dagger U^T$ con $\Sigma^\dagger = \operatorname{diag}(\sigma_1^{-1}, \dots, \sigma_r^{-1}, 0, \dots, 0)$.

Ora consideriamo il caso in cui il nostro termine noto b sia affetto da errore e quindi sia della forma

$$b_{err} = b + \eta$$

dove con η indichiamo l'errore. Allora possiamo riscrivere la soluzione x_* nella forma

$$x_* = A^\dagger b_{err} = \sum_{i=1}^{r} \frac{u_i^T b_{err}}{\sigma_i} v_i = \sum_{i=1}^{r} \frac{u_i^T b}{\sigma_i} v_i + \sum_{i=1}^{r} \frac{u_i^T \eta}{\sigma_i} v_i = x_{LS} + \sum_{i=1}^{r} \frac{u_i^T \eta}{\sigma_i} v_i \tag{1.9}$$

L'ultima espressione ricavata nella (1.9) verrà ripresa nel paragrafo successivo quando andremo a trattare la risoluzione dei problemi inversi.

1.3 Problemi Inversi

In questo capitolo spiegheremo cosa si intende per problemi inversi, dando anche qualche esempio. Non c'è in generale una definizione formale di problema inverso ma generalmente questi problemi sorgono quando si vuole calcolare un informazione su dati interni o altrimenti nascosti, a partire da misurazioni prese esternamente. In poche e schematiche parole, seguendo lo schema Input ⇒ Sistema ⇒ Output, noi siamo a conoscenza sicuramente dell' Output, che però è affetto da errori, e uno solo tra input e sistema. Un esempio di problema inverso è la ricostruzione di immagini, dove si parte da un'immagine distorta e si vuole avere un'immagine più nitida.
I problemi inversi più interessanti e/o difficili da risolvere (spesso le due cose coincidono) sono problemi mal posti. Per definire un problema mal posto, partiamo dalla definizione di problema ben posto, coniata da Hadamard.

Definizione 1.1. Un problema è detto *ben posto* se rispetta le seguenti tre condizioni:

- Esistenza: il problema deve avere una soluzione;
- Unicità: Ci deve essere un'unica soluzione del problema;
- Stabilità: La soluzione deve dipendere con continuità dai dati.

La condizione di esistenza può risultare scontata e banale ma in realtà non lo è. Possiamo facilmente formulare problemi che non hanno soluzione.

Esempio 1.2.1. *Per esempio si può considerare il seguente problema sovradeterminato*

$$\begin{pmatrix}1\\2\end{pmatrix}x = \begin{pmatrix}1\\2.2\end{pmatrix}$$

La condizione di unicità è leggermente più critica, ma può esser spesso aggiustata riformulando il problema, tipicamente aggiungendo dei vincoli o condizioni per la soluzione.
Per quanto riguarda la stabilità, il caso è un pò più difficile perché una violazione di questa regola implica che piccole perturbazioni arbitrarie dei dati possono produrre larghe perturbazioni della soluzione. Nel caso del sistema classico lineare $Ax = b$, che sarà il nostro caso, il problema è causato dal fatto che la matrice A è mal condizionata.
In sostanza, se una sola delle condizioni di Hadamard non viene rispettata, allora si dirà che il problema è *mal posto*. Il trattamento di sistemi lineari mal posti di equazioni è ovviamente più complicato rispetto a quelli ben posti. Chi si interfaccia con questo tipi di problemi deve sapere che tipo di mal condizionamento aspettarsi in modo da poter agire su di essi e inoltre aver conoscenza dei metodi numerici di regolarizzazione da usare per risolvere il problema efficientemente.
Ogni discussione relativa a matrici mal condizionate richiede la conoscenza della decomposizione SVD della matrice A. In particolare il numero di condizionamento della matrice A è il rapporto tra il più grande e il più piccolo dei valori singolari di A. Ci sono due importanti classi di problemi che si possono considerare:

- *Problemi a rango non pieno* sono caratterizzati dal fatto che la matrice A ha un raggruppamento di piccoli valori singolari, e c'è una relativa buona differenza tra piccoli e grandi valori singolari. Già dall'algebra lineare sappiamo che questo implica una o più righe e colonne sono linearmente dipendenti. Quindi la matrice contiene informazioni ridondanti e la chiave del trattamento numerico è quella di estrarre le righe

e colonne linearmente indipendenti di A, per arrivare ad un problema ben condizionato.

- *Problemi discreti mal posti* sorgono dalla discretizzazione di problemi mal posti e di seguito ne parleremo. In questo caso i valori singolari di A, così come le componenti SVD della soluzione, decadono gradualmente a 0 e questo rende difficile determinare il rango numerico. Per questo tipo di problemi l'obiettivo è trovare un bilanciamento tra la norma del residuo e la dimensione della soluzione che corrisponde meglio agli errori nei dati e alle aspettative sulla soluzione calcolata.

Ora passiamo ad una descrizione più matematica. Siano $\mathcal{X}$ e $\mathcal{Y}$ spazi di Hilbert e $\mathcal{A}$: $\mathcal{X} \to \mathcal{Y}$ un operatore lineare compatto tra i due spazi di Hilbert. Con questi dati consideriamo l'equazione

$$\mathcal{A}x = b$$

dove $x \in \mathcal{X}$ e $b \in \mathcal{Y}$ e inoltre che $b \in \mathcal{R}(\mathcal{A})$. In generale non siamo mai a conoscenza di b ma rimane a nostra disposizione solo la sua versione affetta da errore. La versione errata di b la denoteremo con b_{err} e assume la seguente forma

$$b_{err} = b + \eta \quad .$$

Una considerazione che si può fare è che dal fatto che $\mathcal{A}$ è compatto allora si ha che i suoi valori singolari tendono a 0. Di conseguenza i valori singolari di $\mathcal{A}^\dagger$ tendono verso ∞. Quindi $\mathcal{A}^\dagger b_{err}$ è di solito una scarsa ricostruzione della soluzione desiderata $x = \mathcal{A}^\dagger b$. Discretizzando il problema inverso descritto sopra, arriviamo al sistema di equazioni lineari

$$Ax = b_{err} \tag{1.10}$$

dove A è mal condizionata, cioè i suoi valori singolari decadono a 0 molto velocemente. Problemi di questa forma sono spesso denominati *problemi inversi discreti mal posti.*
Consideriamo ora il problema inverso mal condizionato (1.10) dove $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$,$\eta \in \mathbb{R}^n$ ma assumendo che $\eta \notin \mathcal{R}(A)$. Idealmente noi vorremmo risolvere il problema ai minimi quadrati

$$x_{LS} = \underset{x}{\operatorname{argmin}} \|Ax - b\|_2^2 = A^\dagger b$$

ma non avendo a disposizione b ci dovremo accontentare di risolvere il pro-

blema

$$x_* = \underset{x}{\operatorname{argmin}} \|Ax - b_{err}\|_2^2 = A^\dagger b_{err} \tag{1.11}$$

Usando la decomposizione SVD per la matrice A otteniamo l'espressione per la soluzione approssimata x_* (come in (1.9))

$$x_* = x_{LS} + \sum_{\sigma_j > 0} \frac{u_j^T \eta}{\sigma_j} v_j$$

Siccome i valori singolari tendono a zero al crescere di j, allora la quantità $\dfrac{u_j^T \eta}{\sigma_j}$ può essere arbitrariamente grande, e di fatto far diventare x_* una soluzione poco rappresentativa di quella esatta x_{LS}.

La condizione di Picard spiega meglio il problema descritto qua sopra. Riprendiamo ora l'operatore $\mathcal{A}$ definito come sopra, e sia $\mathcal{A}^*$ il suo aggiunto con la proprietà di essere autoaggiunto (ovvero tale che $\langle \mathcal{A}x, y\rangle = \langle x, \mathcal{A}^* y\rangle$). Denotiamo (σ_i, v_i, u_i), con $i \in \mathbb{N}$, la SVE (Singular Value Expansion) di $\mathcal{A}$, dove $\{v_i\}$ è un sistema completo di autovettori per $\mathcal{A}^*\mathcal{A}$, dove $\{u_i\}$ è un sistema completo di autovettori per $\mathcal{A}\mathcal{A}^*$ e $\sigma_i \geq 0$ sono in ordine decrescente fino a 0, che è il suo punto di accumulazione. Date queste definizioni, possiamo scrivere le seguenti relazioni

$$\mathcal{A}x = \sum_{i=1}^{\infty} \sigma_i \langle x, v_i\rangle u_i, \text{ con } x \in \mathcal{X}$$
$$\mathcal{A}^* y = \sum_{i=1}^{\infty} \sigma_i \langle y, u_i\rangle v_i, \text{ con } y \in \mathcal{Y}$$

Definiamo inoltre l'inversa generalizzata $\mathcal{A}^\dagger : \mathcal{D}(\mathcal{A}^\dagger) \subseteq \mathcal{Y} \longrightarrow \mathcal{X}$, con

$$\mathcal{D}(\mathcal{A}^\dagger) = \{y \in \mathcal{Y} | \sum_{i;\sigma_i>0} \sigma_i^{-2} |\langle y, u_i\rangle|^2 < \infty\}$$

definita in questo modo $\mathcal{A}^\dagger y = \sum_{i;\sigma_i>0} \sigma_i^{-1} \langle y, u_i\rangle v_i$.

Se $y \in \mathcal{D}(\mathcal{A}^\dagger)$ allora si dice che y soddisfa la condizione di Picard.

Tornando all'equazione $\mathcal{A}x = b_{err}$, la soluzione approssimata x_* espressa sopra, può essere così riformulata

$$x_* = \sum_{i;\sigma_i>0} \frac{\langle b_{err}, u_i\rangle}{\sigma_i} v_i = x_{LS} + \sum_{i;\sigma_i>0} \frac{\langle \eta, u_i\rangle}{\sigma_i} v_i$$

Visto che η non è regolare, allora le proiezioni sugli u_i non convergono a zero , di conseguenza non è verificata la condizione di Picard per la solu-

zione affetta da errore. Assumendo che $\frac{\langle \eta, u_i \rangle}{\sigma_i}$ non converga a 0 per $i \to \infty$ possiamo avere che $\left\| \sum_{\sigma_i > 0} \frac{\langle u_i, \eta \rangle}{\sigma_i} v_i \right\|_2$ risulti molto grande

Osservazione 1.2.1. Questo può essere esteso intuitivamente anche al caso di dimensione finita. Anche se si può notare che, nel caso finito dimensionale, la condizione di Picard è sempre soddisfatta.

Capitolo 2

Metodi di proiezione in sottospazi di Krylov

2.1 Introduzione

Il nostro punto di partenza sarà risolvere il sistema lineare

$$Ax = b \tag{2.1}$$

con un metodo di proiezione. Tutti i metodi che andremo ad analizzare in questo capitolo si basano sulla scelta del sottospazio di proiezione, che in questo caso ricadrà sul *sottospazio di Krylov*.

Definizione 2.1. Siano A una matrice e $r_0 = b - Ax_0$, definiamo il *sottospazio di Krylov* come

$$\mathcal{K}_m(A, r_0) = span\{r_0, Ar_0, A^2r_0, \ldots, A^{m-1}r_0\}$$

Per comodità di notazione indicheremo con $\mathcal{K}_m$ il sottospazio di Krylov rispetto ad A e r_0.
Risolvendo il sistema (2.1) con un metodo di proiezione in un sottospazio di Krylov si ottiene così una soluzione approssimata x_m dallo spazio affine $x_0 + \mathcal{K}_m$ di dimensione m, con x_0 soluzione iniziale, grazie alla condizione di Petrov-Galerkin

$$b - Ax_m \perp \mathcal{L}_m,$$

dove $\mathcal{L}_m$ è un altro sottospazio di dimensione m. Differenti scelte del sottospazio $\mathcal{L}_m$ e i vari modi in cui un sistema è *precondizionato* ci porteranno a diverse versioni dei metodi sui sottospazi di Krylov.

Supponiamo per adesso che $x_0 = 0$ di conseguenza avremo $\mathcal{K}_m(A, b)$. Se i vettori $A^k b$, con $k = 0, \ldots, m-1$ risultassero essere tutti indipendenti avremmo l'identificazione $\mathcal{K}_m \equiv \mathbb{R}^m$. Se questa condizione non si verificasse, potremmo avere che per un dato $l < m$, il vettore $A^l b$ risulti linearmente dipendente rispetto a quelli precedenti. In sostanza, esisterebbero scalari non tutti nulli α_i t.c.

$$\sum_{i=0}^{l} \alpha_i A^i b = 0$$

Supponendo $\alpha_0 \neq 0$ è possibile esplicitare il vettore b dalla relazione di sopra, ottenendo

$$Ax_* = b = -\frac{1}{\alpha_0} \sum_{i=1}^{l} \alpha_i A^i b = A\left(-\frac{1}{\alpha_0} \sum_{i=1}^{l} \alpha_i A^{i-1} b\right)$$

dove x_* indica la soluzione esatta del sistema lineare. Da tutto ciò possiamo concludere che

$$x_* = -\frac{1}{\alpha_0} \sum_{i=1}^{l} \alpha_i A^{i-1} b \in \mathcal{K}_l$$

e ciò mostra che il metodo terminerebbe in l passi. Se cambiamo il nostro punto di vista e ponendo $l = m$, possiamo arrivare alla conclusione che l'approssimazione ottenuta attraverso il metodo di Krylov è della forma

$$A^{-1}b \approx x_m = x_0 + q_{m-1}(A)r_0$$

dove con q_{m-1} si intende un polinomio di grado $m-1$. Nel caso in cui $x_0 = 0$, l'approssimazione assume una forma ancor più semplice portandoci alla conclusione che $A^{-1}b$ è approssimata da $q_{m-1}(A)b$.
Andremo a trattare due classi di metodi di Krylov che si differenzieranno per la scelta del sottospazio $\mathcal{L}_m$. La prima classe di metodi sarà basata sulla semplice scelta $\mathcal{L}_m = \mathcal{K}_m$ oppure sulla variazione di residuo minimo $\mathcal{L}_m = A\mathcal{K}_m$. La seconda classe di metodi invece è basata su una leggera modifica del sottospazio di Krylov $\mathcal{L}_m = \mathcal{K}_m(A^T, r_0)$

2.2 Accenni teorici

L'idea delle tecniche di proiezione è quella di estrarre una soluzione approssimata del sistema lineare $Ax = b$ da un sottospazio di $\mathbb{R}^n$. Se $\mathcal{K}$ è il sottospazio delle approssimazioni candidate, o sottospazio di ricerca, e se m è la sua dimensione allora, in generale, m vincoli devono essere forniti affinché si trovi una qualche approssimazione. Un tipico modo per descrivere questi vincoli è quello di imporre m condizioni ortogonali indipendenti. In particolare, il vettore residuo,$b - Ax$, è vincolato ad essere ortogonale a m vettori linearmente

indipendenti. Questo definisce un altro sottospazio $\mathcal{L}$ di dimensione m, che verrà chiamato sottospazio dei vincoli. Ci sono due classi di metodi di proiezione: ortogonale e obliquo. Nel caso ortogonale il sottospazio $\mathcal{L}$ coincide con $\mathcal{K}$, mentre in quello obliquo i due spazi sono differenti e possono essere totalmente estranei tra di loro.
Sia A matrice $n \times n$ ad entrate reali mentre $\mathcal{K}$ e $\mathcal{L}$ i sopracitati sottospazi di $\mathbb{R}^n$ di dimensione m. Una tecnica di proiezione sul sottospazio $\mathcal{K}$ e ortogonale a L è un processo in cui si trova $\tilde{x}$, soluzione approssimata di $Ax = b$, imponendo le condizioni che $\tilde{x}$ appartenga a $\mathcal{K}$ e che il nuovo vettore dei residui sia ortogonale a $\mathcal{L}$. In sostanza :

$$\text{Trovare } \tilde{x} \text{ t.c. } b - A\tilde{x} \perp \mathcal{L}$$

Se invece ci ritroviamo con un vettore iniziale x_0 allora la soluzione approssimata bisogna ricercarla nello spazio affine $x_0 + \mathcal{K}$ anziché nello spazio vettoriale $\mathcal{K}$. Partendo da questo spazio, possiamo scrivere la nostra soluzione approssimata $\tilde{x} = x_0 + \delta$, mentre il vettore residuo iniziale sarà $r_0 = b - Ax_0$. Da queste considerazioni la condizione di ortogonalità diventa $b - A(x_0 + \delta)$ $= r_0$ - $A\delta \perp \mathcal{L}$. Riassumendo quindi abbiamo:

$$\tilde{x} = x_0 + \delta, \quad \delta \in \mathcal{K} \tag{2.2}$$

$$\langle r_0 - A\delta, w \rangle = 0, \quad \forall w \in \mathcal{L} \tag{2.3}$$

Questo è un passo base dei metodi di proiezione nella sua forma più generale, inoltre le più comuni tecniche usano una successione di queste proiezioni. Per concludere questa breve introduzione daremo una notazione matriciale, che ci permette di avvicinarci a ciò che realmente si fa negli algoritmi. Sia V $= [v_1, ..., v_m]$ una matrice $n \times m$ le cui colonne, v_i, sono vettori base di $\mathcal{K}$, e in maniera simile sia W $= [w_1, ..., w_m]$ una matrice $n \times m$, con w_i vettori base di $\mathcal{L}$. Se si scrivesse la soluzione approssimata in questo modo

$$\tilde{x} = x_0 + Vy$$

(chiaramente $\delta = Vy$), allora la nostra condizione di ortogonalità (2.3) ci porta al seguente sistema di equazioni per il vettore y:

$$W^T AVy = W^T r_0$$

Sotto l'assunzione che la matrice $W^T AV$, di dimensione $m \times m$, sia non singolare allora riusciamo a ricavare l'espressione della soluzione approssimata

$$\tilde{x} = x_0 + V(W^T AV)^{-1} W^T r_0 \tag{2.4}$$

ALGORITMO *Prototipo metodo proiettivo*

1. $x = x_0$
2. Repeat
3. Seleziona coppia di sottospazi $\mathcal{K}, \mathcal{L}$
4. Scegli base V e W per $\mathcal{K}, \mathcal{L}$
5. $r = b - Ax; \quad y = (W^T AV)^{-1} W^T r_0; \quad x = x + Vy;$
6. Until(condizione di convergenza)

La soluzione approssimata è definita solamente quando la matrice $W^T AV$ è non singolare, e questa assunzione non è garantita nemmeno quando A è non singolare.

Osservazione 2.0.1. Si verifica facilmente che $W^T AV$ è non singolare se e solo se nessun vettore dello spazio A$\mathcal{K}$ è ortogonale al sottospazio $\mathcal{L}$.

Ora esporremmo una proposizione che ci darà due casi in cui è garantita sicuramente la non singolarità di $W^T AV$

Proposizione 2.1. *Sia A matrice $n \times n$, $\mathcal{K}$ $\mathcal{L}$ due sottospazi di $\mathbb{R}^n$, tali che soddisfino una delle seguenti due condizioni:*

i *A definita positiva e $\mathcal{K} = \mathcal{L}$*

ii *A non singolare e $\mathcal{L} = A\mathcal{K}$*

Allora la matrice $B = W^T AV$ è non singolare, per qualunque scelta di V e W come basi rispettivamente di $\mathcal{K}$ e $\mathcal{L}$

Dimostrazione. Siano V e W due basi qualunque di $\mathcal{K}$ e $\mathcal{L}$.
Caso i): Dal fatto che $\mathcal{K}$ e $\mathcal{L}$ sono gli stessi, possiamo sempre esprimere W come W = VG, con G matrice $m \times m$ non singolare. Allora B = $W^T AV$= $G^T V^T AV$.
A è definita positiva e lo è anche $V^T AV$ e questo mostra che B è non singolare.
Caso ii): Ora possiamo scrivere W come W = AVG, dove G è la stessa matrice del caso i). Allora B = $W^T AV$= $G^T (AV)^T AV$.
Siccome A è non singolare, la matrice AV è a rango pieno e da qui risulta $(AV)^T AV$ non singolare. Questo implica infine che B sia non singolare. □

Con le prossime due proposizioni l'obiettivo sarà quello di apprendere qualcosa sulla qualità dell'approssimazione ottenuta da un generico metodo di proiezione.Sotto ipotesi particolari andremo a stabilire l'ottimalità della soluzione approssimata.

Proposizione 2.2. *Sia A una matrice simmetrica e definita positiva e* $\mathcal{L}=\mathcal{K}$. *Il vettore* $\tilde{x}$ *è il risultato di un metodo di proiezione ortogonale su* $\mathcal{K}$, *con vettore iniziale* x_0, *se e solo se esso minimizza la norma-A dell'errore su* $x_0+\mathcal{K}$

$$E(\tilde{x}) = \min_{x \in x_0+\mathcal{K}} E(x)$$

dove $E(x) = \langle A(x_* - x), x_* - x\rangle^{1/2}$

Proposizione 2.3. *Sia A una matrice quadrata e* $\mathcal{L}= A\mathcal{K}$. *Il vettore* $\tilde{x}$ *è il risultato di un metodo di proiezione obliquo su* $\mathcal{K}$ *ortogonalmente a* $\mathcal{L}$, *con vettore iniziale* x_0, *se e solo se esso minimizza la norma-2 del vettore residuo b - Ax*$_0$ *su* $x_0+\mathcal{K}$

$$R(\tilde{x}) = \min_{x \in x_0+\mathcal{K}} R(x)$$

dove $R(x) = \|b - Ax\|_2$

Osservazione 2.3.1. Si può notare come non sia necessario che A sia non singolare nell'ultima proposizione. Nel caso A fosse singolare ci sarebbero un' infinità di $\tilde{x}$ che soddisfino la condizione di ottimalità.

Ricordando sempre che trattiamo i casi $\mathcal{L} = \mathcal{K}$ e $\mathcal{L} = A\mathcal{K}$, in entrambi possiamo interpretare il risultato di un processo di proiezione come l'azione di un operatore di proiezione ortogonale sul residuo iniziale o sull'errore iniziale. Partiamo con il caso $\mathcal{L} = A\mathcal{K}$, sia r_0 il residuo iniziale, $r_0 = b - Ax_0$, e $\tilde{r}$ il residuo ottenuto dal processo di proiezione, $\tilde{r} = b - A\tilde{x}$.

$$\tilde{r} = b - A(x_0 + \delta) = r_0 - A\delta \tag{2.5}$$

dove δ è ottenuto imponendo la condizione che $r_0 - A\delta$ sia ortogonale ad $A\mathcal{K}$. Quindi il vettore Aδ è la proiezione ortogonale del vettore r_0 sul sottospazio A$\mathcal{K}$.

Proposizione 2.4. *Sia* $\tilde{x}$ *la soluzione approssimata ottenuta da un processo di proiezione su* $\mathcal{K}$ *ortogonalmente a* $\mathcal{L} = A\mathcal{K}$ *e sia* $\tilde{r} = b - A\tilde{x}$ *il residuo associato. Allora*

$$\tilde{r} = (I - P)r_0 \tag{2.6}$$

dove P denota l'operatore di proiezione ortogonale sul sottospazio $A\mathcal{K}$.

Una conseguenza della proposizione è che la norma-2 del vettore residuo, ottenuta dopo il processo di proiezione,non supera la norma-2 del vettore

residuo iniziale

$$\|\tilde{r}\|_2 \leq \|r_0\|_2$$

Questa classe di metodi prende il nome di *proiezione residua*.
Ora prendiamo in considerazione il caso $\mathcal{L} = \mathcal{K}$ e A matrice simmetrica e definita positiva. Sia $d_0 = x_* - x_0$ l'errore iniziale, dove con x_* denotiamo la soluzione esatta del sistema, similarmente $\tilde{d} = x_* - \tilde{x}$, dove $\tilde{x} = x_0 + \delta$ è la soluzione approssimata data dallo step di proiezione. Allora dalla (2.5) si ha

$$A\tilde{d} = \tilde{r} = A(d_0 - \delta)$$

Infatti : $\tilde{d} = x_* - \tilde{x} = x_* - x_0 - \delta = d_0 - \delta \quad \Rightarrow \quad \delta = d_0 - \tilde{d}$
$\tilde{r} = r_0 - A(d_0 - \tilde{d}) = A\tilde{d} + (b - Ax_0) - A(x_* - x_0) = A\tilde{d}$
dove δ è ottenuto vincolando il vettore residuo $r_0 - A\delta$ a essere ortogonale a $\mathcal{K}$:

$$\langle r_0 - A\delta, w\rangle = 0 \qquad \Longleftrightarrow \qquad \langle A(d_0 - \delta), w\rangle = 0 \qquad \forall w \in \mathcal{K}$$

2.2.1 Processi di proiezione unidimensionali

In questa sezione si forniranno alcuni facili esempi di processi di proiezione unidimensionali. Essi sono definiti in questo modo:

$$\mathcal{K} = span\{v\} \qquad e \qquad \mathcal{L} = span\{w\}$$

dove v e w sono due vettori. In questo caso la nuova approssimazione prende la forma di $x + \alpha v$ e la condizione di Petrov-Galerkin ($r - A\delta \perp w$ con $\delta = \alpha v$) porta a:

$$\alpha = \frac{< r, w >}{< Av, w >}$$

Discesa rapida

L'algoritmo di discesa rapida è definito per il caso in cui A è Simmetrica e definita positiva. Esso consiste nel prendere ad ogni passo $v = r$ e $w = r$. Questo porta alla seguente procedura iterativa:

ALGORITMO **Discesa rapida**

1. Calcola $r = b - Ax$ e $p = Ar$
2. Repeat
3. $\quad \alpha = \langle r, w\rangle / \langle Av, w\rangle$
4. $\quad x = x + \alpha r$
5. $\quad r = r - \alpha p$
6. $\quad p = Ar$
7. Until (criterio convergenza)

Ad ogni passo si minimizza $f(x) = \langle A(x - x_*), x - x_*\rangle$ su tutti i vettori della forma $x + \alpha d$, dove d è $-\nabla f$. L'opposto del gradiente è la direzione che localmente ha il tasso di decrescita più veloce. Con il successivo lemma dimostreremo che la convergenza è garantita quando A è simmetrica e definita positiva. Per prima cosa enunciamo il seguente lemma

Lemma 2.5. ***Disuguaglianza di Kantorovich.*** *Sia B una qualunque matrice reale simmetrica e definita positiva e λ_{max} e λ_{min} il massimo e il minimo tra gli autovalori. Allora*

$$\frac{\langle Bx, x\rangle\langle B^{-1}x, x\rangle}{\langle x, x\rangle^2} \leq \frac{(\lambda_{max} + \lambda_{min})^2}{4\lambda_{max}\lambda_{min}} \qquad \forall x \neq 0 \tag{2.7}$$

Dimostrazione. Senza ledere la generalità, possiamo supporre che x sia un vettore di norma unitaria. Dal fatto che B sia simmetrica allora sappiamo che esse è unitariamente simile ad una matrice diagonale, $B = Q^T DQ$, allora

$$\langle Bx, x\rangle\langle B^{-1}x, x\rangle \quad = \quad \langle DQx, Qx\rangle\langle D^{-1}Qx, Qx\rangle$$

Ponendo $y = Qx = (y_1, \ldots, y_n)^T$ e $\beta_i = y_i^2$ si può notare che

$$\lambda = \langle Dy, y\rangle = \sum_{i=1}^{n} \beta_i \lambda_i$$

è una combinazione convessa degli autovalori $\lambda_i \forall i$. Tutto ciò porta a

$$\langle Bx, x\rangle\langle B^{-1}x, x\rangle = \lambda\Psi(y) \qquad \text{con} \qquad \Psi(y) = \langle D^{-1}y, y\rangle = \sum_{i=1}^{n} \beta_i \frac{1}{\lambda_i}$$

Si noti che la funzione $f(\lambda) = \frac{1}{\lambda_i}$ è convessa, $\Psi(y)$ è limitata superiormente

dalla curva lineare passante per i punti $(\lambda_1, \frac{1}{\lambda_1})$ e $(\lambda_n, \frac{1}{\lambda_n})$

$$\Psi(y) \leq \tfrac{1}{\lambda_1} + \tfrac{1}{\lambda_n} + \tfrac{\lambda}{\lambda_1\lambda_n}.$$

Quindi

$$\langle Bx, x\rangle\langle B^{-1}x, x\rangle = \lambda\Psi(y) \leq \lambda\left(\tfrac{1}{\lambda_1} + \tfrac{1}{\lambda_n} + \tfrac{\lambda}{\lambda_1\lambda_n}\right)$$

Il massimo dell'espressione a destra si trova per $\lambda = \frac{1}{2}(\lambda_1 + \lambda_n)$, e sostituito all'espressione precedente ci porta alla nostra tesi. □

Il seguente teorema invece, grazie al lemma precedente, ci darà la convergenza del metodo.

Teorema 2.6. *Sia A matrice Simmetrica e definita positiva. Allora la A-norma del vettore errore $d_k = x_* - x_k$ generato dall'algoritmo di Discesa rapida soddisfa la seguente relazione*

$$\|d_{k+1}\|_A \leq \frac{\lambda_{max} - \lambda_{min}}{\lambda_{max} + \lambda_{min}}\|d_k\|_A \tag{2.8}$$

e così converge per qualunque vettore iniziale x_0

Dimostrazione. Si parte esplicitando la A-norma di $d_{k+1} = d_k - \alpha_k r_k$

$$\begin{aligned}\|d_{k+1}\|_A^2 = \langle d_{k+1}, d_k - \alpha r_k\rangle_A = \langle d_{k+1}, d_k\rangle_A - \alpha_k\langle d_{k+1}, r_k\rangle_A \\ = \langle d_{k+1}, Ad_k\rangle - \alpha_k\langle Ad_{k+1}, r_k\rangle = \langle d_{k+1}, r_k\rangle\end{aligned}$$

L'ultima espressione è dovuta all'uguaglianza $Ad_k = r_k$ e all'ortogonalità tra r_k e r_{k+1}. Riprendendo la relazione precedente e sfruttando anche la definizione di α_k si ha

$$\begin{aligned}\|d_{k+1}\|_A^2 = \langle d_k - \alpha_k r_k, r_k\rangle = \langle d_k, r_k\rangle - \alpha_k\langle r_k, r_k\rangle \\ = \|d_k\|_A^2 - \frac{\langle r_k, r_k\rangle}{\langle Ar_k, r_k\rangle}\langle r_k, r_k\rangle = \|d_k\|_A^2\left(1 - \frac{\langle r_k, r_k\rangle}{\langle r_k, Ar_k\rangle}\frac{\langle r_k, r_k\rangle}{\langle r_k, A^{-1}r_k\rangle}\right)\end{aligned}$$

Applicando la disuguaglianza di Kantorovich (2.7) otteniamo la tesi. □

Metodo del residuo minimo

Ora prenderemo in considerazione solamente matrici A definite positive e prenderemo ad ogni step i vettori $v = r$ e $w = Ar$. Il metodo dei residuo minimo è il seguente

ALGORITMO **Residuo Minimo**

1. Calcola $r = b - Ax$ e $p = Ar$
2. Repeat
3. $\quad \alpha = \langle p, r\rangle / \langle p, p\rangle$
4. $\quad x = x + \alpha r$
5. $\quad r = r - \alpha p$
6. $\quad p = Ar$
7. Until (criterio convergenza)

Qui ad ogni passo si minimizza $f(x) = \|b - Ax\|_2^2$. La convergenza del metodo invece è data dal seguente teorema.

Teorema 2.7. *Sia A una matrice reale e simmetrica e siano*

$$\mu = \lambda_{min}(A) \qquad \sigma = \|A\|_2$$

Allora i vettori residui, generati dall'algoritmo del minimo residuo, soddisfano la relazione

$$\|r_{k+1}\|_2 \leq \left(1 - \frac{\mu^2}{\sigma^2}\right)^{\frac{1}{2}} \|r_k\|_2 \tag{2.9}$$

e così l'algoritmo converge per qualunque vettore iniziale x_0

Dimostrazione. Questa prima parte della dimostrazione procede analogamente a quella effettuata per il metodo di discesa ripida. Si parte dalla relazione

$$\begin{aligned} \|r_{k+1}\|_2^2 &= \langle r_k - \alpha_k A r_k, r_k - \alpha_k A r_k\rangle \\ &= \langle r_k - \alpha_k A r_k, r_k\rangle - \alpha_k \langle r_k - \alpha_k A r_k, A r_k\rangle \end{aligned}$$

Per costruzione il nuovo vettore residuo, $r_k - \alpha_k A r_k$, deve essere ortogonale alla direzione di ricerca Ar_k e come risultato di questa assunzione otteniamo che il secondo termine dell'equazione sopra si annulla. Così otteniamo:

$$\begin{aligned} \|r_{k+1}\|_2^2 &= \langle r_k - \alpha_k A r_k, r_k\rangle \\ &= \langle r_k, r_k\rangle - \alpha_k \langle A r_k, r_k\rangle \end{aligned}$$

$$= \|r_k\|_2^2 \left(1 - \frac{\langle Ar_k, r_k\rangle}{\langle r_k, r_k\rangle} \frac{\langle Ar_k, r_k\rangle}{\langle Ar_k, Ar_k\rangle}\right)$$

$$= \|r_k\|_2^2 \left(1 - \frac{\langle Ar_k, r_k\rangle^2}{\langle r_k, r_k\rangle^2} \frac{\|r_k\|_2^2}{\|Ar_k\|_2^2}\right)$$

Ricordandoci la definizione di μ data nelle ipotesi, possiamo assumere che

$$\frac{\langle Ax, x\rangle}{\langle x, x\rangle} \geq \mu > 0$$

A questo punto si arriverà alla tesi usando la diseguaglianza $\|Ar_k\| \leq \|A\|_2 \|r_k\|_2$ □

Osservazione 2.7.1. Una interessante osservazione può essere fatta, se si considera il tutto da un punto di vista goniometrico. Definiamo cosi

$$\cos\,\theta_k = \frac{\langle Ar_k, r_k\rangle}{\|Ar_k\|_2 \|r_k\|_2}$$

Allora possiamo riscrivere la norma di r_{k+1} in funzione di θ_k

$$\begin{aligned}\|r_{k+1}\|_2^2 = \|r_k\|_2^2 \left(\ 1 \text{ - } \frac{\langle Ar_k, r_k\rangle}{\langle r_k, r_k\rangle} \frac{\langle Ar_k, r_k\rangle}{\langle Ar_k, Ar_k\rangle}\ \right) \\ = \|r_k\|_2^2 \left(\ 1 \text{ - } \cos^2\,\theta_k\right) \\ = \|r_k\|_2^2\ \text{sen}^2\ \theta_k\end{aligned}$$

Ad ogni passo la riduzione della norma del residuo è uguale al seno dell'angolo acuto tra r e Ar. Il fattore di convergenza è comunque limitato da

$$\rho = \max_{x \in \mathbb{R}^n, x \neq 0} \sin\,\beta$$

dove β è l'angolo acuto tra x e Ax. Il valore massimo di β è garantito sia minore di $\frac{\pi}{2}$ quando A è definita positiva.

2.2.2 Proprietà

Consideriamo il sottospazio di Krylov con la seguente forma $\mathcal{K}_m(A, v)$, che denoteremo semplicemente con $\mathcal{K}_m$. Una prima considerazione da fare è che la dimensione del sottospazio degli approssimanti aumenta di uno ad ogni passo del processo di approssimazione. Una prima proprietà, già vista nell'introduzione di questo capitolo, riguardate il sottospazio di Krylov è che

$\mathcal{K}_m$ è il sottospazio di tutti i vettori di $\mathbb{R}^n$ che possono essere scritti nella forma $x = p(A)v$, dove p è un polinomio di grado non superiore a $m-1$.
Ricordiamo che il *polinomio minimo* di un vettore v è il polinomio monico non nullo p di grado minore tale che $p(A)v = 0$, mentre con il grado di v intendiamo il grado del polinomio minimo di v. Infine definiamo un *sottospazio invariante* di un operatore lineare $T : V \to V$, dove V è uno spazio vettoriale, come un sottospazio vettoriale W di V tale che $T(W) \subset W$.
Dopo queste premesse possiamo enunciare la seguente proposizione

Proposizione 2.8. *Sia μ il grado di v. Allora $\mathcal{K}_\mu$ è invariante in A e $\mathcal{K}_m = \mathcal{K}_\mu \ \forall m \geq \mu$*

Come abbiamo detto sopra, il grado di $\mathcal{K}_m$ è non decrescente e la seguente proposizione determinerà in generale la dimensione del sottospazio di Krylov.

Proposizione 2.9. *Il sottospazio di Krylov $\mathcal{K}_m$ è di dimensione m se e solo se il grado di v non è minore di m.*

$$dim(\mathcal{K}_m) = m \Longleftrightarrow grado\ v \geq m \tag{2.10}$$

Quindi

$$dim(\mathcal{K}_m) = min\{m, grado\ v\} \tag{2.11}$$

Dimostrazione. I vettori $v, Av, \ldots, A^{m-1}v$ formano una base di $\mathcal{K}_m$ se e solo se per qualunque α_i, con $i = 0, \ldots, m-1$ e con almeno un α_i non nullo, la combinazione lineare $\sum_{i=0}^{m-1} \alpha_i A^i v$ è diverso da 0 . Questo è equivalente alla condizione che l'unico polinomio di grado ≤ m-1 per il quale $p(A)v = 0$ è il polinomio nullo.
La (2.11) è una conseguenza della proposizione precedente. □

2.3 Algoritmi di fattorizzazione

2.3.1 Metodo di Arnoldi

Il metodo di Arnoldi è un metodo di proiezione ortogonale su $\mathcal{K}_m$ per la costruzione di matrici Hessenberg. Esso risulta utile nel caso in cui A sia non simmetrica, e quindi non si possa applicare il metodo di Lanczos(che esporremo in seguito). Cosa molto importante di questo algoritmo è che gli autovalori di una matrice di Hessenberg ottenuti da un numero di passi più piccolo di n possono fornire un'accurata approssimazione di qualche autovalore della matrice originale.

L'algoritmo di Arnoldi si basa sulla costruzione di una base ortogonale del sottospazio di Krylov $\mathcal{K}_m$. In aritmetica esatta, una variante base dell'algoritmo è la seguente:

ALGORITMO **Arnoldi**

1. Scelto un vettore v_1 t.c. $\|v_1\|_2 = 1$
2. for j = 1,...,m
3. for i=1,...,j
4. $h_{ij} = \langle Av_j, v_i \rangle$
5. $w_j = Av_j - \sum_{i=1}^{j} h_{ij} v_i$
6. End
7. $h_{j+1,\,j} = \|w_j\|_2$
8. if ($h_{j+1,j} = 0$) Stop
9. $v_{j+1} = \dfrac{w_j}{h_{j+1,\,j}}$
10. End

Ad ogni passo, l'algoritmo moltiplica il precedente vettore v_j per A e ortogonalizza il vettore risultante w_j rispetto a tutti i precedenti v_i attraverso la procedura di Gram-Schmidt.
Ora descriveremo alcune proprietà dell'algoritmo.

Proposizione 2.10. *Assumiamo che l'algoritmo di Arnoldi non si fermi prima di m passi. Allora i vettori v_i ($i = 1, \ldots, m$) formano una base ortonormale per il sottospazio di Krylov*

$$\mathcal{K}_m = span\{v_1, Av_1, \ldots, A^{m-1}v_1\}$$

Dimostrazione. Per costruzione possiamo dire che v_j sono ortonormali. Il fatto che formino una base per lo spazio $\mathcal{K}_m$ deriva dalla forma di ogni v_j che è del tipo $q_{j-1}(A)v_1$, dove q_{j-1} è un polinomio di grado $j-1$. Dimostriamolo per induzione su j.
$j = 1 : v_1 = q_0(A)v_1$ con $q_0(t) = 1$
Data come ipotesi induttiva vera per j, consideriamo il caso $j+1$.
$j+1$: considerando le linee 5 e 9 dell'algoritmo, allora abbiamo

$$h_{j+1,\,j}v_{j+1} = Av_j - \sum_{i=1}^{j} h_{ij}v_i = Aq_{j-1}(A)v_1 - \sum_{i=1}^{j} h_{ij}q_{j-1}(A)v_1$$

□

Proposizione 2.11. *Siano V_m la matrice $n \times m$ costruita con i vettori colonna $v_1, \ldots, v_m$, $\tilde{H}_m$ la matrice di Hessenberg $(m+1) \times m$ con entrate non nulle h_{ij} definite come nell'algoritmo di Arnoldi, e sia H_m la matrice ottenuta eliminando l'ultima riga da $\tilde{H}_m$. Allora si hanno le seguenti relazioni:*

$$\begin{aligned} AV_m &= V_m H_m + w_m e_m^T \\ &= V_{m+1}\tilde{H}_m \qquad (2.12) \\ V_m^T A V_m &= H_m \qquad (2.13) \end{aligned}$$

Come si può notare facilmente, l'algoritmo potrebbe interrompersi quando la norma di w_j si azzera ad un certo passo j. In questo caso, il vettore v_{j+1} non può essere calcolato e l'algoritmo si ferma.

Proposizione 2.12. *L'algoritmo di Arnoldi si ferma (linea 8 dell'algoritmo) se e solo se il polinomio minimo di v_1 è di grado j.*
Inoltre, in questo caso $\mathcal{K}_j$ è invariante rispetto ad A

2.3.2 Metodo di Lanczos

L'algoritmo di Lanczos può essere visto come una semplificazione del metodo di Arnoldi nel caso particolare in cui A è simmetrica. Quando A è simmetrica allora la matrice di Hessenberg H_m diventa tridiagonale simmetrica. Per introdurre l'algoritmo di Lanczos partiamo prima con un risultato dato dal teorema seguente:

Teorema 2.13. *Data A matrice simmetrica a cui viene applicato il metodo di Arnoldi. Allora i coefficienti h_{ij} generati dall'algoritmo sono tali che*

$$\begin{aligned} h_{ij} &= 0 \ \textit{per}\ 1 \le i \ne j-1 \\ h_{j,j+1} &= h_{j+1,j} \ \textit{per}\ j = 1,\ldots,m \end{aligned}$$

Dimostrazione. La verifica è una semplice constatazione che $H_m = V_m^T A V_m$ è sia simmetrica che di Hessenberg. □

La notazione standard per descrivere l'algoritmo di Lanczos pone $\alpha_j = h_{j,j}$ e $\beta_j = h_{j-1,j}$ e con T_m la matrice H_m risultato dell'algoritmo che è della forma

$$T_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \alpha_3 & & \\ & . & . & . & \\ & & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \beta_m & \alpha_m \end{pmatrix} \tag{2.14}$$

Questo porta al seguente metodo

ALGORITMO **Lanczos**

1. Scelto vettore iniziale v_1 di norma-2 unitaria. $\beta_1 = 0; r_0 = 0$
2. for j = 1,. . . ,m
3. $w_j = Av_j - \beta_j v_{j-1}$
4. $\alpha_j = \langle w_j, v_j \rangle$
5. $w_j = w_j - \alpha_j v_j$
6. $\beta_{j+1} = \|w_j\|_2$
7. if ($\beta_{j+1} = 0$) Stop
8. $v_{j+1} = \dfrac{w_j}{\beta_{j+1}}$
9. End

L'algoritmo garantisce, almeno in aritmetica esatta, che i vettori v_i siano ortogonali. In realtà l'ortogonalità di questi vettori si osserva solamente nei primi passi dell'algoritmo. Dopo qualche passo, i v_i iniziano a perdere rapidamente la loro ortogonalità globale. La differenza con il metodo di Arnoldi, oltre alla forma della matrice H_m, è che solamente tre vettori devono essere salvati.

2.3.3 Metodo di Golub - Kahan

Consideriamo il caso $A \in \mathbb{R}^{m \times n}$ con $m \geq n$, la decomposizione di Golub-Kahan mostra che esistono matrici ortogonali $U = (u_1, \ldots, u_m) \in \mathbb{R}^{m \times m}$ e $V = (v_1, \ldots, v_n) \in \mathbb{R}^{n \times n}$ che se applicate ad A danno come risultato una matrice bidiagonale B, $U^T AV = B$

$$B = \begin{pmatrix} \alpha_1 & 0 & . & . & 0 \\ \beta_2 & \alpha_2 & 0 & . & . \\ 0 & \beta_3 & \alpha_3 & 0 & . \\ . & . & . & . & . \\ . & . & . & \beta_{n-1} & \alpha_n \\ 0 & . & . & 0 & \beta_n \end{pmatrix}$$

Tutto questo deriva dal seguente algoritmo :

ALGORITMO **Golub - Kahan**

1. A a rango pieno; $u_0 \in \mathbb{R}^m$ t.c. $\|u_0\|_2 = 1; k = 1; p_0 = u_0; \beta_1 = 1; v_0 = 0$
2. while $(\beta_k > 0)$
3. $u_{k+1} = \dfrac{p_{k-1}}{\beta_k}$
4. $r_k = A^T u_k - \beta_{k-1} v_{k-1}$
5. $\alpha_k = \|r_k\|_2$
6. $v_k = \frac{r_k}{\alpha_k}$
7. $p_k = A v_k - \alpha_k u_k$
8. $\beta_{k+1} = \|p_k\|_2$
9. $k = k + 1$
10. End

Supponendo che l'algoritmo si fermi al passo k otteniamo le seguenti relazioni

$$AV_k = U_{k+1} B_{k+1,k} \quad ; \quad A^T U_k = V_k B_{k,k}^T \tag{2.15}$$

dove le matrici $U_{k+1} \in \mathbb{R}^{m \times (k+1)}$ e $V_k \in \mathbb{R}^{n \times k}$ hanno colonne ortonormali con

$$\mathcal{R}(U_{k+1}) = \mathcal{K}_{k+1}(AA^T, b) = \text{span}\{b, (AA^T)b, \ldots, (AA^T)^k b\}$$
$$\mathcal{R}(V_k) = \mathcal{K}_k(A^T A, A^T b) = \text{span}\{A^T b, (A^T A)A^T b, \ldots, (A^T A)^{k-1} A^T b\}$$

2.4 Metodi per sistemi lineari

In questo paragrafo introdurremo alcuni algoritmi, quelli di maggiore interesse per noi, che si basano sul sottospazio di Krylov. Per curiosità su altri algoritmi e maggiore approfondimento si rimanda ai testi [1],[4].

2.4.1 Metodo del gradiente coniugato

L'algoritmo del gradiente coniugato è una delle migliori tecniche iterative per risolvere sistemi lineari sparsi con matrice associata A simmetrica e definita positiva. In altre parole è un metodo di proiezione ortogonale sul sottospazio di Krylov $\mathcal{K}_m(r_0, A)$, con r_0 vettore residuo iniziale. Nel metodo del gradiente coniugato, oltre alla condizione di ortogonalità, ci sarà utile la seguente definizione che verrà utilizzata per le direzioni di ricerca.

Definizione 2.2. Date due direzione p e q, esse si dicono *A-coniugate* se $p^T Aq = 0$.

Il vettore x_{j+1} può essere espresso come

$$x_{j+1} = x_j + \alpha_j p_j$$

In questo caso l'indice dei vettori p_j partirà da 0 e non da 1 come fatto precedentemente. Ora possiamo scrivere il vettore residuo

$$r_{j+1} = r_j - \alpha_j A p_j \tag{2.16}$$

Imponendo la condizione di ortogonalità sui vettori r_j e r_{j+1}, otteniamo l'espressione degli α_j che ci garantisce questa proprietà

$$\alpha_j = \frac{\langle r_j, r_j \rangle}{\langle Ap_j, r_j \rangle} \tag{2.17}$$

Si sa che la direzione di ricerca p_{j+1} è combinazione lineare di r_{j+1} e p_j, con p_j aggiustato con un opportuno scalare

$$p_{j+1} = r_{j+1} + \beta_j p_j \tag{2.18}$$

Utilizzando la condizione di ortogonalità tra Ap_j e p_{j-1}, arriviamo alla prima conseguenza della relazione di sopra, ed è

$$\langle Ap_j, r_j \rangle = \langle Ap_j, p_j - \beta_{j-1} p_{j-1} \rangle = \langle Ap_j, p_j \rangle.$$

A questo punto la (2.17) diventa $\alpha_j = \dfrac{\langle r_j, r_j \rangle}{\langle Ap_j, p_j \rangle}$. Mentre se imponiamo ai vettori p_{j+1} e p_j di essere A-coniugati allora nella (2.18) otteniamo la relazione dello scalare β_j

$$\beta_j = -\frac{\langle r_{j+1}, Ap_j \rangle}{\langle p_j, Ap_j \rangle} \tag{2.19}$$

Se dalla (2.16) si ricava Ap_j allora si ottiene una versione diversa di β_j

$$\beta_j = \frac{1}{\alpha_j} \frac{\langle r_{j+1}, r_{j+1} - r_j \rangle}{\langle p_j, Ap_j \rangle} = \frac{\langle r_{j+1}, r_{j+1} \rangle}{\langle r_j, r_j \rangle} \tag{2.20}$$

Dopo aver ricavato tutte queste relazioni possiamo finalmente giungere all'algoritmo

ALGORITMO **Gradiente Coniugato**

1. Calcolo $r_0 = b - Ax_0$, $p_0 = r_0$
2. for $j = 1, \ldots$ fino a convergenza
3. $\alpha_j = \dfrac{\langle r_j, r_j \rangle}{\langle Ap_j, r_j \rangle}$
4. $x_{j+1} = x_j + \alpha_j p_j$
5. $r_{j+1} = r_j - \alpha_j A p_j$
6. $\beta_j = \dfrac{\langle r_{j+1}, r_{j+1} \rangle}{\langle r_j, r_j \rangle}$
7. $p_{j+1} = r_{j+1} + \beta_j p_j$
8. End

Osservazione 2.13.1. Si noti che gli scalari α_j, β_j in questo algoritmo sono differenti da quelli dell'algoritmo di Lanczos.

Per capire come mai il metodo del gradiente coniugato sia considerato un metodo sul sottospazio di Krylov, ci viene in aiuto il seguente teorema

Teorema 2.14. *Sia $Ax = b$ con A matrice simmetrica e definita positiva. Se inizializzato con $x_0 = 0$ allora il metodo del gradiente coniugato procede fintantoché il residuo r_k non si annulla e valgono le seguenti identità per $l = 0, \ldots, k$*

$$\mathcal{K}_l := span(b, Ab, \ldots, A^{l-1}b) = span(x_1, , \ldots, x_l)$$

$$= span(p_0, \ldots, p_{l-1}) = span(r_0, \ldots, r_{l-1})$$

Si ha inoltre per $\forall l$

$$r_l^T r_j = 0 \text{ e } p_l^T A p_j = 0 \text{ per } j = 0, \ldots, l-1$$

Dimostrazione. La dimostrazione procede per induzione su l e qui ne daremo semplicemente un breve schema.
Caso $l = 0$: il risultato è banale poiché la scelta fatta per ipotesi, $x_0 = 0$, implica $r_0 = p_0$ e $x_1 = \alpha_0 p_0$.
Caso $l > 1$ si ha che

$$x_l = x_{l-1} + \alpha_{l-1} p_{l-1} \Rightarrow x_l \in \text{span}(p_0, \ldots, p_{l-1})$$
$$p_{l-1} = r_{l-1} + \beta_{l-2} p_{l-2} \Rightarrow r_{l-1} \in \text{span}(p_0, \ldots, p_{l-1})$$
$$r_{l-1} = r_{l-2} - \alpha_{l-2} A p_{l-2} \Rightarrow A^{l-1} b \in \text{span}(r_0, \ldots, r_{l-1})$$

Il fatto che i residui siano ortogonali e le direzioni di ricerca A-coniugate è facilmente verificabile dalle formule ricavate nel corso del paragrafo.

□

2.4.2 FOM

Dato un vettore iniziale x_0 per il sistema lineare $Ax = b$, consideriamo ora un metodo di proiezione ortogonale, con $\mathcal{L} = \mathcal{K} = \mathcal{K}_m(A, r_0)$ con

$$\mathcal{K}_m(A, r_0) = \text{span}\{r_0, Ar_0, A^2 r_0, \ldots, A^{m-1} r_0\}$$

dove $r_0 = b - Ax_0$. Questo metodo ricerca una soluzione approssimata x_m nel sottospazio affine $x_0 + \mathcal{K}_m$ di dimensione m, imponendo la condizione di Galerkin

$$b - Ax_m \perp \mathcal{K}_m \quad .$$

Se $v_1 = \dfrac{r_o}{\|r_0\|_2}$ nel metodo di Arnoldi descritto prima, e imponiamo $\beta = \|r_0\|_2$ allora abbiamo la (2.13) e

$$V_m^T r_0 = V_m^T(\beta v_1) = \beta v_1$$

Come risultato di ciò, usando il sottospazio m-dimensionale di sopra, otteniamo la soluzione approssimata

$$x_m = x_0 + V_m y_m \tag{2.21}$$

$$y_m = H_m^{-1}(\beta e_1) \tag{2.22}$$

Un metodo basato su questo procedimento viene chiamato FOM (Full Orthogonalization Method) e l'algoritmo seguente utilizza la procedura di Arnoldi

ma con l'utilizzo dell'algoritmo MGS (Modified Gram-Schmidt) per una maggiore stabilità numerica in quanto con il procedere delle iterazioni si può perdere l'ortogonalità dei vettori w_j.

ALGORITMO **FOM**

1. Calcolo $r_0 = b - Ax_0$, $\beta = \|r_0\|_2$ e $v_1 = \frac{r_o}{\beta}$
2. Definire matrice $m \times m$ $H_m = \{h_{ij}\}$
3. for $j = 1, \ldots, m$
4. $w_j = Av_j$
5. for i=1,. . . ,j
6. $h_{ij} = \langle w_j, v_i \rangle$
7. $w_j = w_j - h_{ij}v_i$
8. End
9. $h_{j+1,\,j} = \|w_j\|_2$
10. if ($h_{j+1,j} = 0$) $m = j$ Stop
11. $v_{j+1} = \dfrac{w_j}{h_{j+1,\,j}}$
12. End
13. Calcola $y_m = H_m^{-1}(\beta e_1)$ e $x_m = x_0 + V_m y_m$

L'algoritmo dipende dal parametro m che è la dimensione del sottospazio di Krylov.

Proposizione 2.15. *Il vettore residuo della soluzione approssimata x_m calcolato da FOM è tale che*

$$b - Ax_m = -h_{m+1,m} e_m^T y_m v_{m+1}$$

e quindi

$$\|b - Ax_m\|_2 = h_{m+1,m} |e_m^T y_m| \tag{2.23}$$

Dimostrazione. Dalle relazione date dall'algoritmo segue

$$\begin{aligned} b - Ax_m &= b - A(x_0 + V_m y_m) = r_0 - AV_m y_m \\ &= \beta v_1 - V_m H_m y_m - h_{m+1,m} e_m^T y_m v_{m+1} \end{aligned}$$

Per definizione di y_m si ha $H_m y_m = \beta e_1$ e così si ha $\beta v_1 = V_m H_m y_m$ e da li la tesi. □

2.4.3 Metodo di Lanczos

Il punto di partenza sarà l'algoritmo FOM per il caso A simmetrica. Dato il vettore iniziale x_0 per il sistema lineare $Ax = b$, i vettori v_i con $i = 1, .., m$ insieme alla matrice T_m, la soluzione approssimata ottenuta dal metodo di proiezione su $\mathcal{K}_m$ è dato da

$$x_m = x_0 + V_m y_m, \qquad y_m = T_m^{-1}(\beta e_1)$$

ALGORITMO **Lanczos per sistemi lineari**

1. Calcolo $r_0 = b - Ax_0$, $\beta = \|r_0\|_2$ e $v_1 = \frac{r_0}{\beta}$
2. for $j = 1, \ldots, m$
3. if (j = 1)
4. $w_j = Av_j$
5. else
6. $w_j = Av_j - \beta_j v_{j-1}$
7. $\alpha_j = \langle w_j, v_j \rangle$
8. $w_j = w_j - \alpha_j v_j$
9. $\beta_{j+1} = \|w_j\|_2$
10. if ($\beta_{j+1} = 0$) m = j break End
11. $v_{j+1} = \dfrac{w_j}{\beta_{j+1}}$
12. End
13. $T_m = \text{tridiag}(\beta_i, \alpha_i, \beta_{i+1})$ e $V_m = [v_1, ..., v_m]$

14. Calcola $y_m = T_m^{-1}(\beta e_1)$ e $x_m = x_0 + V_m y_m$

Molti risultati ottenuti per l'algoritmo di Arnoldi sono ancora validi. Per esempio il vettore residuo r_m è tale che

$$b - Ax_m = -\beta_{m+1,m} e_m^T y_m v_{m+1} \tag{2.24}$$

Un'altra variante di questo algoritmo si può ottenere partendo dalla fattorizzazione LU della matrice T_m. Scrivendo quindi la fattorizzazione la matrice prenderà la forma $T_m = L_m U_m$ così fatta

$$T_m = \begin{pmatrix} 1 & & & & \\ \lambda_1 & 1 & & & \\ & \lambda_2 & 1 & & \\ . & . & . & . & . \\ & & & \lambda_m & 1 \end{pmatrix} \begin{pmatrix} \eta_1 & \beta_2 & & & \\ & \eta_2 & \beta_3 & & \\ . & . & . & . & . \\ & & & \eta_{m-1} & \beta_m \\ & & & & \eta_m \end{pmatrix}$$

La soluzione approssimata prende così la forma $x_m = x_0 + V_m U_m^{-1} L_m^{-1}(\beta e_1)$. Siano $P_m = V_m U_m^{-1}$ e $z_m = L_m^{-1}\beta e_1$ allora la soluzione approssimata prende la forma di

$$x_m = x_0 + P_m z_m \quad .$$

L'ultima colonna della matrice P_m, che indicheremo con p_m, può essere calcolata tramite le precedenti colonne p_i e i vettori v_i attraverso la seguente relazione

$$p_m = \eta_m^{-1}[v_m - \beta_m p_{m-1}]$$

Questa espressione si ricava partendo dall'espressione $P_m U_m = V_m$, che possiamo scrivere in forma matriciale in questo modo

$$[p_1, \ldots, p_m] \begin{pmatrix} \eta_1 & \beta_2 & & & \\ & \eta_2 & \beta_3 & & \\ . & . & . & . & . \\ & & & \eta_{m-1} & \beta_m \\ & & & & \eta_m \end{pmatrix} = [v_1, \ldots, v_m]$$

Sfruttando il fatto che la matrice U_m sia una matrice a banda (si ha $u_{ij} = 0$ per $j < i$ e $j > i + 1$), si arriva alla relazione $p_{i-1}\beta_i + p_i\eta_i = v_i$ con $p_0 = \beta_1 = 0$, che per $i = m$ è la relazione scritta sopra.
Queste considerazioni ci portano ad un'altra versione dell'algoritmo di Lanczos per sistemi lineari.

ALGORITMO **D-Lanczos**

1. Calcolo $r_0 = b - Ax_0$, $\zeta_1 = \|r_0\|_2$ e $v_1 = \dfrac{r_0}{\zeta_1}$
2. $\lambda_1 = \beta_1 = 0, p_0 = 0$

3. for $j = 1, 2, \ldots$ (fino a convergenza)

4. if $(j = 1)$

5. $w = Av_j$

6. else

7. $w = Av_j - \beta_j v_{j-1}$

8. $\lambda_j = \dfrac{\beta_j}{\eta_{j-1}}$ e $\zeta_j = -\lambda_j \zeta_{j-1}$ End

9. $\alpha_j = \langle w, v_j \rangle$

10. $\eta_j = \alpha_j - \lambda_j \beta_j$

11. $p_j = \eta_j^{-1}[v_j - \beta_j p_{j-1}]$

12. $x_j = x_{j-1} + \zeta_j p_j$

13. if (x_j converge) break

14. $w = w - \alpha_j v_j$

15. $\beta_{j+1} = \|w\|_2$; $v_{j+1} = \dfrac{w}{\beta_{j+1}}$

16. End

Osservazione 2.15.1. Da notare che β_m è uno scalare calcolato dall'algoritmo di Lanczos mentre η_m risulta dal m-esimo passo dell'eliminazione gaussiana.

Questo algoritmo calcola progressivamente la soluzione del sistema tridiagonale $T_m y_m = \beta e_1$ usando l'eliminazione gaussiana senza pivoting. I due algoritmi esposti in questa parte sono matematicamente equivalenti e portano alla stessa soluzione approssimata se sono entrambi eseguibili.

Osservazione 2.15.2. Osserviamo che il vettore residuo per questo algoritmo segue la direzione di v_{m+1} e si vede dalla formula (2.24) quindi i vettori residui sono ortogonali tra di loro.

Data questa osservazione possiamo introdurre la prossima proposizione (per la facile dimostrazione si veda [1]).

Proposizione 2.16. *Sia $r_m = b - Ax_m$ il vettore residuo prodotto dai due algoritmi di Lanczos, e p_m il vettore prodotto dall'algoritmo D-Lanczos. Allora*

1. *Ogni r_m è della forma $r_m = \sigma_m v_{m+1}$, dove σ_m è un certo scalare. Inoltre i vettori residui sono ortogonali tra di loro.*

2. *i vettori p_i sono A-coniugati $\langle Ap_i, p_j \rangle = 0$ per $i \neq j$*

2.4.4 GMRES

Il metodo GMRES (Generalized Minimum Residual) è un metodo di proiezione che si basa sulla seguente scelta degli spazi $\mathcal{K} = \mathcal{K}_m$ e $\mathcal{L} = A\mathcal{K}_m$, dove con $\mathcal{K}_m$ intendiamo lo spazio di Krylov $\mathcal{K}_m(A, v_1)$ con $v_1 = \frac{r_0}{\|r_0\|_2}$.Questo algoritmo è una tecnica tale che minimizzi la norma del residuo su tutti i vettori della forma $x_0 + \mathcal{K}_m$.
Qualunque vettore x in $x_0 + \mathcal{K}_m$ può essere scritto nella forma

$$x = x_0 + V_m y \tag{2.25}$$

con y un vettore di $\mathbb{R}^m$. Definiamo la funzione $J(y)$

$$J(y) = \|b - Ax\|_2 \tag{2.26}$$

Utilizziamo la relazione ricavata dal processo di Arnoldi $AV_m = V_{m+1}\tilde{H}_m$, con $\tilde{H}_m$ matrice di Hessenberg di dimensione $(m+1) \times m$, per scrivere in maniera differente il residuo

$$\begin{aligned} b - Ax &= b - A(x_0 + V_m y) \\ &= r_0 - AV_m y \\ &= \beta v_1 - V_{m+1}\tilde{H}_m y \\ &= V_{m+1}(\beta e_1 - \tilde{H}_m y) \end{aligned}$$

Ricordando che la matrice V_{m+1} è formata da vettori colonna ortonormali, sostituendo l'ultima relazione trova nella nostra funzione da minimizzare, si arriverà alla formula

$$J(y) = \|\beta e_1 - \tilde{H}_m y\|_2 \tag{2.27}$$

L'approssimazione ottenuta con GMRES è l'unico vettore di $x_0 + \mathcal{K}_m$ che minimizza la (2.26), e sarà ottenuta in questo modo $x_m + V_m y_m$, dove con y_m si intende il minimo della funzione $J(y)$.

ALGORITMO **GMRES**

1. $r_0 = b - Ax$; $\beta = \|r_0\|_2$; $v_1 = \frac{r_0}{\beta}$
2. for $j = 1, \ldots, m$
3. $w_j = Av_j$

4. for $i = 1, \ldots, j$
5. $\quad h_{ij} = \langle w_j, v_i \rangle$
6. $\quad w_j = w_j - h_{ij} v_i$
7. End
8. $h_{j+1,j} = \|w_j\|_2$.
9. if($h_{j+1,j} = 0$) m=j; break End(if)
10. $v_{j+1} = \dfrac{w_j}{h_{j+1,j}}$
11. End
12. Definire $\tilde{H} = h_{i,j}$ con $i = 1, \ldots, m+1$, $j = 1, \ldots, m$
13. $y_m = \underset{y}{\operatorname{argmin}} \, \|\beta e_1 - \tilde{H}_m y\|_2$; $x_m = x_0 + V_m y_m$

Il minimo y_m è poco costoso da calcolare dato che esso richiede la risoluzione di un problema ai minimi quadrati $(m+1) \times m$.

Implementazione mediante matrici di Givens

Una tecnica comune per risolvere il problema dei minimi quadrati $\|\beta e_1 - \tilde{H}_m y\|_2$ è quello di trasformare la matrice di Hessenberg in una triangolare superiore usando la matrici di rotazione (Matrici di Givens). Definiamo la matrice di rotazione

$$\Omega_i = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & c_i & s_i & & \\ & & -s_i & c_i & & \\ & & & & \ddots & \\ & & & & & 1 \end{pmatrix} \leftarrow \text{riga i-esima}$$

con $c_i^2 + s_i^2 = 1$. Al m-esimo passo di GMRES avremo matrici di dimensione $(m+1) \times (m+1)$. I coefficienti s_i, c_i sono scelti in modo tale da eliminare h_{i+1} ad ogni passo. Per esempio, sia $m = 5$ e avremo così

$$\tilde{H}_5 = \begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{14} & h_{15} \\ h_{21} & h_{22} & h_{23} & h_{24} & h_{25} \\ & h_{32} & h_{33} & h_{34} & h_{35} \\ & & h_{43} & h_{44} & h_{45} \\ & & & h_{54} & h_{55} \\ & & & & h_{65} \end{pmatrix}, \qquad \tilde{g}_0 = \begin{pmatrix} \beta \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

e sia Ω_1 la matrice sopra definita con i seguenti coefficienti

$$s_1 = \frac{h_{21}}{\sqrt{h_{11}^2 + h_{21}^2}}, \qquad c_1 = \frac{h_{11}}{\sqrt{h_{11}^2 + h_{21}^2}}.$$

Applicando la nostra matrice di rotazione ad $\tilde{H}_5$ e $\tilde{g}_0$ otteniamo

$$\tilde{H}_5^{(1)} = \begin{pmatrix} h_{11}^{(1)} & h_{12}^{(1)} & h_{13}^{(1)} & h_{14}^{(1)} & h_{15}^{(1)} \\ & h_{22}^{(1)} & h_{23}^{(1)} & h_{24}^{(1)} & h_{25}^{(1)} \\ & h_{32}^{(1)} & h_{33}^{(1)} & h_{34}^{(1)} & h_{35}^{(1)} \\ & & h_{43}^{(1)} & h_{44}^{(1)} & h_{45}^{(1)} \\ & & & h_{54}^{(1)} & h_{55}^{(1)} \\ & & & & h_{65}^{(1)} \end{pmatrix}, \qquad \tilde{g}_1 = \begin{pmatrix} c_1\beta \\ -s_1\beta \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Con questo breve esempio illustrativo possiamo ben capire che dopo m rotazioni applicate ad $\tilde{H}_5$ arriveremo ad una matrice triangolare superiore. Per maggiore comprensibilità si consulti [1] e [4] Tutto quello che abbiamo fatto è stato definire i coefficienti c_i, s_i in modo tale da poter eliminare gli elementi al di sotto della diagonale, grazie al prodotto tra la matrice H con la matrice di rotazione. I coefficienti in generale seguono la seguente formula

$$s_i = \frac{h_{i+1,i}}{\sqrt{(h_{ii}^{(i-1)})^2 + h_{i+1,i}^2}}, \qquad c_i = \frac{h_{11}}{\sqrt{(h_{ii}^{(i-1)})^2 + h_{i+1,i}^2}}.$$

Sia Q_m il prodotto delle matrici Ω e definiamo la matrice $\tilde{R}_m$ e il vettore $\tilde{g}_m$

$$Q_m = \Omega_m \Omega_{m-1} \dots \Omega_1$$

$$\tilde{R}_m = \tilde{H}_m^{(m)} = Q_m \tilde{H}_m$$

$$\tilde{g}_m = Q_m(\beta e_1) = (\gamma_1, \dots, \gamma_{m+1})^T$$

Dati queste equazioni e dal fatto che Q_m sia unitaria possiamo riscrivere il problema dei minimi quadrati

$$\min \|\beta e_1 - \tilde{H}_m y\|_2 = \min \|\tilde{g}_m - \tilde{R}_m y\|_2$$

La soluzione del problema scritto qua sopra è semplicemente ottenuta risolvendo un sistema triangolare superiore, eliminando l'ultima riga dei vettori $\tilde{R}_m$ e $\tilde{g}_m$.

Proposizione 2.17. *Sia $m \leq n$ e siano Ω_i, con $i = 1, \dots, m$, le matrici di rotazione usate per trasformare $\tilde{H}_m$. Siano inoltre $\tilde{R}_m$ e $\tilde{g}_m$ la matrice e*

il vettore definiti come sopra, mentre R_m e g_m sono la matrice e il vettore ottenuti eliminando l'ultima riga da ognuno. Allora

1. *Il rango di AV_m è uguale al rango di R_m. In particolare se $r_{mm} = 0$, allora A è singolare.*

2. *y_m, che minimizza $\|\beta e_1 - \tilde{H}_m y\|_2$, è dato da*

$$y_m = R_m^{-1} g_m$$

3. *il vettore residuo al passo m soddisfa*

$$b - Ax_m = V_{m+1}(\beta e_1 - \tilde{H}_m y_m) = V_{m+1} Q_m^T (\gamma_{m+1} e_{m+1})$$

di conseguenza

$$\|b - Ax_m\|_2 = |\gamma_{m+1}| \tag{2.28}$$

Dimostrazione. Dimostriamo la (1). Usiamo la relazione $AV_m = V_{m+1}\tilde{H}_m$ e svolgiamo alcuni calcoli

$$\begin{aligned} AV_m &= V_{m+1}\tilde{H}_m \\ &= V_{m+1}Q_m^T Q_m \tilde{H}_m \\ &= V_{m+1}Q_m^T \tilde{R}_m \end{aligned}$$

Siccome $V_{m+1}Q_m^T$ è unitaria, il rango di AV_m è lo stesso di $\tilde{R}_m$, e a sua volta è uguale a quello di R_m, visto che differiscono solo per una riga di zeri. Se $r_{mm} = 0$ allora il rango di R_m è $\leq m-1$, e a sua volta anche AV_m. Dal fatto che V_m sia rango pieno, segue che A deve essere singolare.
Caso (2). Per questa parte non c'è nulla di nuovo rispetto a quello descritto precedentemente alla proposizione, basta semplicemente continuare da dove ci eravamo fermati.

$$\begin{aligned} \|\beta e_1 - \tilde{H}_m y\|_2^2 &= \|\tilde{g}_m - \tilde{R}_m y\|_2^2 \\ &= |\gamma_{m+1}|^2 + \|g_m - R_m y\|_2^2 \end{aligned}$$

Il minimo di questa quantità viene raggiunto quando la norma al secondo membro risulta zero. Dal fatto che R_m è non singolare, la soluzione sarà $y = R^{-1} g_m$.
Ultima parte, caso (3). Data la relazione $b - Ax = V_{m+1}(\beta e_1 - \tilde{H}_m y)$, per qualunque $x = x_0 + V_m y$, utilizzando le relazioni di $\tilde{g}_m$ e di $\tilde{R}_m$, si otterrà

$$\begin{aligned} b - Ax &= V_{m+1}(\beta e_1 - \tilde{H}_m y) \\ &= V_{m+1} Q_m^T Q_m (\beta e_1 - \tilde{H}_m y) \\ &= V_{m+1} Q_m^T (\tilde{g}_m - \tilde{R}_m y) \end{aligned}$$

Considerando ciò che è stato dimostrato nella (2), avremo come risultato $b - Ax_m = V_{m+1} Q_m^T (\gamma_{m+1} e_{m+1})$. L'ultimo risultato della tesi (3) è una semplice conseguenza, considerando che le due matrici V_{m+1} e Q_m^T hanno colonne ortonormali. □

Dopo tutto questo, si può inoltre implementare il processo in maniera progressiva ad ogni passo dell'algoritmo GMRES. Il processo avrà l'obiettivo di calcolare la norma del residuo ad ogni step, con virtualmente nessuna operazione addizionale.
Ora riprendiamo l'esempio facile lasciato prima, riguardante i coefficienti s_i e c_i $\tilde{H}$ e $\tilde{g}$, per arrivare ad una relazione che ci sarà utile per trovare il punto di breakdown del metodo GMRES. Assumiamo che le prime $m = 5$ matrici di rotazione siano già state applicate e che sia necessario un ulteriore passo dell'algoritmo. Date queste premesse avremo le matrici (in questo caso omettiamo gli apici nei coefficienti h_{ij} per snellire la notazione, abbiamo inoltre aumentato la dimensione della matrice di 1 rispetto all'esempio precedente)

$$\tilde{H}_6^{(5)} = \begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{14} & h_{15} & h_{16} \\ & h_{22} & h_{23} & h_{24} & h_{25} & h_{26} \\ & & h_{33} & h_{34} & h_{35} & h_{36} \\ & & & h_{44} & h_{45} & h_{46} \\ & & & & h_{55} & h_{56} \\ & & & & & h_{66} \\ & & & & & h_{76} \end{pmatrix}, \qquad \tilde{g}_6^{(5)} = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma 3 \\ . \\ . \\ \gamma_6 \\ 0 \end{pmatrix}$$

Un ulteriore step che deve essere eseguito per ottenere Av_6 e la sesta colonna di $\tilde{H}_6$. L'algoritmo continua nello stesso modo visto precedentemente ottenendo i coefficienti s_6 e c_6. Applicando la matrice di rotazione Ω_6 otteniamo le metrici

$$\tilde{R}_6 = \begin{pmatrix} r_{11} & r_{12} & r_{13} & r_{14} & r_{15} & r_{16} \\ & r_{22} & r_{23} & r_{24} & r_{25} & r_{26} \\ & & r_{33} & r_{34} & r_{35} & r_{36} \\ & & & r_{44} & r_{45} & r_{46} \\ & & & & r_{55} & r_{56} \\ & & & & & r_{66} \\ & & & & & 0 \end{pmatrix}, \qquad \tilde{g}_6 = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma 3 \\ . \\ . \\ c_6 \gamma_6 \\ -s_6 \gamma_6 \end{pmatrix}$$

A questo punto se la norma del residuo, data dal valore di $\mid \gamma_{m+1} \mid$, è piccola a sufficienza, il processo deve essere fermato. Le ultime righe di $\tilde{R}_m$ e $\tilde{g}_m$ vengo-

no eliminate e il sistema triangolare superiore che ne deriva verrà risolto per ottenere y_m. Di conseguenza la soluzione approssimata sarà $x_m = x_0 + V_m y_m$.

Osservazione 2.17.1. Da questo ultimo esempio possiamo arrivare ad una importante relazione che ci sarà utile per individuare successivamente un criterio di stop. La relazione lega i coefficienti di $\tilde{g}_m$ in questo modo

$$\gamma_{j+1} = -s_j \gamma_j \tag{2.29}$$

Punto di breakdown

Come osservato precedentemente ma visto maniera differente, l'algoritmo GMRES può arrestarsi quando entra nel loop di Arnoldi, che si verifica con $w_j = 0$ ossia $h_{j+1,j} = 0$ al passo j. In questa situazione l'algoritmo si ferma perché non può generare il successivo vettore di Arnoldi. Quindi il vettore residuo è nullo e l'algoritmo ci restituirà la soluzione esatta a ogni passo. Strano ma vero, anche il viceversa vale: se l'algoritmo si ferma quando $b - Ax_j = 0$ allo step j, allora $h_{j+1,j} = 0$.

Proposizione 2.18. *Sia A matrice non singolare. L'algoritmo GMRES si interrompe al passo j, ossia $h_{j+1,j} = 0$, se e solo se la soluzione approssimata x_j è esatta.*

Dimostrazione. ($\Rightarrow$) Osserviamo che se $h_{j+1,j} = 0$ allora $s_j = 0$. Infatti, dato che A è non singolare, abbiamo che $r_{jj} = h_{jj}^{j-1}$ è non nullo, grazie alla prima parte della Prop 2.15 e dall'espressione generale s_j otteniamo $s_j = 0$. Allora, sfruttando (2.28) e (2.29) e l'ultima assunzione su s_j, il tutto implica $r_j = 0$. Ora passiamo alla condizione sufficiente ($\Leftarrow$). Dal fatto che la soluzione esatta risulta al passo j e non a quello $j-1$, riesumando di nuovo la (2.29) otteniamo $s_j = 0$. Tenendo a mente l'espressione di s_j l'unica possibilità per avere 0, è che si annulli $h_{j+1,j}$, che è la tesi che volevamo. □

Convergenza del metodo GMRES

Inizieremo trattando subito un risultato globale che richiederà la proprietà di A di essere definita positiva.

Teorema 2.19. *Se A è definita positiva allora GMRES converge $\forall\, m \geq 1$.*

Dimostrazione. Risulta vero in quanto il sottospazio $\mathcal{K}_m$ contiene il vettore residuo iniziale a ogni inizio.Dato che l'algoritmo minimizza la norma del residuo nel sottospazio $\mathcal{K}_m$, ad ogni iterazione la norma del residuo sarà ridotta tanto quanto il risultato di un passo del Metodo del minimo residuo presente nell'allegato A. Quindi la disuguaglianza (2.9) è soddisfatta dai vettori residui prodotti ad ogni iterazione, di conseguenza l'algoritmo converge. □

Ora mostreremo un risultato simile a quello del metodo del gradiente coniugato che fornirà un limite superiore per il criterio di convergenza di GMRES.

Lemma 2.20. *Sia x_m la soluzione approssimata ottenuta al passo m dell'algoritmo GMRES e sia $r_m = b - Ax_m$ il residuo. Allora x_m è della forma*

$$x_m = x_0 + q_m(A)r_0$$

dove q_m è un polinomio di grado m-1 tale che

$$\|r_m\|_2 = \|(I - Aq_m(A))r_0\|_2 = \min_{q \in \mathbb{P}_{m-1}} \|(I - Aq(A))r_0\|_2$$

Dimostrazione. La tesi è vera in quanto x_m minimizza la norma-2 del residuo nel sottospazio affine $x_0 + \mathcal{K}_m$ come risultato della Proposizione 2.3 e dal fatto che $\mathcal{K}_m$ è l'insieme di tutti i vettori della forma $x_0 + q(A)r_0$, dove q è un polinomio si grado $\leq m - 1$. □

Proposizione 2.21. *Sia A matrice diagonalizzabile e $A = X\Lambda X^{-1}$, dove Λ è una matrice diagonale di autovalori λ_i con $i = 1,\ldots,n$. Definiamo*

$$\epsilon^{(m)} = \min_{p \in \mathbb{P}_{m-1}, p(0)=1} \max_i |p(\lambda_i)|$$

Allora la norma del residuo, ottenuta al passo m di GMRES, soddisfa la disuguaglianza

$$\|r_m\|_2 \leq \kappa_2(X)\epsilon^{(m)}\|r_0\|_2$$

dove $\kappa_2(X) = \|X\|_2 \, \|X^{-1}\|_2$ è il numero di condizionamento della matrice X.

Dimostrazione. Sia p un qualunque polinomio di grado $\leq$ m che soddisfa il vincolo $p(0) = 1$, e sia x il vettore di $\mathcal{K}_m$ associato a $b - Ax = p(A)r_0$. Allora

$$\|b - Ax\|_2 = \|\mathrm{X}p(\Lambda)\mathrm{X}^{-1}r_0\|_2 \leq \|\mathrm{X}\|_2 \, \|\mathrm{X}^{-1}\|_2\|r_0\|_2\|p(\Lambda)\|_2$$

Siccome A è diagonale, osserviamo che

$$\|p(\Lambda)\|_2 = \max_{i=1,..,n} |p(\lambda_i)|$$

Dal fatto che x_m minimizza la norma del residuo su $x_0 + \mathcal{K}_m$ allora

$$\|b - Ax_m\|_2 \leq \|b - Ax\|_2 \leq \|\mathrm{X}\|_2 \, \|\mathrm{X}^{-1}\|_2\|r_0\|_2 \max_{i=1,..,n} |p(\lambda_i)|$$

Ora passando al polinomio p che minimizza il secondo membro della disuguaglianza soprastante, ci porta alla tesi desiderata

$$\|b - Ax_m\|_2 \leq \|b - Ax\|_2 \leq \|\mathrm{X}\|_2 \ \|\mathrm{X}^{-1}\|_2 \|r_0\|_2 \epsilon^{(m)}$$

□

2.4.5 LSQR

Il metodo LSQR è estremamente simile al metodo GMRES. L'importante differenza è che, dato $x_0 = 0$ come vettore iniziale, allora x_k è scelto come minimo della norma del residuo $r_k = b - Ax_k$ in $\mathcal{K}_k(A^TA, A^Tb)$

Possiamo ora applicare la bidiagonalizzazione di Golub - Kahan al problema dei minimi quadrati ottenendo così

$$\begin{aligned}
\min_{x \in \mathcal{K}_k(A^TA, A^Tb)} \|Ax - b\|_2 &= \min_{y \in \mathbb{R}^k} \|AV_k y - b\|_2 \\
&= \min_{y \in \mathbb{R}^k} \|U_{k+1}(B_{k+1,k}y - e_1\|b\|_2)\|_2 \\
&= \min_{y \in \mathbb{R}^k} \|B_{k+1,k}y - e_1\|b\|_2\|_2 \qquad (2.30)
\end{aligned}$$

Questo mostra che l'algoritmo è un metodo iterativo del minimo residuo. Dal fatto che $\mathcal{K}_{k-1}(A^TA, A^Tb) \subset \mathcal{K}_k(A^TA, A^Tb)$ l'errore residuo $r_k = b - Ax_k$ soddisfa

$$\|r_k\|_2 \leq \|r_{k-1}\|_2 \text{ con } k = 1, 2, \ldots$$

Il problema dei minimi quadrati del secondo membro si risolvere semplicemente facendo la fattorizzazione QR di $B_{k+1,k}$ con l'aiuto delle rotazioni di Givens, perciò avremo

$$x_k = V_k y_k \in \mathcal{K}_k(A^TA, A^Tb) \qquad (2.31)$$

Capitolo 3

Metodi di regolarizzazione

In questo capitolo vedremo alcuni metodi per il calcolo di una soluzione approssimata che siano meno sensibili alle perturbazioni dei dati. A questi metodi viene associato il termine regolarizzazione per via del fatto che si forza la regolarità della soluzione approssimata, chiedendo spesso che la soluzione abbia un carattere liscio. in un certo senso. Quasi tutti i metodi trattati prossimamente producono soluzioni che possono essere espresse come SVD filtro della forma

$$x_{reg} = \sum_{i=1}^{n} \varphi_i \frac{u_i^T b}{\sigma_i} v_i \tag{3.1}$$

dove φ_i sono i *fattori filtro* associati con il metodo. Denotando con $\Phi = \text{diag}(\varphi_i)$ allora possiamo riscrivere la (3.1) in forma matriciale

$$x_{reg} = V \Phi \Sigma^{\dagger} U^T b_{err} = A^{\dagger} b_{err}$$

Ricordiamo che facciamo riferimento al sistema (1.10) con $A \in \mathbb{R}^{m \times n}$ e $m \geq n$.

3.1 TSVD

Da ciò che abbiamo detto precedentemente, si evince che gli errori estremamente larghi nella soluzione vengono dal rumore nelle componenti della SVD associata con i più piccoli valori singolari. Ma a quanto pare di qualcuno ci si può fidare, in particolare sono i coefficienti della forma $\frac{u_i^T b_{err}}{\sigma_i} \approx \frac{u_i^T b}{\sigma_i}$ (dove con b intendiamo il termine noto non affetto da errori), che corrispondono ai più larghi valori singolari. Oltretutto possiamo assumere che il termine noto b soddisfi la condizione di Picard (altrimenti non ci sarebbe modo di risolvere i problemi discreti mal posti). In questo modo avremo che le componenti

SVD della soluzione esatta con valore più largo sono quei coefficienti che sono approssimati meglio, perché per indici i piccoli avremo $\frac{u_i^T b_{err}}{\sigma_i} \approx \frac{u_i^T b}{\sigma_i} = v_i^T x$.
Queste considerazioni ci portano direttamente al metodo per calcolare la soluzione approssimata regolarizzata, che semplicemente si baserà sullo scartare quei coefficienti che sono dominati dall'errore. Definiamo così il metodo *Truncated SVD* (TSVD) che ci darà la soluzione x_{TSVD} ottenuta conservando le prime k componenti della soluzione

$$x_{TSVD} = \sum_{i=1}^{k} \varphi_i \frac{u_i^T b_{err}}{\sigma_i} v_i \tag{3.2}$$

- dove k prende il nome di *parametro regolarizzatore* ed è tale che $0 < k < k =$ rango(A). Seguendo la definizione di filtro data per (3.1) abbiamo che

$$\varphi_i = \begin{cases} 0 & i \geq k \\ 1 & i \leq k \end{cases}$$

Il parametro k dovrebbe essere scelto in modo tale che tutti i coefficienti SVD dominati dall'errore siano tralasciati.
Il vantaggio del metodo TSVD è che esso è intuitivo e la soluzione x_{TSVD} è facile da calcolare per differenti parametri di troncamento, una volta che la SVD è stata calcolata. Lo svantaggio, invece, è che esso richiede esplicitamente il calcolo della SVD o almeno i principali k valori e vettori singolari. Questo costo computazionale è troppo dispendioso per problemi in larga scala.

3.2 La regolarizzazione di Tikhonov

Con il filtro TSVD un bel pò di informazione viene persa quindi ci serve un metodo più accurato. Quello che andremo ad illustrare di seguito prende un approccio differente rispetto al metodo TSVD in quanto incorpora la richiesta della regolarità nella formulazione del problema. Traducendo ciò in modo chiaro, la soluzione x_λ è definita come la soluzione del problema

$$\min_x \{ \|Ax - b_{err}\|_2^2 + \lambda^2 \|x\|_2^2 \} \tag{3.3}$$

Qui abbiamo il parametro regolarizzatore positivo λ che controlla il peso tra i due addendi.
Il primo termine ci torna in mente come il classico problema dei minimi quadrati e misura quanto la soluzione predice bene il dato rumoroso b.
Il secondo termine ($\|x\|_2^2$) misura la regolarità della soluzione. La speranza è

che controllare la norma di x possa sopprimere (almeno in parte) le componenti di ampio rumore.
Il coefficiente λ funge da arbitro della disputa e bilancia i due termini. Ovviamente più grande e λ, maggiore è il peso che si da alla minimizzazione della norma di x e di conseguenza alla regolarità della soluzione. Al contrario, più piccolo si sceglie λ, più peso si da all'approssimare i dati rumorosi.
L'obiettivo di tutto ciò è quindi trovare un buon bilanciamento tra questi due termini, trovando un giusto valore di λ tale che la soluzione x_λ sia sufficientemente regolare e allora stesso tempo rispecchi i dati a sufficienza.
Giusto per essere alternativi, possiamo riscrivere il problema (3.3) in un altro modo

$$\min_x \left\| \begin{pmatrix} A \\ \lambda I \end{pmatrix} x - \begin{pmatrix} b_{err} \\ 0 \end{pmatrix} \right\|_2 \tag{3.4}$$

che è chiaramente un problema ai minimi quadrati. L'equazione normale associata prende la forma

$$(A^T A + \lambda^2 I)x = A^T b_{err}$$

Dall'equazione normale ricaviamo $x_\lambda = (A^T A + \lambda^2 I)^{-1} A^T b_{err}$. Per ragioni di efficienza computazionale e stabilità numerica, gli algoritmi per il calcolo delle soluzioni di Tikhonov dovrebbero basarsi sulla formulazione (3.4).
Ora possiamo inserire la decomposizione SVD all'interno della soluzione x_λ e ottenere così

$$\begin{aligned} x_\lambda &= (V\Sigma^2 V^T + \lambda^2 V V^T)^{-1} V\Sigma U^T b_{err} \\ &= V(\Sigma^2 V^T + \lambda^2 V)^{-1} V^T V\Sigma U^T b_{err} \\ &= V(\Sigma^2 V^T + \lambda^2 V)^{-1} \Sigma U^T b_{err} \end{aligned}$$

Quindi l'espressione finale sarà $x_\lambda = V(\Sigma^2 V^T + \lambda^2 V)^{-1}\Sigma U^T b_{err}$.
Se riprendiamo l'equazione generale (3.1) di una soluzione regolarizzata, e guardiamo l'espressione di x_λ possiamo allora introdurre i fattori filtro $\varphi_i^{[\lambda]}$ per $i = 1, \ldots, n$

$$\varphi_i^{[\lambda]} = \frac{\sigma_i^2}{\sigma_i^2 + \lambda^2} \approx \begin{cases} 1 & \sigma_i \gg \lambda \\ \frac{\sigma_i^2}{\lambda^2} & \sigma_i \ll \lambda \end{cases} \tag{3.5}$$

Si può notare come per valori di σ_i più grandi di λ, i fattori filtro sono vicini a uno e quindi conservano le componenti SVD che contribuiscono a rendere la soluzione x_λ regolare. Viceversa per valori di σ_i molto più piccoli di λ, i fattori filtro sono a loro volta piccoli e ciò fa si che le componenti indesiderate di SVD siano filtrate.

Osservazione 3.0.1. Da notare che i fattori filtro $\varphi_i^{[\lambda]}$ sono proporzionali a σ_i^2 e quindi decadono tanto velocemente da sopprimere i fattori incremento $\frac{u_i^T b_{err}}{\sigma_i} \approx \frac{u_i^T \eta}{\sigma_i}$

Mettendo tutto assieme, arriviamo all'espressione della soluzione di Tikhonov

$$x_\lambda = \sum_{i=1}^{n} \frac{\sigma_i}{\sigma_i^2 + \lambda^2}(u_i^T b_{err})v_i \tag{3.6}$$

3.3 Regolarizzazione di Tikhonov e metodo di Golub-Kahan

In questo paragrafo sfrutteremo tutto ciò che è stato scritto fino ad ora, mettendo assieme la decomposizione di Golub-Kahan e la regolarizzazione di Tikhonov. Per problemi di larga scala il calcolo della SVD di una matrice A risulta molto dispendioso. Per questo caso riduciamo il problema dei minimi quadrati ad uno di piccola dimensione, eseguendo solamente pochi passi della bidiagonalizzazione di Golub-Kahan. Riprendiamo le relazioni ricavate precedentemente riguardanti la bidiagonalizzazione di Golub-Kahan al passo k (2.15) e il metodo LSQR (2.30). Al passo k avremo la soluzione approssimata $x_k = V_k y_k$ con y_k soluzione del problema ridotto (2.30). Per non appesantire la notazione denotiamo la matrice bidiagonale $B_{k+1,k}$ del metodo di Golub-Kahan con un unico indice, ossia B_k e definiamo ora

$$\rho_k = \|B_k y - \|b_{err} Vert_2 e_1\|_2 = \|Ax_k - b_{err}\|_2 \tag{3.7}$$

Assumiamo che siano stati effettuati $l \geq k$ passi della bidiagonalizzazione di Golub-Kahan e sostituendo la fattorizzazione all'interno del problema (3.3) avremo

$$\min_{y \in \mathbb{R}^l} \left\| \begin{pmatrix} B_l \\ \lambda I_l \end{pmatrix} y - \|b_{err}\|_2 e_1 \right\|_2 \tag{3.8}$$

che ha unica soluzione $y_{\lambda,l}$ per ogni $\lambda > 0$. Una approssimazione della soluzione x_λ del problema di Tikhonov è fornita da $x_{\lambda,l} = V_l y_{\lambda,l}$.

Capitolo 4

Test numerici

In questo capitolo vengono descritti alcuni esperimenti numerici effettuati per verificare l'efficienza dei metodi su sottospazi di Krylov. Gli algoritmi verranno analizzati generando diversi sistemi lineari dati da differenti scelte della matrice A, a seconda del tipo di algoritmo che si andrà a considerare. Per arrivare al sistema lineare (1.10) abbiamo bisogno di generare per prima cosa la matrice A e per questo può aiutare la funzione **gallery** di Matlab. La sua sintassi è la seguente

$$[\text{A,B,C,...}] = \text{gallery}(matname, P1, P2, ...)$$

Il parametro in input *matname* ci permette di scegliere che tipo di matrice si vuole, infatti la funzione gallery contiene oltre 50 differenti tipi di matrici test utili per testare gli algoritmi. I restanti parametri, $P1$,$P2$, etc. dipendono dalla famiglia di matrici selezionata con il primo input. Sono state prese in considerazione 3 scelte del parametro *matname* e sono le seguenti:

- *parter*
- *orthog*
- *kms*

Di seguito forniamo una breve descrizione delle scelte citate, per maggiori dettagli su diversi tipi di matrici si può consultare la documentazione di Matlab.

parter

sintassi: $A = \text{gallery('parter',}n)$

Descrizione: restituisce una matrice A di dimensione n tale che le sue entrate soddisfino la relazione

$$C(i,j) = \frac{1}{i-j+\frac{1}{2}}.$$

A risulta essere una matrice di Cauchy e di Toepliz e molti dei suoi autovalori sono vicini al valore π

orthog

Sintassi: A = gallery('orthog',n,i)

Descrizione: questa scelta del parametro *matname* ci restituisce una matrice ortogonale di ordine n, che può essere scelta in base al parametro $i \in \{-1,-2,1,2,3,4,5,6\}$. Si prenderà in considerazione la scelta del parametro $i = 2$ che darà una matrice ortogonale e simmetrica.

kms

Sintassi: A = gallery('kms',n,ρ)

Descrizione: questa scelta del parametro *matname* restituisce una matrice quadrata di ordine n di Kac-Murdock-Szego, tale che $A(i,j) = \rho^{|i-j|}$, con ρ reale. La matrice A ha le seguenti proprietà: è definita positiva se e solo se $0 < |\rho| < 1$; la sua inversa è tridiagonale; è una matrice di Toeplitz.

4.1 Risultati

Questo capitolo riporta i risultati ottenuti utilizzando le matrici test descritte nella sezione precedente ai fini di verificare l'efficienza degli algoritmi numerici GMRES, LSQR, Gradiente Coniugato (che per semplicità di notazione denomineremo con GC) e D-Lanczos riportati nel capitolo 2. Ottenute le matrici test descritte prima, si crea il vettore soluzione x tramite la funzione *randn*, che sarà la nostra soluzione esatta x_{LS}, utilizzando la notazione dei capitoli precedenti. Il termine noto b sarà una facile conseguenza del prodotto tra matrice A e vettore x, avendo così tutti gli elementi utili del sistema lineare (1.1). Visto che il termine noto b non è affetto da errore, si andrà ad aggiungere la quantità η per ottenere l'equazione descritta in (1.10). Si è deciso di confrontare gli algoritmi citati misurando sia l'errore relativo rispetto alla soluzione esatta x_{LS} e soluzione approssimata x_*, sia l'andamento del

residuo col passare delle iterazioni. Verranno effettuate diverse misurazioni variando la dimensione della matrice A, tramite l'argomento input n delle funzioni test, e il valore dell'errore η.
Tutti i risultati sono stati ottenuti su un laptop dotato di CPU Intel Core i5-7200U e 4 GB di memoria RAM a disposizione, utilizzando il software MATLAB versione R2018a. Le tabelle 4.1 e 4.2 che si trovano alla fine di questo capitolo e mostrano i risultati ottenuti e i valori utilizzati per n e η oltre al valore k, che indica l'indice di iterazione in cui l'algoritmo trova la miglior soluzione approssimata.

Il primo caso considerato, figure 4.1 e 4.2, è quello relativo alla matrice A generata dalla scelta del parametro *matname* = *parter*. Per questo tipo di matrice potremo considerare solo gli algoritmi GMRES e LSQR in quanto D-Lanzcos e GC si applicano a matrici simmetriche e simmetriche definite positive. Dando uno sguardo non solo alle figure che sono visivamente più efficaci ma anche alle tabelle 4.1 e 4.2, si può notare come, sia in termini di residuo che in termini di errore nella approssimazione della soluzione esatta, l'algoritmo LSQR agisca in maniera più veloce ed efficiente rispetto all'algoritmo GMRES. Considerando la figura 4.1 relativa all'errore nella soluzione, entrambi gli algoritmi arrivano allo stesso errore relativo ma a differenti iterazioni (alla terza per LSQR e quindicesima per GMRES). Due aspetti importanti si possono evidenziare: il primo è che tenendo fisso il parametro η, relativo alla perturbazione nel termine noto, l'errore non cambia al crescere della dimensione della matrice.

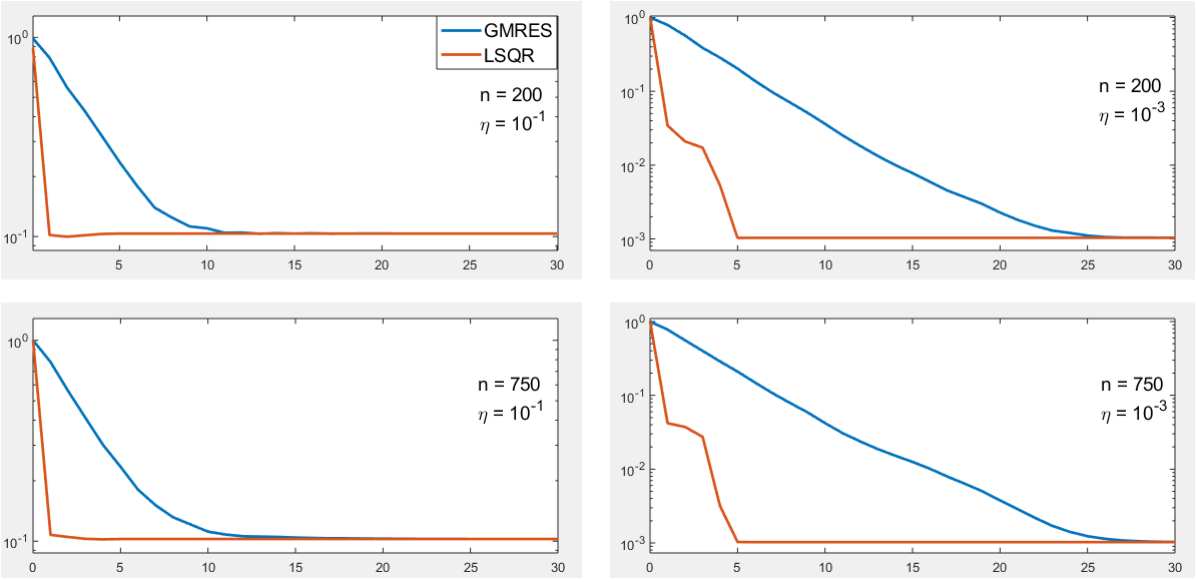

Figura 4.1: Confronto errore relativo, caso parter

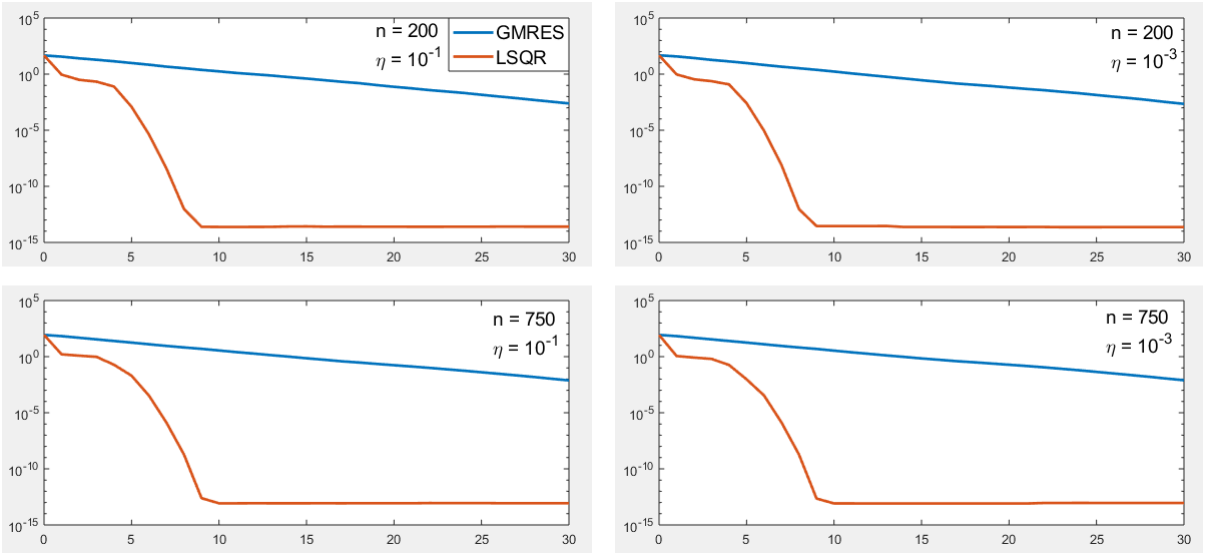

Figura 4.2: Confronto residuo, caso parter

Il secondo aspetto invece riguarda il viceversa, ossia tenendo fissa la dimensione matriciale si osserva come l'errore diminuisca notevolmente di un fattore dell'ordine di 10^{-2}. Per quanto concerne il residuo, risalta subito come l'algoritmo LSQR sia nettamente migliore rispetto al suo antagonista. In termini di valori, controllando la tabella 4.2, si vede come l'errore residuo rimanga sempre dell'ordine di 10^{-14} mentre l'algoritmo GMRES solo alla 30esima iterazione arriva ad un risultato acettabile ma insufficiente se paragonato con il rivale LSQR. Un'altra considerazione che risalta da entrambe le figure è come sia l'algoritmo LSQR sia l'algoritmo GMRES(per questo algoritmo riguarda solo la soluzione approssimata) dopo un tot di iterazioni non migliorino ma tendano asintoticamente ad un valore. Questo mostra come con un opportuno criterio di stop possa interrompere gli algoritmi quando ormai la soluzione approssimata e/o il residuo non migliorano con l'andare delle iterazioni. Sicuramente inserendo una tolleranza tra il valore ottenuto alla k-esima iterazione e la precedente si otterrebbe un algoritmo più efficiente e meno perdita di tempo nel calcolo.

Ora si prenderà in esame il secondo caso scelto che vedrà l'utilizzo dell'algoritmo D-Lanczos, oltre a quelli precedenti. Questo caso riguarda la scelta di *matname* = *orthog*, che genera una matrice simmetrica, oltre che ortogonale. Prendendo in esame le figure 4.3 e 4.4, da una prima occhiata si può constatate come non ci sia una netta prevaricazione di un algoritmo sull'altro. Sicuramente l'algoritmo LSQR ottiene una migliore soluzione approssimata a qualche iterazione precedente rispetto agli altri due algoritmi.

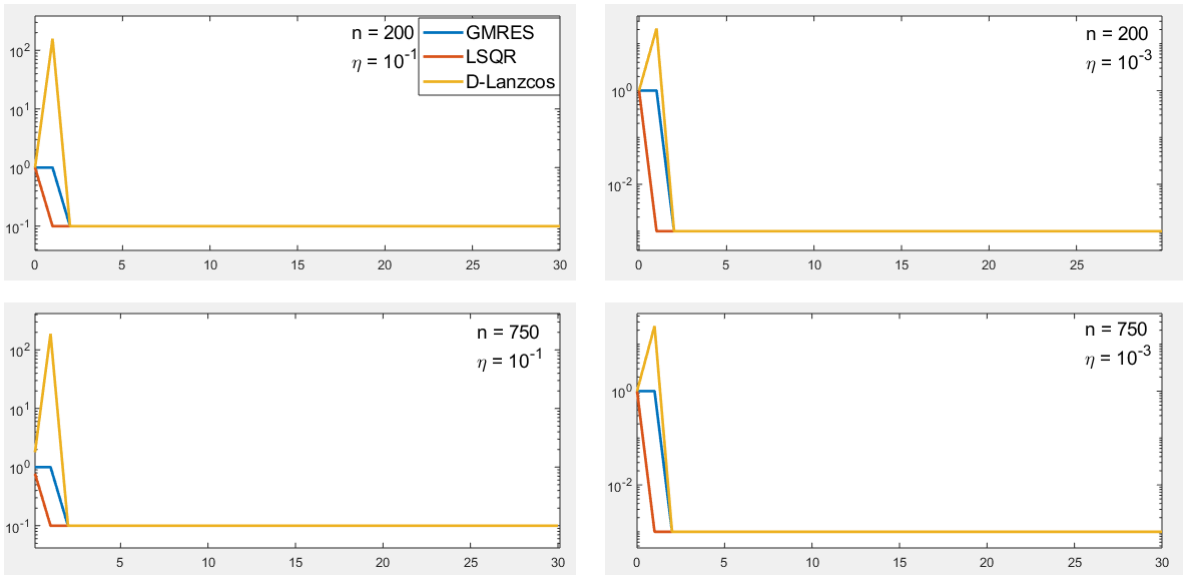

Figura 4.3: Confronto Errore relativo, caso orthog

Dando uno sguardo anche alle tabelle si nota il fatto che, tenendo fisso il valore della perturbazione η, il valore dell'errore relativo non cambi al crescere di n. Invece, cambiando valore di η, si ha un netto miglioramento della soluzione approssimata, di 10^{-2} volte inferiore.

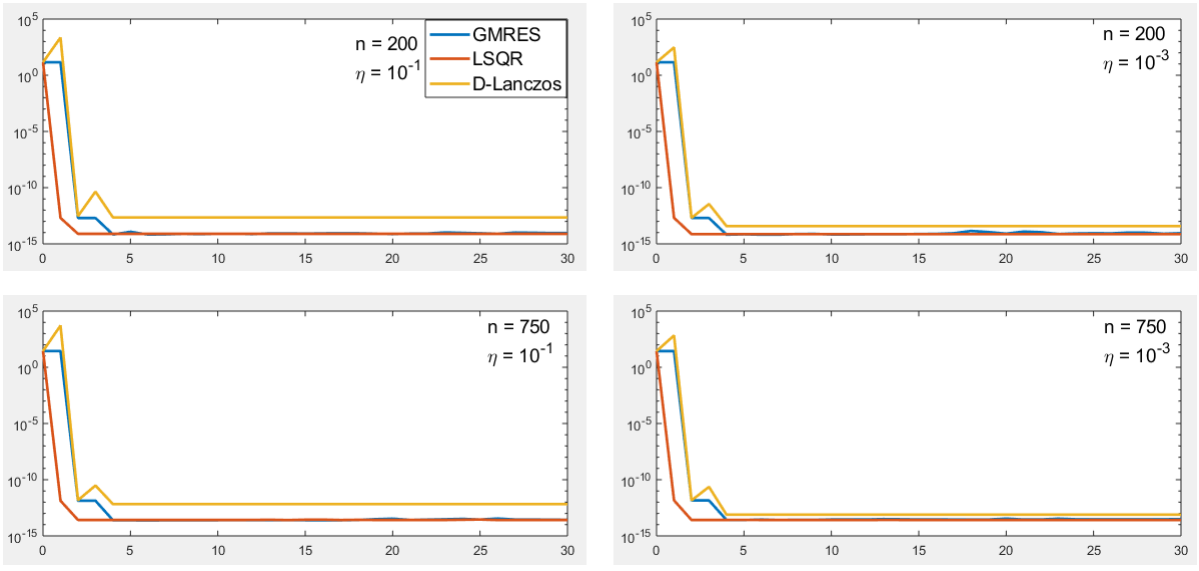

Figura 4.4: Confronto residuo, caso orthog

Questo discorso invece è evidentemente superfluo se si guarda la tabella 4.2 in quanto l'errore nel residuo rimane intorno al 10^{-14} o 10^{-15}, si può evidenziare solo un suo leggero incremento all'aumentare della dimensione. Mentre rimane identica la considerazione fatta per il caso precedente, ossia optare per una differente scelta del criterio di stop, per rendere migliore l'efficienza e il tempo di esecuzione degli algoritmi.

L'ultimo caso da affrontare ci porta ad una matrice A simmetrica e definita positiva attraverso il parametro $matname = kms$. Le figure 4.5 e 4.6, in questa pagina e nella prossima, saranno da guida per l'analisi di tutti gli algoritmi compreso il gradiente coniugato, ultimo rimasto. Il primo aspetto che si nota e che va sottolineato è il fatto che, per quanto riguarda l'errore nel residuo e anche nella soluzione approssimata, gli algoritmi D-Lanzcos e GC abbiano gli stessi identici risultati, senza alcuna minima differenza. A questa osservazione va aggiunto l'algoritmo GMRES che a meno di piccole variazioni appare del tutto identico agli altri due, mentre LSQR impiega più iterazioni per arrivare allo stesso errore relativo. L'errore relativo, figura sottostante, risente della perturbazione nel termine noto b, infatti diminuendo η ma mantenendo la stessa dimensione della matrice A, si passa da un errore di circa 0.2 ad uno 0.002 circa. Viceversa, non si nota nessuna differenza nell'errore all'aumentare della dimensione n, mantenendo fissa la perturbazione .

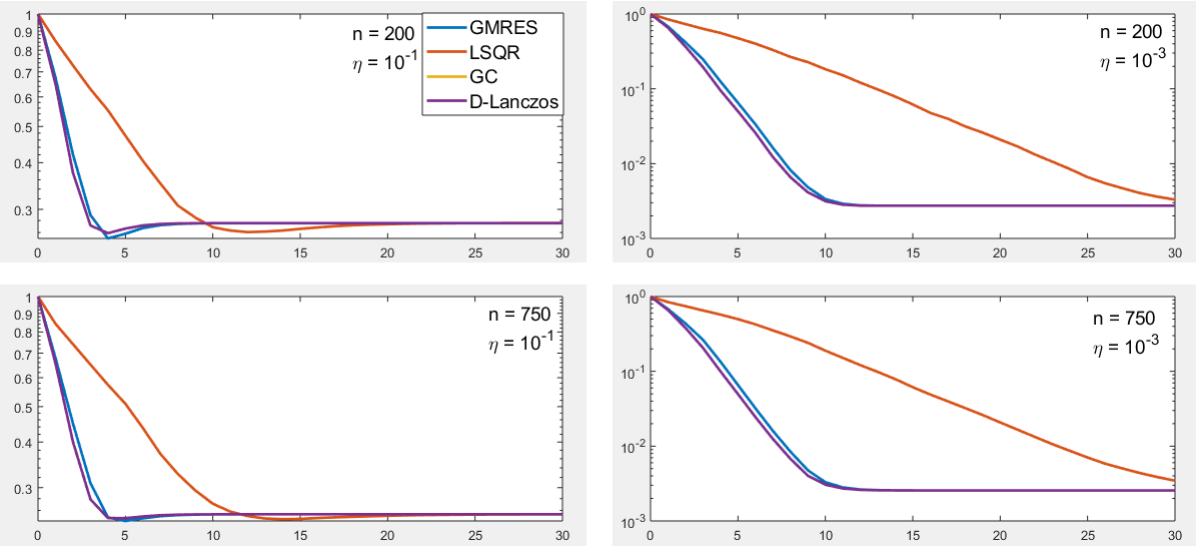

Figura 4.5: Confronto Errore relativo, caso kms

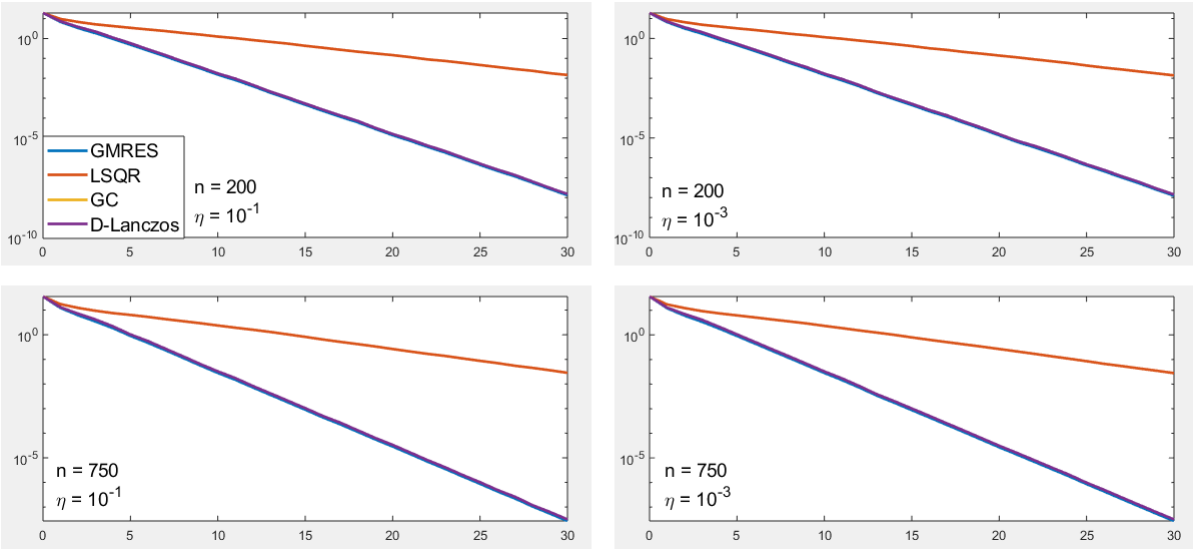

Figura 4.6: Confronto residuo, caso kms

Passando ad analizzare il residuo, il trio di metodi rimane sempre appaiato mentre LSQR si vede che risulta di molto superiore, 10^{-2} contro il 10^{-8} degli altri. Da osservare, però, come con il procedere delle iterazioni tutti e quattro gli algoritmi tendano a migliorarsi diminuendo sostanzialmente l'errore residuo. Infine appare evidente come sia necessario un criterio di stop similmente ai due casi precedentemente trattati e per le stesse motivazioni descritte.

Test	$\boldsymbol{n}$	η	**Errore Relativo $\mathbf{x}_{LS}$ *(k)***			
			GMRES	**LSQR**	**D-Lanczos**	**GC**
parter	100	10^{-1}	0.109(15)	0.109(5)	-	-
		10^{-3}	1.1×10^{-3}(24)	1.1×10^{-3}(5)	-	-
	200	10^{-1}	0.104(12)	0.099 (2)	-	-
		10^{-3}	1.06×10^{-3}(26)	1.03×10^{-3}(5)	-	-
	500	10^{-1}	0.1(15)	0.1 (5)	-	-
		10^{-3}	1.09×10^{-3}(25)	1.09×10^{-3}(5)	-	-
	750	10^{-1}	0.105 (13)	0.102 (3)	-	-
		10^{-3}	1.08×10^{-3}(27)	1×10^{-3}(5)	-	-
orthog	100	10^{-1}	0.1 (2)	0.1 (1)	0.1 (2)	-
		10^{-3}	0.001 (2)	0.001 (1)	0.001 (2)	-
	200	10^{-1}	0.1 (2)	0.1 (1)	0.1 (2)	-
		10^{-3}	0.001 (2)	0.001 (1)	0.001 (2)	-
	500	10^{-1}	0.1 (2)	0.1 (1)	0.1 (2)	-
		10^{-3}	0.001 (2)	0.001 (1)	0.001 (2)	-
	750	10^{-1}	0.1 (2)	0.1 (1)	0.1 (2)	-
		10^{-3}	0.001 (2)	0.001 (1)	0.001 (2)	-
kms	100	10^{-1}	0.23 (7)	0.23(23)	0.23 (7)	0.23 (7)
		10^{-3}	2.29×10^{-3}(12)	2.59×10^{-3}(30)	2.29×10^{-3}(12)	2.29×10^{-3}(12)
	200	10^{-1}	0.25(4)	0.26 (12)	0.259 (4)	0.259 (4)
		10^{-3}	2.74×10^{-3}(12)	3.3×10^{-3}(30)	2.74×10^{-3}(12)	2.74×10^{-3}(12)
	500	10^{-1}	0.246(5)	0.25(15)	0.25(4)	0.25(4)
		10^{-3}	2.25×10^{-3}(12)	3.12×10^{-3}(30)	2.25×10^{-3}(12)	2.25×10^{-3}(12)
	750	10^{-1}	0.243(5)	0.246 (14)	0.247(5)	0.247(5)
		10^{-3}	2.6×10^{-3}(12)	3.45×10^{-3}(30)	2.6×10^{-3}(12)	2.6×10^{-3}(12)

Tabella 4.1: Errore relativo su x

Test	$\boldsymbol{n}$	η	**Residuo** ***(k)***			
			GMRES	**LSQR**	**D-Lanczos**	**GC**
parter	100	10^{-1}	$8.57\times10^{-4}(30)$	$1.27\times10^{-14}(8)$	-	-
		10^{-3}	$9.3\times10^{-4}(30)$	$2\times10^{-14}(8)$	-	-
	200	10^{-1}	$2.44\times10^{-3}(30)$	$2.45\times10^{-14}(9)$	-	-
		10^{-3}	$2.2\times10^{-3}(30)$	$2.86\times10^{-14}(9)$	-	-
	500	10^{-1}	$4.38\times10^{-3}(30)$	$7.12\times10^{-14}(9)$	-	-
		10^{-3}	$4.3\times10^{-3}(30)$	$5.49\times10^{-14}(9)$	-	-
	750	10^{-1}	$7.64\times10^{-3}(30)$	$8.49\times10^{-14}(10)$	-	-
		10^{-3}	$7.89\times10^{-3}(30)$	$8.28\times10^{-14}(9)$	-	-
orthog	100	10^{-1}	$6\ 10^{-15}(4)$	$4.710^{-15}(3)$	$1.9810^{-14}(4)$	-
		10^{-3}	$5.65\ 10^{-15}(4)$	$5.21\ 10^{-15}(3)$	$3.23\ 10^{-13}(4)$	-
	200	10^{-1}	$6.8\ 10^{-15}(4)$	$7.84\ 10^{-15}(2)$	$2.25\ 10^{-13}(4)$	-
		10^{-3}	$6.59\ 10^{-15}(4)$	$7.3\ 10^{-15}(2)$	$3.77\ 10^{-14}(4)$	-
	500	10^{-1}	$1.7\ 10^{-14}(5)$	$1.67\ 10^{-14}(4)$	$7\ 10^{-14}(4)$	-
		10^{-3}	$1.89\ 10^{-14}(4)$	$1.75\ 10^{-14}(4)$	$4.3\ 10^{-14}(4)$	-
	750	10^{-1}	$2.39\ 10^{-14}(4)$	$2.6\ 10^{-14}(4)$	$6.7\ 10^{-13}(6)$	-
		10^{-3}	$2.6\ 10^{-14}(4)$	$2.6\ 10^{-14}(3)$	$7.74\ 10^{-14}(4)$	-
kms	100	10^{-1}	$7.99\times10^{-8}(30)$	$6\times10^{-3}(30)$	$7.99\times10^{-8}(30)$	$7.99\times10^{-8}(30)$
		10^{-3}	$7.98\times10^{-8}(30)$	$6.1\times10^{-3}(30)$	$7.98\times10^{-8}(30)$	$7.98\times10^{-8}(30)$
	200	10^{-1}	$1.52\times10^{-8}(30)$	$0.0146(30)$	$1.52\times10^{-8}(30)$	$1.52\times10^{-8}(30)$
		10^{-3}	$1.43\times10^{-8}(30)$	$0.013(30)$	$1.43\times10^{-8}(30)$	$1.43\times10^{-8}(30)$
	500	10^{-1}	$2.57\times10^{-8}(30)$	$2.3\times10^{-2}(30)$	$2.57\times10^{-8}(30)$	$2.57\times10^{-8}(30)$
		10^{-3}	$2.49\times10^{-8}(30)$	$2.19\times10^{-3}(30)$	$2.49\times10^{-8}(30)$	$2.49\times10^{-8}(30)$
	750	10^{-1}	$3.15\times10^{-8}(30)$	$0.028(30)$	$3.15\times10^{-8}(30)$	$3.15\times10^{-8}(30)$
		10^{-3}	$3.17\times10^{-8}(30)$	$0.0278(30)$	$3.17\times10^{-8}(30)$	$3.17\times10^{-8}(30)$

Tabella 4.2: Norma del residuo

Bibliografia

[1] Yousef Saad, *Iterative Methods for Sparse Linear System*, SIAM (2003)

[2] Per Christian Hansen, *Rank-Deficient and Discrete Ill-Posed Problems (Numerical Aspects of Linear Inversion)*, SIAM (1997)

[3] Per Christian Hansen, *Discrete Inverse Problems (Insight and Algorithms)* , SIAM (2010)

[4] Giuseppe Rodriguez, *Algoritmi numerici*, Pitagora Editrice (2008)

[5] M.E. Hochstenbach, L. Reichel, G. Rodriguez, *Regularization parameter determination for discrete ill-posed problems*, Journal of Computational and Applied Mathematics (2013)

[6] Lothar Reichel, *Numerical Linear Algebra for Ill-Posed Problems*, seminario presso Università degli studi di Cagliari, Giugno 2019

[7] Alessandro Buccini, *Regularization of ill-posed linear inverse problem*, Corso di Dottorato del Dipartimento di Matematica e Informatica di Cagliari, Giugno 2020