Metodi numerici

Appunti lezioni

Giovanni Antonioni

Indice

Capitolo 1

Introduzione

1.1 Analisi numerica.

L'analisi numerica è la materia che si pone come obbiettivo quello di attribuire una risposta di tipo numerico (supportata da un calcolatore) ad un problema matematico che modellizza un problema reale.

Esempio 1.1.1. Calcolare l'area della superficie terrestre

Per procedere al calcolo di tale area occorre iniziare con delle osservazioni. La prima è che la terra, a livello figurativo, può essere rappresentata come una sfera, dato dunque un raggio che possiamo attribuirle ($r \approx 6370km$) adoperiamo la formula dell'area della superficie della sfera:

$$4\pi r^2$$

In questo ragionamento è possibile individuare dei passaggi chiave (Fig 1.1) . Siamo partiti da un problema reale (*Area supericie terrestre*), abbiamo costruito un modello matematico ed infine applicato l'algoritmo per il calcolo dell'area.
Nell'effettuare questi è evidente come vengono compiute diverse approssimazioni. Anzitutto la figura del pianeta è stata **approssimata ad una forma sferica** e, secondo, tramite delle valutazioni empiriche abbiamo attribuito **un valore approssimativo alla lunghezza del raggio**. Adoperando un calcolatore per effettuare le nostre operazioni sappiamo che il valore di π deve essere anch'esso approssimato poiché lo spazio di immagazzinamento delle informazioni che abbiamo a disposizione è ovviamente limitato.
Il risultato ottenuto è dunque un'approssimazione del vero valore dell'area terrestre.

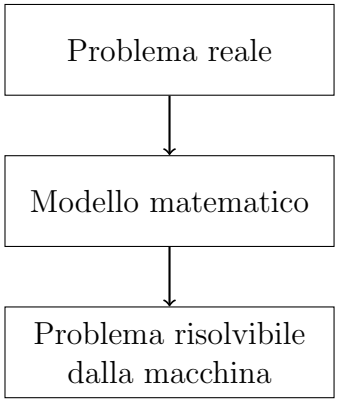


Figura 1.1: Passaggi da problema reale a computazionale

Le considerazioni fatte quà sopra ci permettono dunque di delineare quello che è lo scopo dell' analisi numerica, ovvero :

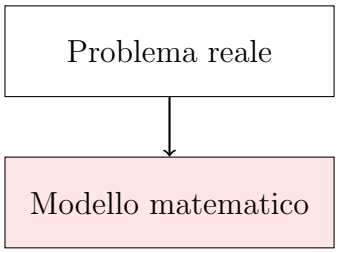
1. Trovare un algoritmo risolutivo ad un problema reale tale per cui questo possa essere calcolato da un calcolatore.

2. Verificare che l'algoritmo del punto precedente fornisca un risultato esatto.

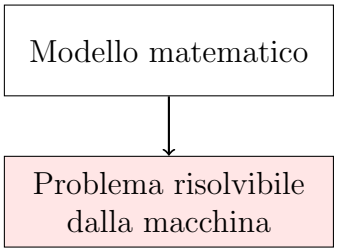
1.1.1 Sorgenti di errore nella risoluzione di problemi matematici al calcolatore.

Quello che vogliamo fare ora è cercare di definire delle classi di errore in cui possiamo ricadere nel tentare di risolvere un determinato problema tramite una computazione effettuata da un calcolatore:

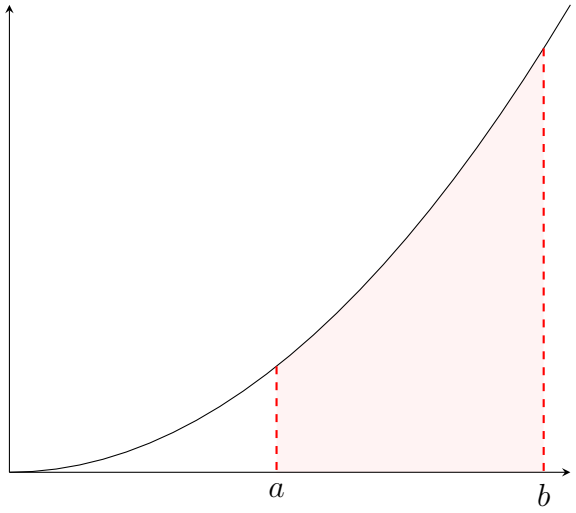
1. **Problemi modello matematico** : Tipologia di errori che vengono commessi durante il passaggio dal problema reale al modello matematico sono causati talvolta dall'introduzione di ipotesi semplificative.



2. **Errori del modello numerico-computazionale** : Tipologia di errori che vengono commessi durante il passaggio dal modello matematico al problema numerico risolvibile dalla macchina. Vengono chiamati anche errori di discretizzazione o troncamento e rappresentano errori introdotti nel sintetizzare un procedimento infinito in uno finito



Esempio 1.1.2. Sia data la seguente funzione $f(x) = x^2$, si voglia trovare l'area della figura sottesa (evidenziata in rosso) tra i punti a e b.



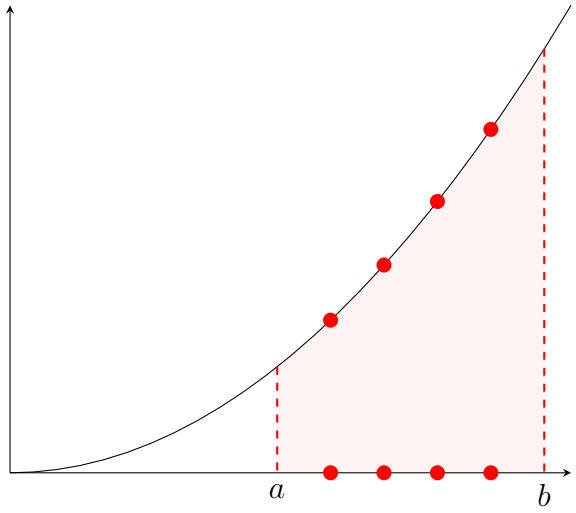
Per risolvere il seguente problema abbiamo bisogno di calcolare l'integrale :

$$\int_a^b f(x)dx$$

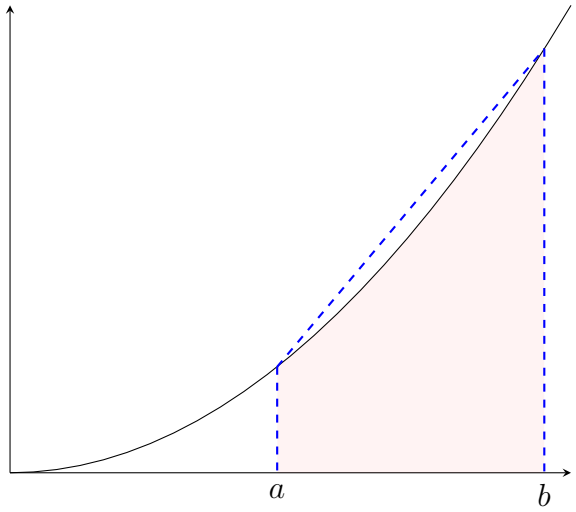
Adoperando il calcolatore per effettuare le operazioni possiamo utilizzare delle approssimazioni denominate *formule di quadratura*:

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} w_i f(x_i)$$

dove $x_1, x_2, .., x_n \in [a, b]$.



Alternativamente possiamo utilizzare un'ulteriore formula di quadratura che ci permette di approssimare l'area sottesa tra a e b con l'area di trapezio descritto in questo modo :



La formula dell'area risultante sarà:

$$\int_a^b f(x)dx \approx \frac{(f(a) - f(b))(b - a)}{2}$$

Questi non sono altro che esempi di introduzione di sorgenti di errore nel passaggio dal modello matematico al modello computazionale.

3. **Errori nei dati** : Sono errori presenti all'interno dei dati su cui stiamo lavorando e che possono essere dovuti dalla sensibilità dello strumento che adoperiamo per le misurazioni (*Errori sistematici*) o dovuti da fenomeni non prevedibili.

4. **Errori di arrotondamento**.

1.1.2 Classificazione dei problemi di tipo numerico-computazionale.

Definizione 1.1.1. Si definisce **Problema numerico** una descrizione chiara e non ambigua di una determinata relazione funzionale ϕ tra dei dati in ingresso e dei risultati (o anche detti dati di uscita).

Siano dati x, y come dati di ingresso e uscita del problema (non necessariamente in quest'ordine). Nell'effettuare il passaggio dal modello matematico al numerico computazionale possiamo incontrare tre tipologie di problemi differenti:

1. **Problemi diretti** : tipologie di problemi in cui siamo a conoscenza di x, ϕ e vogliamo trovare y

   Esempio 1.1.3. E' il calcolo di un valore delle ordinate di una funzione dato un valore delle ascisse :

   $$f(x) = y$$

2. **Problemi diretti** : tipologie di problemi in cui siamo a conoscenza di y, ϕ e vogliamo trovare x

   Esempio 1.1.4. La risoluzione di un sistema lineare è un esempio di problema inverso:

   $$A_{n\times n}x_{1\times n} = y_{n\times 1}$$

   conosciamo la matrice dei coefficienti A che rappresenta la nostra relazione funzionale ϕ. Conosciamo inoltre i nostri termini noti dati dalla variabile y e vogliamo trovare il vettore incognito x.

3. **Problemi d'identità** : tipologie di problemi in cui siamo a conoscenza di x, y e vogliamo trovare ϕ.
   Come esempio di quest'ultima categoria di problemi vogliamo presentare un problema di approssimazione :

   Esempio 1.1.5. Sia data una serie di n punti (x_i, y_i) $\forall i = 1, 2, ...n$ e con le ascisse tutte distinte.
   Vogliamo individuare se $\exists p$ polinomio di grado $n-1$ tale che $p(x_i) = y_i$ $\forall i = 1, 2, ...n$.
   Se $n = 3$ per, esempio allora p è una parabola denominata, in questo caso, polinomio di interpolazione dei dati.
   E' da notare in questo caso che io conosco sia il dato x (corrispondente alle ascisse) e il dato y (corrispondente alle ordinate) e ne voglio trovare una relazione ϕ che in questo caso corrisponde al polinomio p di grando $n-1$.

1.1.3 Problemi ben posti e ben condizionati

In questo corso ci interesserà andare a cercare di risolvere dei problemi definiti **ben posti**.

Definizione 1.1.2. Si definisce **problema ben posto** una particolare tipologia di problema la quale soluzione è caratterizzata da queste 3 proprietà:

1. Esiste.
2. E' unica.
3. Dipende in modo continuo dai dati del problema.

Se un problema non è ben posto allora si dice che questio sia **mal posto**. I problemi che andremo a risolvere sul calcolatore saranno sia **ben posti che ben condizionati**

Definizione 1.1.3. Si definisce **problema ben condizionato** un problema nel quale a piccole perturbazioni su dati corrispondono piccole perturbazioni sui risultati ottenuti.

Capitolo 2

Numeri finiti e aritmetica della macchina

2.1 La notazione posizionale

Trattando problemi matematici di tipo aritmetico ci viene naturale rappresentare un determinato valore in base decimale. Questo è dovuto alla semplicità con la quale possiamo usare i valori ed effettuare le operazioni; tuttavia in ambito informatico i calcolatori, per effettuare i loro controlli e le loro operazioni adottano una rappresentazione binaria per i valori. Abbiamo bisogno, dunque, di una metodologia per effettuare agilmente dei cambi di base.
Data β una base :

$$(a_n....a_0.b_1b_2....)_\beta = \sum_{k=0}^{n} a_k\beta^k + \sum_{k=1}^{\infty} b_k\beta^{-k}$$

Esempio 2.1.1. La rappresentazione del numero 56.78 in base $\beta = 10$:

$$56.78 = 6 \times 10^0 + 5 \times 10^1 + 7 \times 10^{-1} + 8 \times 10^{-2}$$

2.1.1 La forma normalizzata

Scelta una base β vogliamo ora vedere come possiamo rappresentare un qualsiasi $x \in \mathbb{R} \setminus \{0\}$.

Definizione 2.1.1. Un numero x si dice **rappresentato in forma normalizzata** se, scelta una base $\beta \geq 2$, questo si presenta nella forma:

$$\pm(0.d_1d_2d_3....)_\beta \times \beta^p.$$

Dove $1 \leq d_1 \leq \beta - 1$ e $0 \leq d_i \leq \beta \ \forall i > 1$

$0.d_1d_2d_3....$ è detta *mantissa* del numero e questa può assumere valori compresi tra β^{-1} e 1.

2.2 L'insieme dei numeri finiti

Come ormai detto più volte uno dei principali limiti nel rappresentare dei numeri reali all'interno del calcolatore è il fatto che questo possiede un quantitativo di memoria finito.
Adottando la notazione posizionale in forma normalizzata per la rappresentazione dei nostri numeri definiamo ora un insieme $F(\beta, t, L, U)$ dipendente da 4 parametri:

1. β: base usata.
2. t: è il numero di cifre di cui si compone la mantissa $\pm(0.d_1d_2d_3...d_t)$.
3. L: *Lower bound* dell'esponente p. L si intende come intero negativo.
4. U: *Upper bound* dell'esponente p. In questo caso U è inteso essere un intero positivo.

Possiamo rappresentare F come:

$$F(\beta, t, L, U) := \left\{ x \in \mathbb{R} \setminus \{0\} : x = \pm\left(\sum_{i=1}^{t} d_i \beta^{-i} \right) \beta^p \right\} \cup \{0\}$$

F è detto *Insieme dei numeri finiti* o anche *insieme numeri macchina* e possiamo rappresentarlo graficamente nella seguente maniera:

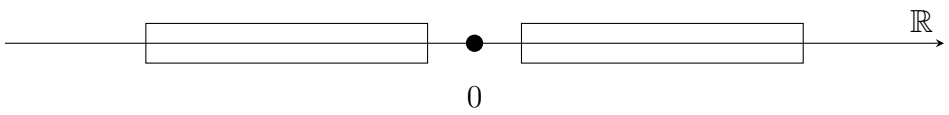


Figura 2.1: Rappresentazione grafica di F

I due riquadri comprendono i numeri positivi e negativi appartenenti ad F, come è possibile notare questi sono speculari tra loro. Lo 0 da definizione è compreso in F ed è rappresentato all'interno della retta.
Ovviamente il tentare di rappresentare numeri $x \notin F$ può comportare una serie di errori:

1. **Overflow**: Può essere di tipo positivo o negativo in base al segno di x. Si tratta del caso in cui il valore $|x|$ è maggiore del massimo numero rappresentabile in F

2. **Underflow**: Errore opposto all'overflow: $|x|$ è minore del più piccolo numero rappresentabile in F. Anche in questo caso può avere tipologia positiva o negativa.

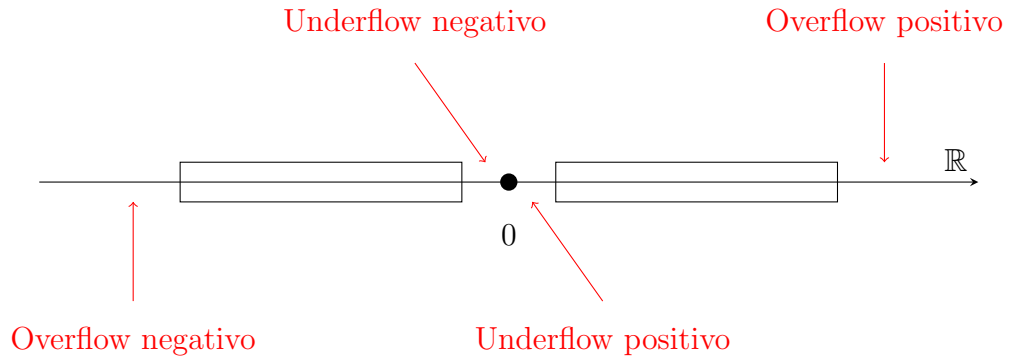


Figura 2.2: Tipologie di errori di rappresentazione in F

E' importante notare che gli intervalli rappresentati dai due rettangoli nelle figure 2.1 e 2.2 presentano dei "buchi" al loro interno . Due numeri $x_1, x_2 \in F$ e contigui tra loro presentano uno spazio in cui ricadono tutti quei valori non appartenenti ad F.

Esempio 2.2.1. Si rappresenti $F(2, 3, -2, 1) := \{\pm 0.1 d_1 d_2 \times \beta^p\} \cup \{0\}$ dove $d_i \in \{0, 1\}$ e $-2 \leq p \leq 1$.

Notiamo subito che, per essere in forma normalizzata, il primo numero dopo la virgola deve essere necessariamente diverso da 0. questo impone, se $\beta = 2$ che la prima cifra dopo la virgola sia un 1, quindi le uniche cifre che varieranno saranno due (ovvero d_1 e d_2).

m \ p	-2	-1	0	1
100	0.100×2^{-2}	0.100×2^{-1}	0.100×2^{0}	0.100×2^{1}
101	0.101×2^{-2}	0.101×2^{-1}	0.101×2^{0}	0.101×2^{1}
110	0.110×2^{-2}	0.110×2^{-1}	0.110×2^{0}	0.110×2^{1}
111	0.111×2^{-2}	0.111×2^{-1}	0.111×2^{0}	0.111×2^{1}

Possiamo subito vedere come i numeri positivi in F siano 16, sapendo che i numeri negativi sono speculari e che $0 \in F$ allora possiamo dedurre che la cardinalità di questo insieme è $16 + 16 + 1 = 33$.
Convertiamo ora i valori in base 10 e facciamo una considerazione:

m \ p	-2	-1	0	1
100	$\frac{4}{32}$	$\frac{4}{16}$	$\frac{4}{8}$	$\frac{4}{4}$
101	$\frac{5}{32}$	$\frac{5}{16}$	$\frac{5}{8}$	$\frac{5}{4}$
110	$\frac{6}{32}$	$\frac{6}{16}$	$\frac{6}{8}$	$\frac{6}{4}$
111	$\frac{7}{32}$	$\frac{7}{16}$	$\frac{7}{8}$	$\frac{7}{4}$

Ogni colonna, come abbiamo visto, rappresenta una potenza di p che eleva β; se provassimo a sottrarre tra loro due numeri consecutivi di una stessa potenza p noteremo che per la prima colonna il valore ottenuto sarebbe $\frac{1}{32}$, per la seconda $\frac{1}{16}$, per la terza $\frac{1}{8}$ e per l'ultima $\frac{1}{4}$.
Questi valori ottenuti rappresentano quello che è denominato *spacing*.
Lo spacing tra due numeri della stessa potenza diminuisce tanto più che un valore si avvicina allo 0.

2.3 La cardinalità di F

Tramite l'esempio 2.2.1 svolto precedentemente abbiamo visto come sia possibile contare il numero di elementi di un insieme F dati dei parametri β, t, L, U rappresentativi. In questo caso però il numero di valori contenuti al suo interno era abbastanza piccolo per essere contato manualmente effettuando alcuni accorgimenti.
Vogliamo astrarre i procedimenti adoperati chiedendoci come si possa calcolare la cardinalità di un generico insieme F.

Sempre riferendoci alla figura 2.1 ricordiamo che l'insieme dei numeri positivi è speculare rispetto l'insieme dei numeri negativi quindi occorrerà calcolare la cardinalità di solo uno di questi.
Per iniziare contiamo il numero di valori positivi di F.
La prima cosa che vogliamo capire è quale sia il massimo e il minimo numero in valore assoluto che possiamo rappresentare in F.

- Il numero più piccolo che possiamo rappresentare è $0.1d_1d_2.....dt \times \beta^L$ dove $\forall i \ d_i = 0$, quindi : $\beta^{-1} \times \beta^L = \beta^{L-1}$
- il numero più grande che possiamo rappresentare invece è $0.d_1d_2.....dt \times \beta^U$ dove $\forall i \ d_i = \beta - 1$.

$$(\beta - 1) \times \beta^{-1}(\beta - 1) \times \beta^{-2}...(\beta - 1) \times \beta^{-t} \times \beta^U$$

$$(\beta - 1)\beta^{-t}(1 + \beta + \beta^2 + ... + \beta^{t-1})\beta^U$$

Facciamo ora un osservazione :

$$(1 + \beta + \beta^2 + ... + \beta^{t-1}) = \frac{1 - \beta^t}{1 - \beta}$$

Poichè :

$$(1-\beta)(1+\beta+\beta^2+.....+\beta^t) = (1+\beta+\beta^2+.....+\beta^{t-1})-(\beta+\beta^2+.....+\beta^{t-1}+\beta^{t-1})$$

Tutti gli elementi uguali si eliminano fra loro facendo rimanere all'interno dell'espressione solamente il valore $1 - \beta^t$. Possiamo sostituire questo risultato nella formula precedente:

$$(\beta - 1)\beta^{-t}\frac{1 - \beta^t}{1 - \beta}\beta^U = (\beta^t - 1)\beta^{-t}\beta^U = (1 - \beta^{-t})\beta^U$$

Quindi il massimo valore in F è $(1 - \beta^{-t})\beta^U$

Importante: $\beta^U \notin F \ \beta^L \in F$.

Ora ci concentriamo nel capire quanti numeri siano compresi tra il massimo e il minimo.
Il primo passo che facciamo è osservare quanti segmenti del tipo $[\beta^p, \beta^{p+1}]$ sia possibile individuare in un determinato sottoinsieme (positivo o negativo) di F.

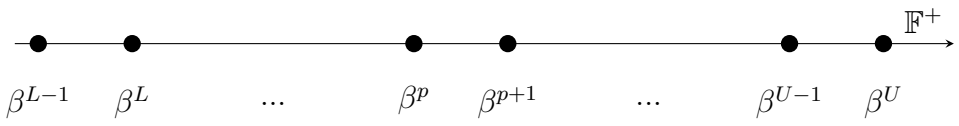


Figura 2.3: Segmenti tra più potenze successive di F^+

Il nostro esponente p può assumere un valore massimo U ed uno minimo L il numero di intervalli che avremmo sarà quindi $U - L + 1$ dove l'aggiunta di 1 permette di comprendere anche $[\beta^{L-1}, \beta^L]$. Internamente ad ogni

intervallo tra due potenze successive di β vi è una stessa quantità di numeri macchina. Questi sono equispaziati tra loro di un valore variabile (di segmento in segmento) s detto spacing. Calcoliamo s portando la nostra attenzione ad un generico segmento $[\beta^p, \beta^{p+1}]$.
Il numero macchina più piccolo in questo intervallo è uguale a $x_1 = 0.1000000...0000 \times \beta^{p+1}$, il suo successivo, invece, è uguale alla stessa cifra con l'unica differenza che la mantissa è incrementata di uno alla t-esima posizione: $x_2 = 0.1000000...0001 \times \beta^{p+1}$. La differenza tra questi due valori ci permette di ricavare lo spazio s che intercorre tra loro:

$$x_2 - x_1 = 0.1000000...0001 \times \beta^{p+1} - 0.1000000...0000 \times \beta^{p+1} = \beta^{-t}\beta^{p+1} = \beta^{p-t+1}$$

Il quantitativo di numeri macchina compresi in questo intervallo è :

$$\frac{\beta^{p+1} - \beta^p}{\beta^{p-t+1}} = (\beta - 1)\beta^{t-1}$$

Finalmente possiamo calcolare la cardinalità del nostro insieme F tenendo conto dei passaggi fatti fin'ora:

$$|F(\beta, t, L, U)| = 2(\beta - 1)\beta^{t-1}(U - L + 1) + 1$$

2.4 Approssimazione

Vogliamo porci ora il seguente problema:
Ipotizziamo di aver definito un insieme F con degli opportuni parametri β, tL, U. Dato un valore $x \notin F$ vogliamo ottenere un valore $x_i \approx x$ tale che $x_i \in F$.
Quello che dobbiamo fare in questo caso è effettuare delle approssimazioni tali da trasformare il nostro numero in partenza in un determinato valore rappresentabile internamente ad F.
Abbiamo due possibili soluzioni differenti che ci permettono di proseguire:

2.4.1 Approssimazione per troncamento

sia $x = \pm(d_1 d_2 .... d_t d_{t+1} ....) \times \beta^p$ definiamo come approssimazione per troncamento :

$$fl_T(x) = \pm(d_1 d_2 .... d_t) \times \beta^p.$$

Se il nostro valore x ha un numero di cifre dopo la virgola maggiore del t semplicemente quelle in eccesso vengono scartate

Esempio 2.4.1. Se definiamo F con $\beta = 10$ e $t = 4$ e vogliamo approssimare il valore $x = 0.143578$ per troncamento il risultato che otterremmo sarà $fl_T(x) = 0.1435$

2.4.2 Approssimazione per arrotondamento

In questo caso richiediamo che la base in cui lavoriamo sia pari. sia $x = \pm(d_1 d_2 .... d_t d_{t+1} ....) \times \beta^p$ definiamo come approssimazione per arrotondamento :

$$fl_A(x) = \begin{cases} fl_T(x) & 0 \leq d_{t+1} \leq \frac{\beta}{2} \\ fl_T(x) + \beta^{p-t} & \frac{\beta}{2} \leq d_{t+1} \leq \beta - 1 \end{cases}$$

si aggiunge al numero x il valore $\frac{\beta}{2}\beta^{-(t+1)}$ poi si effettua l'approssimazione per troncamento al risultato ottenuto.

- $\beta = 10$, $t = 4$, $x = 0.3798165$, $fl_A(x) = 0.3798$
- $\beta = 10$, $t = 4$, $x = 0.1265873$, $fl_A(x) = 0.1266$

2.4.3 Approssimazione per arrotondamento ai pari

Questo è un arrotondamento che viene effettuato in casi speciali quando un numero x risiede esattamente tra due numeri macchina consecutivi.
Abbiamo quindi che : $d_{t+1} = \frac{\beta}{2}$ e $d_i = 0 \ \forall i > t + 1$; l'approssimazione che si effettua è portare x al numero macchina pari più vicino.

- $\beta = 10$, $t = 2$, $x = 0.185$, $fl_A(x) = 0.18$
- $\beta = 10$, $t = 4$, $x = 0.37975$, $fl_A(x) = 0.3798$
- $\beta = 2$, $t = 4$, $x = 0.101110$, $fl_A(x) = 0.1100$

2.5 Errori di approssimazione

2.5.1 Precisione macchina e roundoff unit

Per introdurre il concetto relativo agli errori è opportuno iniziare con qualche definizione :

Definizione 2.5.1. Si definisce **precisione macchina** lo spacing relativo al segmento $[\beta^0, \beta^1]$ il cui valore è $eps = \beta^{1-t}$.

Definizione 2.5.2. Si definisce **roundoff unit** la metà della precisione macchina : $u = \frac{1}{2}eps = \frac{1}{2}\beta^{1-t}$.

Ragioniamo nel concreto cosa questi valori vogliano significare nel nostro studio.
Partendo dalla prima definizione portiamoci nel segmento $[\beta^0, \beta^1]$, avendo definito $eps = \beta^{1-t}$ lo spacing di questo intervallo possiamo subito capire che la distanza tra due numeri macchina consecutivi in questo è proprio eps. Per esempio, se volessimo conoscere il numero macchina immediatamente successivo ad 1 (β^0) occorrerebbe semplicemente sommare ad 1 il valore eps.
Cosa accade quando si aggiunge ad 1 il nostro valore u rappresentante la roundoff unit?
$1 + u \notin F$ quindi occorrerà approssimare il nostro risultato ottenuto ad 1 o $1 + eps$. In questo caso sappiamo che il valore 1 è un valore che ha nella mantissa dei valori uguali a 0 eccetto per la prima cifra subito dopo la virgola : $0.1d_1d_2d_3...d_t \ \forall i \ d_i = 0$ Il successivo macchina invece avrà una mantissa uguale fatta eccezione per la t-esima cifra la quale possiede un 1 come valore. Dovendo approssimare questo risultato avremmo che, tramite la regola del rouding to even, il nostro valore approssimato risulterà proprio 1.

$$fl_A(1 + u) = 1$$

La roundoff unit ci ritornerà utile sopratutto nel dimostrare che l'errore massimo che posso commettere nell'approssimare un numero reale in un numero macchina è inferiore o uguale a u stessa.

2.5.2 Errore relativo ed errore assoluto

Sia dato un valore $x \in \mathbb{R}$ con $x \neq 0$ e sia $fl(x)$ una sua qualsiasi approssimazione.

Definizione 2.5.3. Si definisce errore assoluto il valore:

$$E_{ass} := |x - fl(x)|$$

Definizione 2.5.4. Si definisce errore relativo il valore:

$$E_{rel} := \frac{E_{ass}}{|x|} = \left| \frac{x - fl(x)}{x} \right|$$

Entrambe le definizioni date permettono di individuare un errore commesso durante l'approssimazione di un numero reale in un numero macchina, tuttavia queste presentano delle differenze sostanziali tra loro: Anzitutto possiamo notare come l'errore relativo non venga influenzato dall'ordine di grandezza del numero che va ad approssimare mentre l'errore assoluto, al contrario, si.
L'errore relativo da delle maggiori indicazioni sulla tipologia di approssimazione che si è effettuata nella mantissa. Si vuole ora procedere con un osservazione:

2.5.3 Errori floating per troncamento

Immaginiamo di voler rappresentare un determinato numero x_1 tramite rappresentazione per troncamento. Riprendendo quanto fatto in precedenza tal numero, non ancora approssimato, si presenta in questa forma :

$$x = 0.d_1d_2...d_td_{t+1}... \times \beta^p$$

Effettuando l'approssimazione per troncamento rimuoviamo dalla mantissa le cifre decimali in eccesso riducendole esattamente a t:

$$\widetilde{x} = 0.d_1d_2...d_t \times \beta^p$$

Calcoliamo ora l'errore commesso :

$$E^T_{ass} = |x - \widetilde{x}| = 0.d_{t+1}d_{t+2}... \times \beta^{p-t} < 1 \Longrightarrow E^T_{ass} < \beta^{p-t}$$

$$E^T_{rel} = \frac{E^T_{ass}}{|x|} < \frac{\beta^{p-t}}{\beta^{p-1}} = \beta^{1-t} = esp$$

essendo infatti $x = 0.d_1d_2...d_td_{t+1}... \geq \beta^{-1}$ e quindi $|x| \geq \beta^{p-1}$ passando ai reciproci avremmo che : $\frac{1}{|x|} \leq \frac{1}{\beta^{p-1}}$

2.5.4 Errori floating per arrotondamento

Abbiamo in questo caso un valore $x \notin F$ il quale risiede nello spacing tra due numeri macchina consecutivi $\widetilde{x_1}, \widetilde{x_2}$

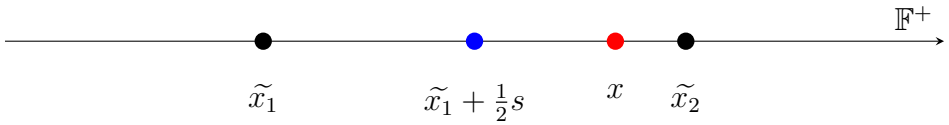


Figura 2.4: Floating per arrotondamento

Vogliamo adoperare il floating per arrotondamento in modo da trasformarlo in un numero che risiede all'interno di F. Il risultato ottenuto da questa approssimazione, per quanto visto in precedenza, è il numero macchina più vicino a x. Prendendo come riferimento la figura 2.4 avremmo $fl_A(x) = \widetilde{x_2}$. Graficamente è possibile da subito notare come questo risultato non ci sorprenda, il nostro numero x si trova a destra del valore centrale dello spacing tra $\widetilde{x_1}$ e $\widetilde{x_2}$. L'errore massimo assoluto commesso dal floating per arrotondamento di un determinato valore x è minore di $\frac{1}{2}\beta^{p-t} = \frac{1}{2}s$. Calcoliamo ora l'errore relativo :

$$E^A_{rel} = \frac{E^A_{ass}}{|x|} < \frac{\frac{1}{2}\beta^{p-t}}{\beta^{p-1}} = \frac{1}{2}\beta^{p-t-p+1} = u$$

Per quanto osservato in precedenza, inoltre, possiamo vedere come il floating per arrotondamento sia, rispetto il floating per troncamento, più preciso in quanto :

$$E^T_{rel} < esp$$
$$E^A_{rel} < u$$
$$E^A_{rel} < E^T_{rel}$$

2.5.5 Il significato della roundoff unit

Come visto precedentemente il valore della roundoff unit rappresenta un limite superiore all'errore relativo che si può commettere approssimando i numeri in virgola mobile.
Essendo u vincolata ad un valore relativo possiamo trascurare l'ordine di grandezza e concentrarci sui numeri compresi nel segmento $[\beta^0, \beta 1]$. Come visto precedentemente il valore massimo dell'errore assoluto, e quindi del numeratore per calcolare l'errore relativo, in caso di floating per arrotondamento è la metà dello spacing. Al denominatore invece è presente

il numero x che occorre approssimare e che, per rendere massimo l'errore relativo, dev'essere il più piccolo possibile. In caso quindi di numeri compresi nel segmento $[\beta^0, \beta 1]$ il massimo valore che possiamo avere per l'errore relativo è quando approssimiamo un valore nella forma : $x = 1 + a$ con $0 < a < \frac{1}{2}\beta^{1-t}$. Tramite floating il valore x viene approssimato ad 1 e quindi:

$$E_{rel} = \left|\frac{x - fl(x)}{x}\right| = \left|\frac{a}{1+a}\right| < \left|\frac{u}{1+a}\right| < u$$

2.6 Operazioni di macchina

Per iniziare a parlare delle operazioni elementari che possono essere eseguite all'interno della macchina vogliamo anzitutto introdurre una nuova notazione :

$$\varepsilon_x = \frac{x - fl(x)}{x} \quad |\varepsilon_x| < u$$

Questa ci permette di mettere in relazione il valore approssimato $fl(x)$ con l'effettivo valore x poiché :

$$\varepsilon_x x = x - fl(x) \Longrightarrow fl_A(x) = x(1 - \varepsilon_x)$$

Il quale ritornerà utile nel parlare di errori commessi nel calcolare il risultato di una specifica operazione.

2.6.1 Calcolo di un'operazione di macchina generica

Si ipotizzi di avere due valori $x, y \in \mathbb{R} \setminus \{0\}$ ed un operatore generico $\cdot \in \{+, -, \times, \div\}$ e si voglia calcolare il risultato dell'operazione $x \cdot y$. Il primo passaggio che effettuiamo è quello di capire se x ed y appartengano all'insieme dei numeri di macchina F. In caso questi non vi appartenessero quello che dobbiamo fare è sicuramente utilizzare le operazioni di floating:

$$x \longrightarrow fl_A(x) \in F,$$

$$y \longrightarrow fl_A(y) \in F$$

A questo punto possiamo calcolare il risultato della nostra operazione con i nuovi valori ottenuti $fl_A(x) \cdot fl_A(y)$ tuttavia non sappiamo se il risultato ottenuto appartenga ad F quindi in questo caso dobbiamo effettuare un'ulteriore approssimazione di questo:

$$fl_A(fl_A(x) \cdot fl_A(y)) \in F$$

Fatto ciò calcoliamo il valore dell'errore relativo tra il risultato esatto tra i due numeri x, y e le loro approssimazioni:

$$E_{rel} = \left| \frac{x \cdot y - fl_A(fl_A(x) \cdot fl_A(y))}{x \cdot y} \right|$$

Cerchiamo di riscrivere ora l'equazione trattata sopra ricordando :

$$fl_A(x) = x(1 - \varepsilon_x) \quad |\varepsilon_x| \leq u$$

$$fl_A(y) = y(1 - \varepsilon_y) \quad |\varepsilon_y| \leq u$$

sia $r := x(1 - \varepsilon_x) \cdot y(1 - \varepsilon_y)$ allora $fl_A(r) = r(1 - \varepsilon_r)$. L'equazione iniziale si trasforma in :

$$E_{rel} = \left| \frac{x \cdot y - (x(1 - \varepsilon_x) \cdot y(1 - \varepsilon_y))(1 - \varepsilon_r)}{x \cdot y} \right| \quad |\varepsilon_x|, |\varepsilon_y|, |\varepsilon_r| \leq u$$

Dove $(1 - \varepsilon_x)$ e $(1 - \varepsilon_y)$ sono gli errori commessi nell'approssimare i valori x e y mentre $(1 - \varepsilon_r)$ rappresenta l'errore di approssimazione del risultato ottenuto.

2.6.2 Calcolo operazioni

Di seguito viene definito come effettuare le basiche operazioni algebriche (somma e moltiplicazione) all'interno dell'insieme dei numeri macchina F.

Somma algebrica: Siano dati due numeri $x, y \in F$. Per effettuare l'operazione $x + y$ procediamo nella seguente maniera:

- Si rappresentano i due numeri **in forma normalizzata**.
- Si prende il numero tra x e y che presenta l'esponente minore e lo si trasforma in modo che entrambi i valori presentino lo stesso esponente. Ovviamente in questo passaggio il numero trasformato perde la forma normalizzata.
- Le mantisse vengono sommate tra loro (gli esponenti vengono lasciati invariati tra di loro
- Il risultato approssimato tramite le varie tecniche di floating mostrate in precedenza.

Vediamo di seguito un esempio di questi passaggi:

Esempio 2.6.1. L'insieme F sul quale lavoriamo è definito nella seguente maniera: $F(10, 2, -3, 2)$. Sia $x = 0.29 \times 10^0$ e $y = 0.25 \times 10^1$; è evidente che $x, y \in F$. Effettuiamo i passaggi elencati qui sopra :

- I due numeri sono espressi già in forma normalizzata, non occorre effettuare alcuna correzione sotto questo aspetto.

- I due numeri presentano degli esponenti diversi tra loro, occorre prendere il valore avente quello più piccollo (x) e trasformarlo in modo che il suo esponente diventi 1. Il valore finale di x dopo questo passaggio sarà 0.029×10^1.

- Possiamo ora sommare le mantisse tra loro lasciando invariati i due esponenti.

$$\begin{array}{r} 0.029 \times 10^1 \\ +\ 0.25 \times 10^1 \\ \hline 0.279 \times 10^1 \end{array}$$

- Il risultato che abbiamo ottenuto nell'ultima operazione non è conforme all'insieme F definito all'inizio dell'esempio. Dobbiamo quindi approssimare il risultato che abbiamo ottenuto effettuando il floating per arrotondamento

$$fl_A(x + y) = fl_A(0.279 \times 10^1) = 0.28 \times 10^1$$

Prodotto algebrico:

- Anche in questo caso occorre trasformare il valore (se non lo è già) in forma normalizzata.

- Si moltiplicano o dividono le mantisse e successivamente si sommano o sottraggono i vari esponenti.

- Se il risultato non è un valore che rientra all'interno dell'insieme F allora occorre, anche in questo caso effettuare il floating del risultato.

Esempio 2.6.2. L'insieme F e i valori x e y sul quale operiamo sono gli stessi dell'esempio precedente, questa volta l'operazione che vogliamo effettuare è il prodotto.

- Come prima, i valori sono già in forma normalizzata.

- Effettuiamo il prodotto tra le due mantisse.

$$\begin{array}{r} 0.29 \times 10^0 \\ \times\ 0.25 \times 10^1 \\ \hline 0.0725 \times 10^1 \end{array}$$

- Effettuiamo ora l'arrotondamento:

$$fl_A(0.0725 \times 10^1) = 0.072 \times 10^1 \; per \; R.T.E$$

Esempio 2.6.3. Immaginiamoci ora di voler eseguire delle operazioni nell'insieme $F(2, 3, -3, 2)$.
Vogliamo sommare i valori $x = 1$ e $y = 2^{-2}$. In base $\beta = 2$ avremmo che $x = 0.100 \times 2^1$ poichè il primo 1 dopo la virgola decimale moltiplica il valore β^{-1} e, invece $y = 0.100 \times 2^{-1}$, per calcoli analoghi.

$$\begin{aligned} x \oplus y &= fl_A(0.100 \times 2^1 + 0.100 \times 2^{-1}) \\ &= fl_A(0.100 \times 2^1 + 0.001 \times 2^1) \\ &= 0.101 \times 2^1 > 1 \end{aligned} \tag{2.1}$$

Effettuiamo lo stesso calcolo ma questa volta con valore $y = 2^{-3}$

$$\begin{aligned} x \oplus y &= fl_A(0.100 \times 2^1 + 0.100 \times 2^{-2}) \\ &= fl_A(0.100 \times 2^1 + 0.0001 \times 2^1) \\ &= 0.100 \times 2^1 = 1 \end{aligned} \tag{2.2}$$

Soffermiamoci a ragionare sui calcoli che sono stati svolti in questo esempio. Visto che stiamo lavorando nell'insieme $F(2, 3, -3, 2)$ vogliamo calcolare di questo il valore di *eps* il quale vale $\beta^{1-t} = 2^{1-3} = 2^{-2}$ ed il valore della roundoff unit $u = \frac{1}{2}\beta^{1-t} = 2^{-3}$.
Questi valori che abbiamo ottenuto sono esattamente gli stessi delle y che abbiamo usato nei casi precedenti; nel primo conto (somma di 1 con *eps*) siamo riusciti ad ottenere il successivo numero di macchina di 1 mentre nel secondo l'arrotondamento ci ha restituito il numero x di partenza. Proprio l'ultimo calcolo ci fa capire che lo 0 non è più l'elemento neutro rispetto la somma, infatti basterebbe sommare a x qualsiasi numero $y \leq u$ per ottenere il valore iniziale di x.

2.6.3 Stima degli errori relativi di moltiplicazione e divisione

In questa sezione vogliamo stimare gli errori relativi che vengono effettuati nel calcolare un operazione di moltiplicazione o divisione in aritmetica macchina.
Dati due valori $x, y \in \mathbb{R}$ tali che $x, y \notin F$ vogliamo effettuare l'operazione $x \otimes y$.
La prima cosa da fare è quella di ottenere dei valori rappresentativi di x e y all'interno di F, effettuiamo un arrotondamento :

$$x \longrightarrow fl_A(x) = x(1+\varepsilon_x), \quad |\varepsilon_x| \leq u$$

$$y \longrightarrow fl_A(y) = y(1+\varepsilon_y), \quad |\varepsilon_y| \leq u$$

Il calcolo da effettuare all'interno della macchina è :

$$fl_A(x) \otimes fl_A(y) = fl_A(fl_A(x) fl_A(y)) = (x(1+\varepsilon_x)y(1+\varepsilon_y))(1+\varepsilon_p) = xy(1+\varepsilon_x)(1+\varepsilon_y)(1+\varepsilon_p).$$

L'errore relativo sarà :

$$\begin{aligned} E_{rel} &= \left| \frac{xy - xy(1+\varepsilon_x)(1+\varepsilon_y)(1+\varepsilon_p)}{xy} \right| \\ &= |1 - (1+\varepsilon_x)(1+\varepsilon_y)(1+\varepsilon_p)| \\ &\approx |1 - (1+\varepsilon_x+\varepsilon_y+\varepsilon_p)| \\ &= |\varepsilon_x + \varepsilon_y + \varepsilon_p| \\ &< |3u| \end{aligned} \tag{2.3}$$

Sviluppando $|(1+\varepsilon_x)(1+\varepsilon_y)(1+\varepsilon_p)|$ ottengo dei valori del tipo $\varepsilon\varepsilon$ o $\varepsilon\varepsilon\varepsilon$ trascurabili. Da questo risultato otteniamo che l'operazione del prodotto è un operazione sicura o *stabile* poichè indifferentemente dai valori x e y di partenza l'errore relativo massimo che possiamo ottenere, quando moltiplichiamo. è minore di tre volte la roundoff unit.
Cosa succede se, invece che moltiplicare, io dividessi i valori di x e y di partenza?
I passaggi iniziali rimangono invariati, ovvero, se avessi che $x, y \notin F$ allora dovrei operare delle approssimazioni tali per cui possa rappresentare sia x che y all'interno di F.
Calcolo il valore della divisione internamente alla macchina:

$$x \oslash y = fl_A\left(\frac{fl_A(x)}{fl_A(y)}\right) = \left(\frac{x(1+\varepsilon_x)}{y(1+\varepsilon_y)}\right)(1+\varepsilon_d)$$

L'errore relativo è in questo caso :

$$\begin{aligned} E_{rel} &= \left| \frac{\frac{x}{y} - \left(\frac{x(1+\varepsilon_x)}{y(1+\varepsilon_y)}\right)(1+\varepsilon_d)}{\frac{x}{y}} \right| \\ &= \left| 1 - \frac{(1+\varepsilon_x)}{(1+\varepsilon_y)}(1+\varepsilon_d) \right| \end{aligned} \tag{2.4}$$

A questo punto utilizziamo l'espansione di serie di Taylor di una funzione $f(x)$ in un intorno di x_0 arrestata al primo ordine:

$$f(x) \approx f(x_0) f^{'}(x_0)(x - x_0)$$

dove nel nostro caso $x = \varepsilon_y$, $x_0 = 0$:

$$f(\varepsilon_y) \approx f(0) f^{'}(0)\varepsilon_y$$

La funzione $f(\varepsilon) = \frac{1}{1+\varepsilon_y}$ mentre $f^{'}(\varepsilon) = \frac{1}{(1+\varepsilon_y)^2}$ sostituendo avremmo che :

$$f(\varepsilon_y) \approx 1 - \varepsilon_y$$

$$\begin{aligned} E_{rel} &= |1 - (1+\varepsilon_x)(1-\varepsilon_y)(1+\varepsilon_d)| \\ &\approx |\varepsilon_x - \varepsilon_y + \varepsilon_d| \\ &< |3u| \end{aligned} \tag{2.5}$$

Anche in questo caso abbiamo ottenuto un'ulteriore importante risultato, ovvero, che esattamente come il prodotto anche la divisione risulta essere un'operazione stabile.

2.6.4 Stima degli errori relativi della somma

Esattamente per come abbiamo fatto per divisione e moltiplicazione vogliamo vedere qual'è l'errore relativo che possiamo commettere se effettuiamo una somma algebrica tra due numeri $x, y \notin F$. Si omettono di seguito le varie operazioni di floating per i valori di x e di y, passiamo direttamente al calcolo :

$$x \oplus y = fl_A(fl_A(x) + fl_A(y)) = (x(1+\varepsilon_x)y(1+\varepsilon_y))(1+\varepsilon_s)$$

L'errore relativo sarà :

$$
\begin{aligned}
E_{rel} &= \left| \frac{(x+y) - [x(1+\varepsilon_x) + y(1+\varepsilon_y)](1+\varepsilon_s)}{x+y} \right| \\
&\approx \left| \frac{x\varepsilon_x + y\varepsilon_y + (x+y)\varepsilon_s}{x+y} \right| \\
&\leq \left| \frac{x}{x+y} \right| \varepsilon_x + \left| \frac{y}{x+y} \right| \varepsilon_y + \varepsilon_s \\
&\leq \left| \frac{x}{x+y} \right| u + \left| \frac{y}{x+y} \right| u + u \\
&= \left(\left| \frac{x}{x+y} \right| + \left| \frac{y}{x+y} \right| + 1 \right) u
\end{aligned}
\tag{2.6}
$$

Rispetto al caso della moltiplicazione e divisione qui abbiamo che l'errore relativo dipende dai valori di x e y. Infatti mentre nel caso precedente il fattore che moltiplicava u era una costante di valore 3, qui abbiamo una funzione dipendente unicamente dalle variabili x ed y la quale potrebbe dare un valore grande quanto più il valore del denominatore $(x+y)$ è piccolo.
Se x e y hanno segno discorde e valori assoluti molto prossimi allora il valore dei denominatori tende a 0 e il valore che moltiplica la u tende a diventare molto elevato. Questo fenomeno viene detto **fenomeno della cancellazione algebrica**.

2.7 Osservazioni conclusive

Abbiamo visto in questo capitolo come l'effettuare operazioni all'interno della macchina comporti una serie di approssimazioni che coinvolgono sia gli operandi che i risultati delle operazioni stesse.
Con un esempio vediamo come operazioni algebricamente identiche possano produrre talvolta risultati inaccettabili all'interno di un insieme F.

Esempio 2.7.1. Sia dato un insieme $F(10, 2, L, U)$ e due numeri $a, b \in F$ vogliamo calcolarne la loro media adoperando due algoritmi distinti che algebricamente producono lo stesso risultato

- $\frac{a+b}{2}$
- $a + \frac{b-a}{2}$

Algoritmo 1: Sia $a = 0.96 \times 10^{-1}$ e $b = 0.99 \times 10^{-1}$ voglio prima sommare il loro valore e, successivamente, dividere il tutto per 2.

$$\begin{array}{r} 0.96 \times 10^{-1} \\ +\ 0.99 \times 10^{-1} \\ \hline 1.95 \times 10^{-1} \end{array}$$

che in forma normalizzata è 0.195^0. Ora vogliamo assicurarci che il risultato ottenuto sia sempre interno all'interno di F per questo motivo effettuiamo un arrotondamento di quanto ottenuto :

$$fl_A(a+b) = fl_A(0.195^0) = 0.2 \times 10^0 \quad per\ R.T.E$$

Infine calcoliamo l'approssimazione della somma ottenuta moltiplicata per 0.5 (ovvero divisa per due).

$$fl_A((a+b) \times 0.5) = 0.1 \times 10^0$$

Cosa notiamo da questo risultato ?
Se visualizzassimo l'intervallo $[a, b]$ noteremo sicuramente che il risultato ottenuto da questo primo algoritmo non vi appartiene.

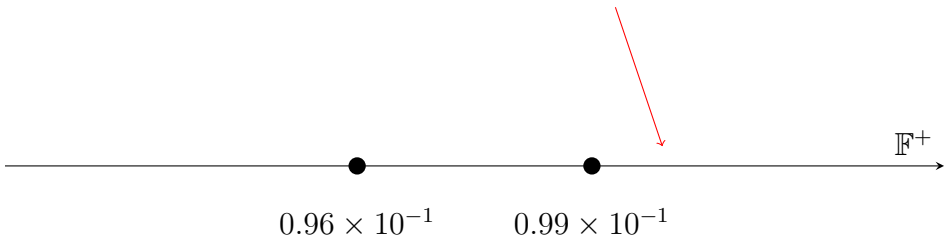


Figura 2.5: Visualizzazione del primo algoritmo

Algoritmo 2: Calcoliamo anzitutto $b - a = 0.03 \times 10^{-1} = 0.3 \times 10^{-2}$, il risultato ottenuto da questa operazione rimane ancora all'interno di F, non abbiamo bisogno di ulteriori arrotondamenti.
Procediamo poi a dividere il risultato per 2 ed effettuare successivamente un floating :

$$fl_A(0.3 \times 10^{-2}) = 0.15 \times 10^{-2}$$

A questo punto sommiamo al risultato ottenuto il valore di a.

$$fl_A(0.96 \times 10^{-1} + 0.15 \times 10^{-2}) = fl_A(0.975 \times 10^{-1}) = 0.98 \times 10^{-1} \quad per\ R.T.E$$

Possiamo notare subito il perchè, rispetto al primo, questo algoritmo ci dia un risultato più soddisfacente, infatti il risultato che abbiamo ottenuto è effettivamente un valore compreso tra i due estremi a e b che si ritrova in una posizione abbastanza centrale tra i due.

Capitolo 3

Norme

3.1 Norme vettoriali

Definizione 3.1.1. Una norma vettoriale è una funzione che associa ad un vettore $\underline{x}$ di $\mathbb{R}^n$ un numero reale maggiore o uguale di 0:

$$\|\underline{x}\| : \mathbb{R}^n \longrightarrow \mathbb{R}_+ \cup \{0\}$$

Una norma vettoriale deve godere delle seguenti proprietà:

- $\|\underline{x}\| \geq 0 \ \forall \underline{x} \in \mathbb{R}^n$.
- $\|\underline{x}\| = 0$ solamente se $\underline{x} = 0$, ovvero il vettore è nullo.
- $\|\alpha\underline{x}\| = |\alpha|\|\underline{x}\|$ con $\alpha \in \mathbb{R}$
- $\|\underline{x} + \underline{y}\| \leq \|\underline{x}\| + \|\underline{y}\|$ per la disuguaglianza triangolare.

Le norme che utilizzeremo da quì in avanti godono di tutte queste proprietà (dimostrazione non fatta in classe).

3.1.1 Norma infinito

La norma infinito di un vettore è definita nella seguente maniera :

$$\|\underline{x}\|_\infty = \max_{\{i=1,\dots,n\}} |x_i|$$

Esempio 3.1.1. Sia dato il vettore :

$$\underline{x} = \begin{pmatrix} 5 \\ 2 \\ -8 \end{pmatrix}$$

la sua norma infinito sarà :

$$\|\underline{x}\|_\infty = 8$$

come da definizione.

3.1.2 Norma uno

Per calcolare la norma uno di un vettore basta semplicemente fare la somma dei valori assoluti dei suoi componenti :

$$\|\underline{x}\|_1 = \sum_{i=1}^{n} |x_i|$$

Esempio 3.1.2. Il risultato della norma 1 della matrice usata nell'esempio precedente è uguale a :

$$\|\underline{x}\|_1 = \sum_{i=1}^{n} |x_i| = |5| + |2| + |8| = |15|$$

3.1.3 Norma due (o norma euclidea)

Esistono due scritture equivalenti per esprimere questa tipologia di norma. La prima è la seguente :

$$\|\underline{x}\|_2 = \sqrt{\underline{x}^T \underline{x}}$$

dove $\underline{x}^T$ sta a significare la trasposizione del vettore x. Un vettore solitamente è rappresentato come **vettore colonna**, ovvero, i suoi valori nella rappresentazione matriciale sono disposti verticalmente su una sola colonna. L'operazione di trasposizione di un vettore significa che il dato vettore verrà rappresentato su una riga al posto di una colonna.

Se moltiplicassimo un vettore per il suo trasposto avremmo il seguente risultato :

$$\underline{x}^T \underline{x} = \begin{pmatrix} x_1 \\ x_2 \\ . \\ x_n \end{pmatrix} \begin{pmatrix} x_1 \; x_2 \; . \; x_n \end{pmatrix} = (x_1 \times x_1) + (x_2 \times x_2) + ... + (x_n \times x_n) = \sum_{i=1}^{n} x_i^2$$

Quindi possiamo scrivere la norma due di un vettore nel seguente modo :

$$\|\underline{x}\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

Esempio 3.1.3. Il risultato della norma 2 della matrice usata nell'esempio precedente è uguale a :

$$\|\underline{x}\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2} = \sqrt{25 + 4 + 64} = \sqrt{93} = 9.64$$

Vogliamo ora fare una notazione importante :
Immaginiamo di avere un vettore $\underline{y} \in \mathbb{R}^n$ tale per cui $\underline{y} = A\underline{x}$ dove A è un vettore ortogonale, ovvero, $A^T A = AA^T = I$. Calcoliamo la norma due del

vettore $\underline{y}$:

$$\|\underline{y}\|_2 = \sqrt{\underline{y}^T \underline{y}} = \sqrt{(A\underline{x})^T A\underline{x}} = \sqrt{A^T \underline{x}^T A\underline{x}} = \sqrt{\underline{x}^T \underline{x}} = \|\underline{x}\|_2$$

3.2 Errore relativo vettoriale

Si vuole estendere ora il concetto di errore relativo definito nel capitolo precedente applicandolo al caso in cui si vuole lavorare con dati in input che non sono più singolo valori ma bensì devi vettori. Si immagini di avere come dati di input un vettore $\underline{x}$ ed una funzione $f(\underline{x})$ la quale produce un risultato. Per rappresentare il nostro vettore all'interno della nostra macchina occorre effettuare delle approssimazioni ed ottenere quello che è un dato perturbato. Sia quindi $\widetilde{\underline{x}}$ il nostro dato perturbato e $\underline{x}$ il vettore di origine. Possiamo calcolare il nostro errore relativo vettoriale nella seguente maniera :

$$E_{rel} = \frac{\|\underline{x} - \widetilde{\underline{x}}\|}{\|\underline{x}\|}$$

Ovviamente la cosa da notare è che le norme utilizzate per il denominatore e il numeratore devono coincidere come tipo. Verranno adoperate solamente norme che abbiamo visto in precedenza (1,2, ∞).

3.2.1 Il teorema di equivalenza

Trattiamo ora la relazione che esiste tra le varie norme quando si va a calcolare un errore relativo, per fare ciò vogliamo mettere a confronto tra

loro tutte le tipologie di norme prese una alla volta. Siano dati due valori $A, B \in \mathbb{R}$ con $0 < A \leq B$ e siano $\|\cdot\|_+$ e $\|\cdot\|_*$ due norme aventi tipo 1,2 o ∞ tali che valga:

$$A\|\underline{x}\|_+ \leq \|\underline{x}\|_* \leq B\|\underline{x}\|_+ \tag{3.1}$$

Sottraiamo ad ogni elemento della nostra catena di disequazioni il vettore $\widetilde{\underline{x}}$:

$$A\|\widetilde{\underline{x}} - \underline{x}\|_+ \leq \|\widetilde{\underline{x}} - \underline{x}\|_* \leq B\|\widetilde{\underline{x}} - \underline{x}\|_+ \tag{3.2}$$

Possiamo subito notare che $\|\widetilde{\underline{x}} - \underline{x}\|_+$ è il numeratore di un errore relativo. Se prendessimo sempre l'equazione 3.1 e calcolassimo i reciproci avremmo :

$$\frac{1}{B}\frac{1}{\|\underline{x}\|_+} \leq \frac{1}{\|\underline{x}\|_*} \leq \frac{1}{A}\frac{1}{\|\underline{x}\|_+} \tag{3.3}$$

Che rappresenta il denominatore di un errore relativo. Dividendo tra loro 3.2 e 3.3 avremmo il seguente risultato:

$$\frac{A}{B}\frac{\|\widetilde{\underline{x}} - \underline{x}\|_+}{\|\underline{x}\|_+} \leq \frac{\|\widetilde{\underline{x}} - \underline{x}\|_*}{\|\underline{x}\|_*} \leq \frac{B}{A}\frac{\|\widetilde{\underline{x}} - \underline{x}\|_+}{\|\underline{x}\|_+} \tag{3.4}$$

La 3.5 ci fa vedere proprio come i vari errori relativi calcolati nelle norme $\|\cdot\|_+$ e $\|\cdot\|_*$ siano messi in relazione tra loro.
Questo risultato che abbiamo ottenuto è molto importante. Se avessimo infatti calcolato il risultato di un errore relativo utilizzando una determinata norma possiamo ottenere una stima dello stesso calcolato adoperando una norma differente senza svolgere dei nuovi conti, il che è molto conveniente se stiamo adoperando dei vettori aventi delle dimensioni elevate.
Per le norme che abbiamo visto valgono le seguenti proposizioni:

- se $* = 2$ e $+ = \infty$ allora $A = 1$ e $B = \sqrt{n}$ dove n è la dimensione del vettore.
- se $* = 1$ e $+ = \infty$ allora $A = 1$ e $B = n$ dove n è la dimensione del vettore.
- se $* = 1$ e $+ = 2$ allora $A = 1$ e $B = \sqrt{n}$ dove n è la dimensione del vettore.

Il rapporto tra i due errori relativi è :

$$\frac{A}{B} \leq \frac{E^*_{rel}}{E^+_{rel}} \leq \frac{B}{A} \tag{3.5}$$

3.3 Norme matriciali

Definizione 3.3.1. Una norma matriciale **generalizzata** è una funzione che, analogamente a quella definita per le norme vettoriali, associa ad una matrice $\underline{A}$ un numero reale maggiore o uguale a 0 :

$$\|\underline{A}\| : \mathbb{R}^{m\times n} \longrightarrow \mathbb{R}_+ \cup \{0\}$$

Una norma matriciale generalizzata gode delle seguenti proprietà (anche queste analoghe alla norma vettoriale):

- $\|\underline{A}\| \geq 0 \ \forall \underline{A} \in \mathbb{R}^{m\times n}$.
- $\|\underline{A}\| = 0$ solamente se $\underline{A} = 0$, ovvero la matrice è vuota
- $\|\alpha\underline{A}\| = |\alpha|\|\underline{A}\|$ con $\alpha \in \mathbb{R}$
- $\|\underline{A} + \underline{B}\| \leq \|\underline{A}\| + \|\underline{B}\|$ per la disuguaglianza triangolare.

Definizione 3.3.2. Si definisce una norma matriciale una funzione che oltre ad essere una norma matriciale generalizzata rispetta anche la proprietà submoltiplicativa (detta anche di consistenza):

$$\|\underline{AB}\| \leq \|\underline{A}\|\|\underline{B}\|$$

Non tutte le norme matriciali generalizzate quindi possono essere norme matriciali.

Esempio 3.3.1. Si immagini di aver definito una norma matriciale nel seguente modo :

$$\|\underline{A}\| = \max_{\{i,j\}} |a_{ij}|$$

Questa è una norma matriciale generalizzata che però non risulta essere una norma matriciale poichè date due matrici definite nel seguente modo :

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

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

di cui :

$$\underline{AB} = \begin{pmatrix} 2 & 1 \\ 1 & 0 \end{pmatrix}$$

Avremo che $\|\underline{AB}\| = 2$ e invece $\|\underline{A}\| = \|\underline{B}\| = 1$

Definizione 3.3.3. Una norma matriciale $\|\cdot\|_M$ si dice compatibile con una norma vettoriale $\|\cdot\|_V$ se la seguente relazione è valida:

$$\|\underline{A}\underline{x}\|_V \leq \|\underline{A}\|_M \|\underline{x}\|_V$$

Dove $\underline{A}$ e $\underline{x}$ sono rispettivamente una matrice ed un vettore che possono essere moltiplicati tra loro.

Definizione 3.3.4. Una norma matriciale $\|\cdot\|_M$ si dice essere naturale o indotta da una norma vettoriale $\|\cdot\|_V$ solamente se produce la più piccola costante C tale per cui vale la relazione :

$$\|\underline{A}\underline{x}\|_V \leq C\|\underline{x}\|_V$$

3.3.1 Norma matriciale infinita e uno

Definiamo una matrice $\underline{A}$:

$$\underline{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & \ddots & \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} \in \mathbb{R}^{m \times n}$$

Definizione 3.3.5. Si definisce norma matriciale infinito la funzione :

$$\|\underline{A}\|_\infty = \max_{\{i=1,\dots m\}} \sum_{j=1}^{n} |a_{i,j}|$$

prendendo come riferimento $\underline{A}$ siano :

- $C_1 := |a_{11}| + |a_{12}| + \dots + |a_{1n}|$
- $C_2 := |a_{21}| + |a_{22}| + \dots + |a_{2n}|$
- ...
- $C_m := |a_{m1}| + |a_{m2}| + \dots + |a_{mn}|$

Allora possiamo scrivere la funzione precedente come :

$$\|\underline{A}\|_\infty = \max\{C_1, C_2, \dots, C_m\}$$

Definizione 3.3.6. Si definisce norma matriciale uno la funzione :

$$\|\underline{A}\|_1 = \max_{\{j=1,\dots n\}} \sum_{i=1}^{m} |a_{i,j}|$$

Consideriamo ora :

- $D_1 := |a_{11}| + |a_{21}| + \cdots + |a_{m1}|$
- $D_2 := |a_{12}| + |a_{22}| + \cdots + |a_{m2}|$
- ...
- $D_n := |a_{1n}| + |a_{2n}| + \cdots + |a_{mn}|$

possiamo riscrivere la funzione precedente come :

$$\|\underline{A}\|_1 = \max\{D_1, D_2, \dots, D_m\}$$

3.3.2 Norma matriciale due

Definizione 3.3.7. Si definisce norma matriciale due la funzione :

$$\|\underline{A}\|_2 = \sqrt{\rho(\underline{A}^T \underline{A})}$$

dove $\rho(X)$ indica il raggio spettrale di una matrice X. Il raggio spettrale rappresenta l'autovalore di valore assoluto massimo.
La matrice $\underline{M} = \underline{A}^T \underline{A}$ di cui andiamo a calcolare il nostro raggio spettrale gode di due proprietà :

1. E' simmetrica : $\underline{M} = \underline{M}^T$. E' facile dimostrare tale proprietà infatti : $\underline{M}^T = (\underline{A}^T \underline{A})^T = \underline{A}^T \underline{A} = \underline{M}$.
2. E' semidefinita positiva : $\forall \underline{x} \in \mathbb{R}^n \setminus \{0\}$ si ha che $\underline{x}^T \underline{M} \underline{x} \geq 0$.

3.4 Numero di condizionamento in norma due

Prendiamo in considerazione la matrice definita precedentemente $\underline{A}^T \underline{A}$, come sappiamo questa è semidefinita positiva e simmetrica, inoltre possiede n autovalori reali e non nulli. Possiamo ordinare per valore crescente quest'ultimi e in particolare interessarci dei due aventi valore massimo e minimo:

$$\lambda_{max}(\underline{A}^T \underline{A}) \quad \lambda_{min}(\underline{A}^T \underline{A})$$

Definizione 3.4.1. Si definisce numero di condizionamento in norma 2 di una matrice $\underline{A}$ il valore :

$$K_2(\underline{A}) = \frac{\sqrt{\lambda_{max}(\underline{A}^T\underline{A})}}{\sqrt{\lambda_{min}(\underline{A}^T\underline{A})}}$$

ovvero, il rapporto delle radici dell'autovalore massimo e minimo della matrice $\underline{A}^T\underline{A}$

Possiamo denotare tre proprietà per il numero di condizionamento :

1. $K_2(\underline{A}^T\underline{A}) = K_2(\underline{A})^2 = \frac{\lambda_{max}(\underline{A}^T\underline{A})}{\lambda_{min}(\underline{A}^T\underline{A})}$
2. Se $\underline{A} \in \mathbb{R}^{n\times n}$ è una matrice non singolare (determinante diverso da zero) allora possiamo riscrivere il suo valore di condizionamento nel seguente modo :
$$K_2(\underline{A}) = \|\underline{A}\|_2\|\underline{A^{-1}}\|_2$$
3. Se $\underline{A} \in \mathbb{R}^{n\times n}$ è una matrice simmetrica allora possiamo riscrivere il suo valore di condizionamento nel seguente modo :
$$K_2(\underline{A}) = \frac{\max_{\{i=1,\dots,n\}} |\lambda_i(A)|}{\min_{\{i=1,\dots,n\}} |\lambda_i(A)|}$$

Capitolo 4

Condizionamento di un problema e stabilità di un algoritmo

Iniziamo questo capitolo introducendo nuovamente alcuni elementi che avevamo visto già precedentemente. Quando lavoriamo all'interno della macchina vogliamo svolgere alcune operazioni e ottenere dei risultati da queste, per far ciò occorre essere in possesso di due informazioni :

- un dato esatto x su cui operare.
- una funzione f che mappa il dato di partenza in un risultato che chiamiamo $f(x)$

Ovviamente, come già visto oramai in diversi esempi, occorrerà memorizzare x all'interno della macchina adoperando alcune approssimazioni, ottenendo in questo caso un valore perturbato $\tilde{x}$. Anche la funzione f ovviamente mapperà questo valore perturbato in un risultato $f(\tilde{x})$ anch'esso perturbato. Definiamo quindi:

- $f(x)$ risposta ottenuta dall'applicazione risolvente partendo da un dato x esatto.
- $f(\tilde{x})$ risposta ottenuta dall'applicazione risolvente partendo da un dato $\tilde{x}$ che possiede perturbazioni dovute ad approssimazioni.
- $\Psi(\tilde{x})$ risposta ottenuta dall'applicazione risolvente applicando l'algoritmo Ψ, implementato sulla macchina, ed adoperando una dato $\tilde{x}$ avente perturbazioni al suo interno.

Nota bene: è da sottolineare ciò che contraddistingue $f(\tilde{x})$ e $\Psi(\tilde{x})$. Nel primo caso ho il risultato di una funzione f la quale parte da un dato che presenta degli errori, nel secondo caso Ψ rappresenta un algoritmo scritto all'interno della macchina di cui ad ogni passo introduce (possibilmente) ulteriori errori di rappresentazione. Quest'ultimo opera su dati che sono stati soggetti a perturbazioni ($\tilde{x}$).

4.1 Errore inerente, algoritmico e totale

Definizione 4.1.1. Si definisce errore inerente una tipologia di errore che mette a confronto il risultato dell'applicazione della funzione f su un dato esatto e su un dato perturbato:

$$E_{in} = \frac{f(\tilde{x}) - f(x)}{f(x)}$$

L'errore inerente **non è influenzato dall'algoritmo scelto** e quindi dalla serie di operazioni che svolgiamo per produrre il risultato ma solamente dal dato in ingresso.

Definizione 4.1.2. Si definisce errore algoritmico una tipologia di errore che mette a confronto i risultati dell'applicazione dell'algoritmo $\Psi(\tilde{x})$ con il risultato della funzione $f(\tilde{x})$.

$$E_{alg} = \frac{\Psi(\tilde{x}) - f(\tilde{x})}{f(\tilde{x})}$$

L'errore algoritmico, dunque assume di star lavorando già con dei dati appartenenti all'insieme F della macchina e valuta l'errore che si è commesso adoperando una determinata tipologia di algoritmo.

Definizione 4.1.3. Si definisce errore totale una tipologia di errore che tiene conto sia di errori che avvengono su dati sia di errori che avvengono a livello di algoritmo.

$$E_{tot} = \frac{\Psi(\tilde{x}) - f(x)}{f(x)}$$

Possiamo mettere in relazione tra loro queste diverse tipologie di errore; in particolar modo, partendo dall'errore totale, possiamo definire la seguente relazione :

$$E_{tot} = \frac{\Psi(\tilde{x}) - f(x)}{f(x)} = \frac{\Psi(\tilde{x})}{f(x)} - 1 = \frac{\Psi(\tilde{x})}{f(\tilde{x})}\frac{f(\tilde{x})}{f(x)} - 1 = (1 + E_{alg})(1 + E_{in}) - 1$$

Da cui otteniamo:

$$E_{tot} \approx E_{in} + E_{alg}$$

Che conferma ciò che abbiamo detto, ovvero, che l'errore totale è influenzato sia dall'errore commesso nel rappresentare dei dati all'interno dell'insieme dei numeri macchina sia dall'errore generato nell'esecuzione di un determinato algoritmo all'interno della macchina.

L'errore inerente è strettamente legato con il condizionamento; quest'ultimo è una caratteristica del problema e non dipende in alcun modo da come questo viene calcolato.
In altre parole il condizionamento del problema non dipende in nessun modo da errori di approssimazione delle operazioni di macchina o dalla tipologia di algoritmo utilizzata. E' l'errore algoritmico che dipende strettamente dalla tipologia di algoritmo risolutivo che viene adoperato, in particolare questo è influenzato da:

- Numero di operazioni eseguite.
- Ordine di operazioni.
- Tipologia di operazioni.

Le osservazioni fatte fin ora forniscono un importante risultato:
E' inutile tentare di risolvere un problema che **presenta un errore inerente eccessivamente elevato.**

4.2 Studio del condizionamento

Sia x un dato esatto e $\tilde{x}$ il corrispettivo perturbato, ovvero:

$$\tilde{x} = fl(x) = x(1 + \epsilon_x) = x + \delta x \quad t.c\ \delta x \approx 0$$

Utilizziamo Taylor e calcoliamone la serie arrestata al primo ordine della funzione $f(\tilde{x})$

$$f(\tilde{x}) \approx f(x) + f^{'}(x)(\tilde{x} - x)$$

dove $(\tilde{x} - x)$ non è altro che δx per quanto scritto sopra.

Riscriviamo l'equazione e applichiamo le norme ad entrambi i membri :

$$f(\tilde{x})-f(x) \approx f^{'}(x)(\tilde{x}-x) \rightarrow \|f(\tilde{x})-f(x)\| \approx \|f^{'}(x)(\tilde{x}-x)\| \leq \|f^{'}(x)\|\|(\tilde{x}-x)\|$$

supponendo che $\|f(x)\| \neq 0$ divido entrambi i membri dell'equazione per $\|f(x)\|$:

$$\frac{\|f(\tilde{x})-f(x)\|}{\|f(x)\|} \leq \frac{\|f^{'}(x)\|}{\|f(x)\|}\|(\tilde{x}-x)\|$$

Fatto ciò posso notare che l'espressione a sinistra rappresenta un errore inerente. Ponendo $\|x\| \neq 0$ posso veder che relazione ho con l'errore relativo sui dati del problema:

$$\frac{\|f(\tilde{x})-f(x)\|}{\|f(x)\|} \leq \frac{\|f^{'}(x)\|}{\|f(x)\|}\|x\|\frac{\|(\tilde{x}-x)\|}{\|x\|}$$

L'espressione $\frac{\|f^{'}(x)\|}{\|f(x)\|}\|x\|$ rappresenta il numero di condzionamento del problema K il quale può assumere valori ≥ 1 .
Se K assume valori vicini ad 1 allora il valore inerente tenderà ad assumere simili all'errore relativo mentre per valori K grandi avremmo un amplificazione dell'errore relativo sui dati.

Esempio 4.2.1. Per $f : \mathbb{R} \rightarrow \mathbb{R}$ dove $f(x) : x - 1$ l'indice di condizionamento per tale applicazione risolvente sarà:

$$K = \frac{|f^{'}(x)||x|}{|f(x)|} = \frac{|1-0||x|}{|x-1|} = \frac{|x|}{|x-1|}$$

Quindi per valori di x tendenti ad 1 il valore di K tende a diventare estremamente grande ed il problema è mal condizionato.

Esempio 4.2.2. Per $f : \mathbb{R} \rightarrow \mathbb{R}$ dove $f(x) : e^x$ l'indice di condizionamento per tale applicazione risolvente sarà:

$$K = \frac{|f^{'}(x)||x|}{|f(x)|} = \frac{|e^x||x|}{|e^x|} = |x|$$

Quindi il problema è mal condizionato per valori assoluti di x molto più grandi di 1.

Prendiamo ora come riferimento le applicazioni risolventi del tipo $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$.

Definizione 4.2.1. Una funzione $f : \mathbb{R}^n \to \mathbb{R}^n$ è una particolare tipologia di funzione che permette di mappare un vettore in un altro di dimensioni uguali a quello di partenza :

$$\underline{x} = \begin{pmatrix} x_1 \\ x_2 \\ . \\ x_n \end{pmatrix} \mapsto f(\underline{x}) = \begin{pmatrix} f_1(\underline{x}) \\ f_2(\underline{x}) \\ . \\ f_n(\underline{x}) \end{pmatrix}$$

Dove $f_1, f_2, \ldots, f_n$ sono funzioni del tipo $f_i : \mathbb{R}^n \to \mathbb{R}$

Consideriamo ora di avere una matrice $\underline{A}$, e due vettori $\underline{x}$ e $\underline{b}$ e si voglia risolvere il sistema lineare :

$$\underline{Ax} = \underline{b}$$

Dove

- $\underline{A}$ è la matrice dei coefficienti.
- $\underline{x}$ è il vettore delle incognite (e rappresenta il risultato che vogliamo trovare).
- $\underline{b}$ è il vettore dei termini noti.

Sia ora :

$$\underline{A\tilde{x}} = \underline{\tilde{b}}$$

sempre il nostro sistema lineare nel quale al quale è stato indotto un errore di approssimazione a livello dei dati. Come nel caso precedente anche qui vogliamo studiare l'errore inerente:

$$\frac{\|f_A(\tilde{b}) - f_A(b)\|}{\|f_A(b)\|} \leq K \frac{\|\tilde{b} - b\|}{\|b\|}$$

Ovvero :

$$\frac{\|\tilde{x} - x\|}{\|x\|} \leq K \frac{\|\tilde{b} - b\|}{\|b\|}$$

Definizione 4.2.2. Si definisce matrice Jacobiana una matrice del seguente tipo :

$$\mathbb{J} = \begin{bmatrix} \dfrac{\partial \mathbf{f}(\mathbf{x})}{\partial x_1} & \cdots & \dfrac{\partial \mathbf{f}(\mathbf{x})}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f_1(\mathbf{x})}{\partial x_1} & \cdots & \dfrac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_m(\mathbf{x})}{\partial x_1} & \cdots & \dfrac{\partial f_m(\mathbf{x})}{\partial x_n} \end{bmatrix}$$

Esempio 4.2.3. Sia $f : \mathbb{R}^3 \to \mathbb{R}^3$ una funzione :

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} 2x + 3y - 4z \\ x^2 + y - z \\ x - 2y + 3z \end{pmatrix}$$

La matrice $\mathbb{J}$ di questa sarà :

$$\mathbb{J} = \begin{bmatrix} 2 & 3 & -4 \\ 2x & 1 & -1 \\ 1 & -2 & 3 \end{bmatrix}$$

Sia quindi la funzione $f_A : b \mapsto x = A^{-1}b$ l'applicazione risolvente del sistema lineare $\underline{Ax} = \underline{b}$. Definiamo il numero di condizionamento K :

$$K = \frac{\|f_A^{'}(b)\|\|b\|}{\|f_A(b)\|} = \frac{\|\underline{A}^{-1}\|\|\underline{Ax}\|}{\|\underline{x}\|} \leq \|\underline{A}^{-1}\|\|\underline{A}\|$$

- Per calcolare $f_A^{'}(b)$ occorrerebbe adoperare la matrice Jacobiana definita sopra. Tuttavia notiamo che l'applicazione risolvente è lineare in b. Quest'ultimo termine, quando si effettua la derivata, scompare facendo rimanere solamente la matrice A^{-1}.

- La sostituzione ($\|b\|$) deriva proprio dalla definizione del problema $\underline{Ax} = \underline{b}$

- La norma adoperata è una norma vettoriale e vale quindi la relazione : $\frac{\|\underline{Ax}\|}{\|x\|} \leq \|A\|$. Quest'ultima giustifica la disuguaglianza finale.

In caso di funzioni $f : \mathbb{R}^n \to \mathbb{R}$, ovvero, funzioni che date in input un vettore $\underline{x}$ restituiscono un numero reale, possiamo calcolare il nostro indice di condizionamento nella seguente maniera :

$$K = \sum_{i=1}^{n} \left| \frac{\partial f(\underline{x})}{\partial x_1} \right| \left| \frac{x_i}{f(\underline{x})} \right|$$

Esempio 4.2.4. $f(a, b, c) = a + b + c$

$$K = \frac{|a|}{|a + b + c|} + \frac{|b|}{|a + b + c|} + \frac{|c|}{|a + b + c|}$$

Il problema risulta mal condizionato per valori di $|a + b + c|$ tendenti a 0.

Esempio 4.2.5. $f(a, b) = a^2 - b^2$

$$K = |2a|\frac{|a|}{|a^2 - b^2|} + |-2b|\frac{|b|}{|a^2 - b^2|} = \frac{2a^2}{|a^2 - b^2|} + \frac{2b^2}{|a^2 - b^2|}$$

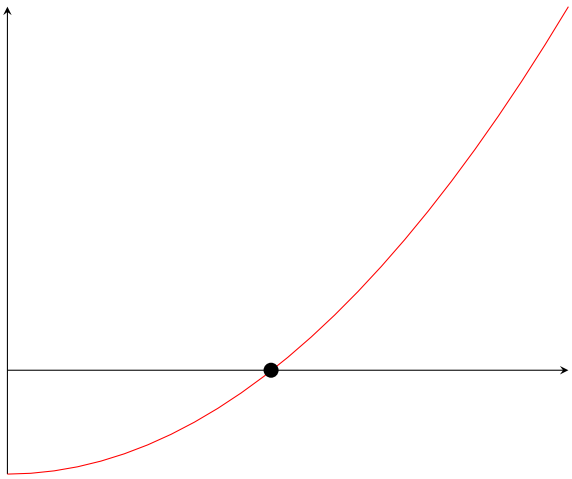
Il problema risulta mal condizionato per valori di $|a^2 - b^2|$ tendenti a 0 .

Capitolo 5

Ricerca degli zeri di funzione

In questo capitolo vengono discussi gli algoritmi per determinare gli zeri o radici di una funzione

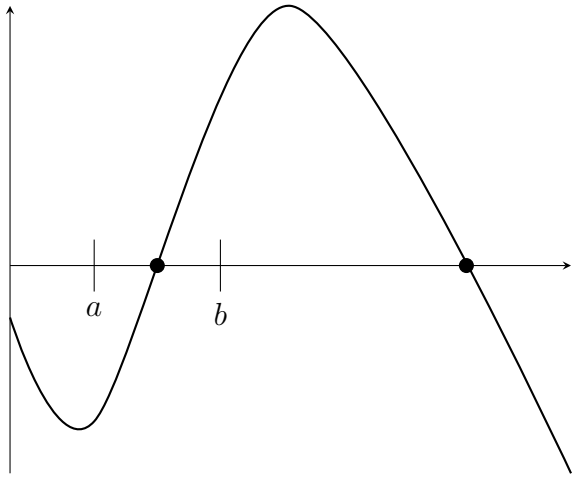
Definizione 5.0.1. Data una funzione $f : \mathbb{R} \mapsto \mathbb{R}$ continua in tutto il suo dominio e differenziabile si dice che il valore α sia uno zero o una radice di questa se $f(\alpha) = 0$



Distinguiamo due categorie distinte di radici :

- α si dice sia una radice semplice della funzione $f(x)$ se $f(\alpha) = 0$ e $f^{'}(\alpha) \neq 0$
- α si dice sia uno radice multipla di molteplicità m della funzione $f(x)$ se $f(\alpha) = 0, f^{'}(\alpha) = 0, f^{''}(\alpha) = 0, \ldots, f^{m-1}(\alpha) = 0$ e $f^{m}(\alpha) \neq 0$.

Per procedere nella risoluzione di un problema di questo tipo occorre che questo sia ben posto, ovvero, vogliamo che nell'intervallo in cui è definita la nostra funzione $f(x)$ in esame sia presente **una sola radice**.
Ovviamente esistono casi in cui la funzione possiede più zeri all'interno del suo dominio ed occorrerà effettuare degli step preliminari in cui individueremo un intervallo specifico di $f(x)$ dove vi è una sola radice (avente una qualsiasi molteplicità). Questa operazione è detta differenziazione degli zeri di una funzione.



Nell'esempio quì sopra considereremo la figura nell'intervallo $[a, b]$, ovvero, $f : [a, b] \mapsto \mathbb{R}$.
Una volta che siamo certi di lavorare su un problema che è ben posto occorrerà controllare che questo sia anche ben condizionato.
Abbiamo già visto in più occasioni che lavorare all'interno ad una macchina per trovare la soluzione ad un problema causa un determinato quantitativo di errore che abbiamo definito precedentemente come errore totale. Sappiamo che composto da due fattori : l'errore inerente e l'errore algoritmico il cui primo subisce variazioni di un determinato fattore K che abbiamo chiamato indice di condizionamento di cui vogliamo, anche in questo caso, andarne ad analizzare il valore.

5.1 Studio del condizionamento

L'approccio che adottiamo nello studio del condizionamento è anche in questo caso quello di simulare una perturbazione sui dati del problema e vedere come si comportano i risultati.

Problema perturbato:
Determinare il valore $\tilde{\alpha} = \alpha + \delta$ tale che $\tilde{f(\tilde{\alpha})} = 0$ dove $\tilde{f} = f + \varepsilon g$,
$g : \mathbb{R} \mapsto \mathbb{R}$ differenziabile dove

- δ è una perturbazione sul risultato del problema.
- ε è una perturbazione sui dati del problema che viene condizionata dalla funzione g.

Procediamo considerando lo sviluppo di Taylor arrestato al prim'ordine sulla funzione $\tilde{f(\tilde{\alpha})}$.

$$\tilde{f(\tilde{\alpha})} = 0 = \tilde{f(\alpha)} + \tilde{f'(\alpha)}(\tilde{\alpha} - \alpha) \tag{5.1}$$

Possiamo riscrivere $\tilde{f(\alpha)}$ come $f(\alpha) + \varepsilon g(\alpha)$ ed essendo $f(\alpha) = 0$ avremmo che $\tilde{f(\alpha)} = \varepsilon g(\alpha)$, inoltre $(\tilde{\alpha} - \alpha)$ rappresenta il valore δ.

$$\tilde{f(\alpha)} = \varepsilon g(\alpha) + \tilde{f'(\alpha)}\delta \tag{5.2}$$

Come da definizione sappiamo che le funzioni $f, \tilde{f}, g$ sono differenziabili.
Tra di loro è possibile stabilire la relazione $f'(x) = \tilde{f}'(x) + \varepsilon g'(x)$.

$$\begin{aligned} \tilde{f(\tilde{\alpha})} &= \varepsilon g(\alpha) + \delta(\tilde{f}'(\alpha) + \varepsilon g'(\alpha)) \\ &= \varepsilon g(\alpha) + \delta \tilde{f}'(\alpha) + \delta\varepsilon g'(\alpha) \\ &\approx \varepsilon g(\alpha) + \delta \tilde{f}'(\alpha) \end{aligned} \tag{5.3}$$

E' da notare in questi ultimi passaggi come il termine $\delta\varepsilon g'(\alpha)$ sia stato eliminato poichè il valore che assumeva era irrilevante (dato che sia δ che ϵ sono entrambi molto piccoli).
Facciamo ora delle osservazioni :

$$\begin{aligned} \varepsilon g(\alpha) + \delta \tilde{f}'(\alpha) &\approx 0 \\ \delta &\approx -\frac{\varepsilon g(\alpha)}{\tilde{f}'(\alpha)} \end{aligned} \tag{5.4}$$

Passiamo ai valori assoluti :

$$\begin{aligned} |\delta| &\approx \frac{|\varepsilon g(\alpha)|}{|\tilde{f}'(\alpha)|} \\ |\tilde{\alpha} - \alpha| &\approx \frac{1}{|\tilde{f}'(\alpha)|}|\varepsilon g(\alpha)| \\ &\approx \frac{1}{|\tilde{f}'(\alpha)|}|\tilde{f(\alpha)} - f(\alpha)| \end{aligned} \tag{5.5}$$

Possiamo notare come il valore $\frac{1}{|\tilde{f}'(\alpha)|}$ amplifichi l'errore assoluto sui dati del problema e che il risultato ottenuto da tale prodotto sia uguale (circa) all'errore assoluto che ho sui risultati del problema. Possiamo quindi definire $K = \frac{1}{|\tilde{f}'(\alpha)|}$ il nostro indice di condizionamento.

Avremmo un mal condizionamento nel caso di un valore piccolo di $|\tilde{f}'(\alpha)|$

5.2 Algoritmi iterativi

Per la risoluzione del problema della ricerca degli zeri di una funzione ci avvarremmo di algoritmi definiti iterativi.

Definizione 5.2.1. Un algoritmo si dice esser iterativo quando a partitre da un valore in input x_0, chiamato valore di innesco, genera una successione di valori $x_1, x_2, \ldots, x_i, x_{i+1}, \ldots$ tali che:

$$\lim_{i \to \infty} x_i = \alpha$$

Dove α è il valore del risultato che vogliamo trovare.

Possiamo ulteriormente raggruppare gli algoritmi iterativi per la ricerca degli zeri in due categorie distinte:

- Algoritmi a convergenza globale: Indipendentemente dal valore di innesco x_0 permettono di risalire al valore del risultato α.
- Algoritmi a convergenza locale: Dipendono fortemente dalla scelta di x_0 il quale valore deve essere vicino al valore α per ottenere un risultato corretto. Algoritmi risolutivi di questa tipologia sono molto più veloci rispetto quelli a convergenza globale.

In termini di pseudo codice avremmo :

Algorithm 1 Generico algoritmo iterativo

Require: x_0
Ensure: $x_i \approx \alpha$
  while $COND = TRUE \vee i \geq 0$ **do**
    $x_{i+1} \leftarrow e(x_i)$
  end while

Dove $COND$ è una condizione di arresto per il ciclo while mentre e è una generica funzione che permette di calcolare il valore di x_{i+1} a partire da x_i.

5.2.1 Tipologie di condizioni d'arresto

Le condizioni di arresto di un algoritmo iterativo si basano su tre possibili criteri di arresto i quali operano su degli errori assoluti e relativi.

Definizione 5.2.2. Si definisce errore i-esimo, o errore al passo i, il valore $e_i = |x_i - \alpha|$ dove α è il risultato cercato.

Possiamo utilizzare il valore dell'errore i-esimo per stabilire una prima condizione d'arresto infatti, una volta determinata una tolleranza Tol, possiamo imporre $e_i \leq Tol$.Tuttavia il problema che si presenta è quello che noi ancora non siamo in grado di conoscere il valore di α potremmo dunque riscrivere il valore di e_i come $|x_i - x_{i+1}|$ poichè x_{i+1} può essere considerato una migliore approssimazione di α rispetto al termine x_i e richiedere all'algoritmo di uscire dal loop while quando Tol è minore di $|x_i - x_{i+1}|$.

Questa condizione d'arresto tuttavia non è sicura poiché la tolleranza dovrebbe dipendere dall'ordine di grandezza degli elementi x_i, x_{i+1} utilizzeremo allora un errore relativo :

$$\frac{|x_i - x_{i+1}|}{|x_{i+1}|} \leq Tol$$

Definizione 5.2.3. Dato un valore di tolleranza Tol si definisce criterio d'arresto su valore assoluto la relazione :

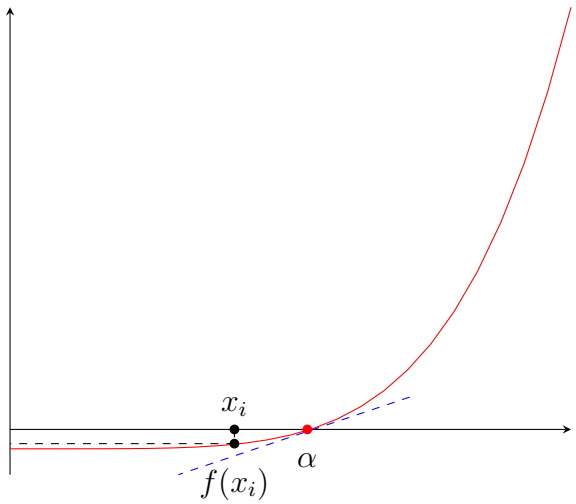
$$|x_i - x_{i+1}| \leq Tol$$

Definizione 5.2.4. Dato un valore di tolleranza Tol si definisce criterio d'arresto su valore relativo la relazione :

$$\frac{|x_i - x_{i+1}|}{|x_{i+1}|} \leq Tol$$

Come detto esiste un 'ulteriore terzo criterio d'arresto chiamato controllo sul residuo in cui viene verificato che la relazione $|f(x_i)| < Tol$ sia rispettata. Quest'ultimo non può essere adoperato da solo come condizione

d'arresto poiché potrebbe tornare una falsa condizione d'uscita, possiamo vedere questo in un esempio grafico :



Ipotizziamo che il valore $|f(x_i)|$ sia più piccolo di una certa tolleranza Tol, in questo caso ci aspettiamo che il corrispondente x_i approssimi in una maniera ottimale il valore di α, tuttavia non è così poichè, come si può vedere, α è molto più distante. Per avere una spiegazione appropriata di cosa stia succedendo occorre osservare la tangente alla figura nel punto α e notare che l'angolo formato tra questo e l'asse orizzontale risulta essere molto piccolo. Questo ci riporta a quanto detto circa il condizionamento del problema in cui il valore di K in questo caso era dato da $\dfrac{1}{f'(\alpha)}$; in figura abbiamo che $f'(\alpha)$ risulta essere molto piccolo e quindi di conseguenza abbiamo che K è molto elevato ed il problema risulta essere mal condizionato.
In conclusione se volessi quindi adoperare anche questo terzo criterio d'arresto lo dovrei fare assieme ad uno degli altri due definiti precedentemente.

5.2.2 Ordine di convergenza

Definizione 5.2.5. Una successione di valori $\{x_i\}_{i\geq 0}$ tendente ad un limite α si dice essere convergente di ordine p se esistono due valori $p \geq 1$ e $C \geq 0$ (fattore di convergenza) tali per cui :

$$\lim_{x\to\infty} \frac{|e_{k+1}|}{|e_k|^p} = C \qquad e_k := x_k - \alpha$$

In altre parole per valori di k grandi risulta valere : $|e_{k+1}| \approx C|e_k|^p$

Definizione 5.2.6. Un metodo iterativo si dice avere ordine di convergenza n se i valori x_i generati durante le varie iterazioni formano una successione avente come ordine di convergenza proprio n.

Per capire l'ordine di convergenza di un algoritmo iterativo vogliamo partire dalla definizione appena data:

$$\begin{aligned}
&|e_{k+1}| \approx C|e_k|^p \\
&|e_{k+2}| \approx C|e_{k+1}|^p \\
&\frac{|e_{k+2}|}{|e_{k+1}|} \approx \left(\frac{|e_{k+1}|}{|e_k|}\right)^p \\
&\log\left(\frac{|e_{k+2}|}{|e_{k+1}|}\right) \approx p \log\left(\frac{|e_{k+1}|}{|e_k|}\right) \\
&p \approx \frac{\log\left(\frac{|e_{k+2}|}{|e_{k+1}|}\right)}{\log\left(\frac{|e_{k+1}|}{|e_k|}\right)}
\end{aligned} \tag{5.6}$$

A questo punto vogliamo ricordare la relazione $e_k := x_k - \alpha$ la quale può essere approssimata in $x_k - x_{k+i}$. Andando a sostituire otterremmo :

$$p \approx \frac{\log\left(\frac{|x_{k+2} - x_{k+3}|}{|x_{k+1} - x_{k+2}|}\right)}{\log\left(\frac{|x_{k+1} - x_{k+2}|}{|x_k - x_{k+1}|}\right)}$$

Nei metodi che studieremo avremo:

- $p = 1$: Convergenza lineare
- $1 < p < 2$: Convergenza superlineare
- $p = 2$: Convergenza quadratica

Il fattore di convergenza C è soggetto ad un ulteriore vincolo nel caso in cui l'ordine di convergenza sia uguale ad 1,ovvero il suo valore deve essere minore di 1; infatti dato $p = 1$ avremmo :

$$
\begin{aligned}
|e_k| &\approx C|e_{k-1}| \\
&\approx C^2|e_{k-2}| \\
&\approx C^3|e_{k-3}| \\
&\vdots \\
&\approx C^n|e_0|
\end{aligned}
\tag{5.7}
$$

Da cui abbiamo $0 < C < 1$.
Nota: La stima di p dati dei valori $x_1, x_2, \ldots, x_n$ è solitamente chiesta all'esame.

5.2.3 Metodo di bisezione

Il metodo di bisezione è un algoritmo a convergenza globale avente come parametri $p = 1$ e $C = \frac{1}{2}$. Il costo computazionale è dato dal numero di iterazioni eseguite +2.

Posso applicare il metodo di bisezione nel caso :

- La funzione sia continua.
- La funzione presenti segni discordi negli estremi dell'intervallo che si osserva.

Algoritmo

L'algoritmo di bisezione prende due valori di ingresso a, b i quali rappresentano gli estremi dell'intervallo di partenza. Ad ogni passo dell'iterazione viene calcolato il punto medio x_i tra questi e vengono formati così due sotto intervalli $[a, x_i]$ e $[x_i, b]$ dei quali scegliamo quello in cui la funzione assume valore discorde alle estremità (poichè sarebbe l'intervallo in cui questa attraversa l'asse delle ascisse). Procedo in questo modo finché non mi ritrovo nella condizione d'arresto (questa viene spiegata poco più avanti).

Il calcolo del punto medio tra $[a_i, b_i]$ deve essere svolto in modo tale che l'operazione sia stabile, ovvero, che il punto medio ricada effettivamente all'interno dell'intervallo; si deve preferire dunque l'operazione $\frac{b_i - a_i}{2} + a_i$ rispetto a $\frac{a_i + b_i}{2}$.
Un ulteriore raccomandazione è quella di non calcolare direttamente il risultato di $f(x_i)$ ma semplicemente ricavarne il segno utilizzando l'apposita funzione $sign()$ fornita dal linguaggio della macchina.

Da queste considerazioni otteniamo il seguente pseudocodice:

Algorithm 2 Algoritmo di bisezione

Require: a, b , $f : \mathbb{R} \mapsto \mathbb{R}$ continua e derivabile
Ensure: $x_{\hat{i}} \approx \alpha$

```
a_0 ← a
b_0 ← b
while COND = TRUE ∨ i ≥ 0 do
    x_{i+1} ← (b_i − a_i)/2 + a_i
    if f(x + 1) × f(a) < 0 then
        a_{i+1} ← a_i
        b_{i+1} ← x_{i+1}
    end if
    if f(x + 1) × f(b) < 0 then
        a_{i+1} ← x_{i+1}
        b_{i+1} ← b_i
    end if
    if f(x + 1) = 0 then
        x_{i+1} ← α
    end if
end while
```

Condizione d'arresto

La condizione d'arresto di questo algoritmo impone di proseguire a suddividere l'intervallo finchè la lunghezza di questo ancora è maggiore o uguale ad una certa tolleranza, ovvero, $|b_i - a_i| < Tol$. Lavorando in aritmetica di macchina tuttavia potremmo ricadere nel seguente problema :

Esempio 5.2.1. Immaginiamoci di avere ad un certo punto delle nostre iterazioni i seguenti valori per gli estremi dell'intervallo considerato :

- $a_i = 98.5$
- $b_i = 98.6$

e si ipotizzi che la tolleranza da considerare sia $Tol = 0.05$.
Il risultato di $|b_i - a_i|$ è 0.1, ciò significa che possiamo continuare con le nostre iterazioni ricavando un nuovo valore x_{i+1}. Svolgendo l'operazione $\frac{a_i+b_i}{2}$ otteniamo come risultato nuovamente $98.6 = b_i$ e, quindi, avremmo che $[a_{i+1}, b_{i+1}] = [a_i, x_{i+1}] = [98.5, 98.6]$ ricadendo in un loop infinito.

Occorre quindi modificare la condizione d'arresto nel seguente modo :

$$|b_i - a_i| < Tol + eps \times \max\{|a_i|, |b_i|\}$$

Costo computazionale

Ad ogni calcolo di x_i occorre calcolare anche il valore di $f(x_i)$ ed, inoltre, all'inizio dell'algoritmo calcoleremo anche $f(a_0), f(b_0)$ avremo quindi un costo computazionale di $n + 2$ dove n è il numero di iterazioni.

Stima dell'errore

Mostriamo infine l'ordine di convergenza del nostro algoritmo.
Al passo n-esimo dovremmo calcolare il punto medio x_n in un intervallo $[a_{n-1}, b_{n-1}]$ e scegliere uno tra gli intervalli così formati $[a_{n-1}, x_n]$ o $[x_n, b_{n-1}]$. L'errore che si va a commettere nello stimare il punto medio è :

$$|e_n| = |x_n - \alpha| \approx \frac{b_{n-1} - a_{n-1}}{2} \approx \frac{b_0 - a_0}{2^n}$$

Possiamo dire allora che :

$$|e_{n-1}| \approx \frac{b_0 - a_0}{2^{n-1}}$$

$$\frac{|e_n|}{|e_{n-1}|} = \frac{1}{2}$$

Da questo risultato possiamo trarre alcune conclusioni :

1. abbiamo mostrato che $C = \frac{1}{2}$ e $p = 1$ dalla relazione $|e_{k+1}| \approx C|e_k|^p$ dove in questo caso $|e_{k+1}| = |e_n|$ e $|e_k| = |e_{n-1}|$.

2. E' possibile stimare il numero di iterazioni necessarie per approssimare α in un numero finito tale per cui $|x_n - \alpha| \leq \varepsilon$

Per questo ultimo punto si procede nel seguente modo:
Anzitutto occorre ricordare che $|x_n - \alpha|$ è l'errore assoluto commesso al n-esimo passo, ovvero, $|e_n|$ il quale è possibile approssimare in $\frac{b_0 - a_0}{2^n}$. Data quindi la relazione $\frac{b_0 - a_0}{2^n} \leq \varepsilon$ vogliamo che la disuguaglianza sia verificata e vogliamo ricavarne da questa il valore di n:

$$
\begin{aligned}
\frac{b_0 - a_0}{2^n} &\leq \varepsilon \\
2^n &\geq \frac{b_0 - a_0}{\varepsilon} \\
n &\geq \log_2 \frac{b_0 - a_0}{\varepsilon} = \frac{\log_{10} \frac{b_0 - a_0}{\varepsilon}}{\log_{10} 2} \\
n &\approx 3.3 \times \log_{10} \frac{b_0 - a_0}{\varepsilon} \\
n_\varepsilon &\approx \lceil 3.3 \times \log_{10} \frac{b_0 - a_0}{\varepsilon} \rceil
\end{aligned}
\tag{5.8}
$$

5.2.4 Metodo di falsa posizione (regula falsi)

Il metodo di falsa posizione è un miglioramento dell'algoritmo di bisezione in quanto si basa su principi iterativi simili a quest'ultimo. Le condizioni di applicazione rimangono invariate.

Algoritmo

L'algoritmo prende in ingresso due valori a, b i quali determinano l'intervallo della funzione che deve essere analizzato nella ricerca degli zeri. Si valuta a questo punto la funzione negli estremi dell'intervallo ottenendo come risultato i valori $f(a), f(b)$ e si traccia successivamente la retta passante nei punti $(a, f(a))$ e $(b, f(b))$; verrà individuato in questo modo un punto x_i il quale rappresenta l'intersezione della retta creata con l'asse delle ascisse. Analogamente al metodo di bisezione, a questo punto, si sceglie uno tra gli intervalli creati $[a, x_i]$ ed $[x_i, b]$ in cui cui la funzione assume segno discorde agli estremi e si ripete l'iterazione con tale intervallo scelto.

Per quanto riguarda lo pseudocodice del metodo di falsa posizione possiamo partire da quello di bisezione facendo alcuni accorgimenti circa il calcolo del punto x_i che abbiamo visto essere il punto di intersezione tra l'asse delle ascisse e la retta passante tra $(a, f(a))$ e $(b, f(b))$.

$$
\begin{cases}
y - f(a_i) = \frac{f(a_i) - f(b_i)}{b_i - a_i}(x - a_i) \\
y = 0
\end{cases}
$$

per la quale avremo $x_i = a_i - f(a_i)\frac{b_i - a_i}{f(b_i) - f(a_i)}$.

Si immagini di calcolare il valore di ogni iterata di x_i usando al posto di $f(a_i)$ e $f(b_i)$ la funzione $sign(f(x))$.
Abbiamo due casi distinti:

1. $sign(f(a_i)) = +$, $sign(f(b_i)) = -$

$$x_i = a_i - \frac{b_i - a_i}{-1 - 1} = a_i + \frac{b_i - a_i}{2}$$

2. $sign(f(a_i)) = -$, $sign(f(b_i)) = +$

$$x_i = a_i + \frac{b_i - a_i}{1 - (-1)} = a_i + \frac{b_i - a_i}{2}$$

Grazie a questo risultato abbiamo espresso in un'ulteriore maniera l'algoritmo di bisezione utilizzando il metodo di falsa posizione.

Algorithm 3 Algoritmo di falsa posizione

Require: a, b , $f : \mathbb{R} \mapsto \mathbb{R}$ continua e derivabile
Ensure: $x_{\hat{i}} \approx \alpha$

```
a_0 ← a
b_0 ← b
while COND = TRUE ∨ i ≥ 0 do
    x_{i+1} ← a_i − f(a_i) (b_i − a_i)/(f(b_i) − f(a_i))
    if f(x + 1) × f(a) < 0 then
        a_{i+1} ← a_i
        b_{i+1} ← x_{i+1}
    end if
    if f(x + 1) × f(b) < 0 then
        a_{i+1} ← x_{i+1}
        b_{i+1} ← b_i
    end if
    if f(x + 1) = 0 then
        x_{i+1} ← α
    end if
end while
```

Condizione d'arresto

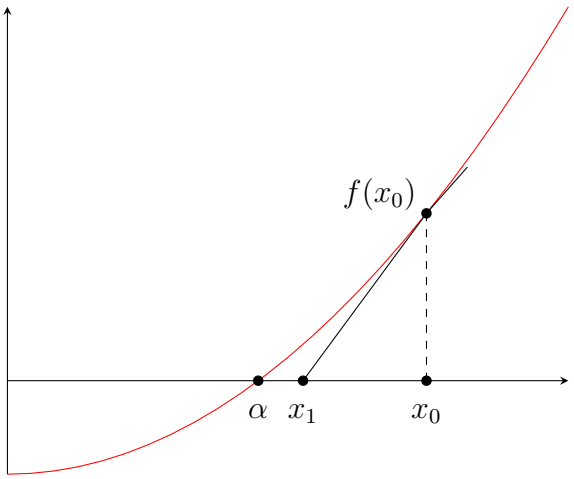
In questo caso non possiamo basare la nostra condizione d'arresto sull'ampiezza dell'intervallo della funzione; occorrerà quindi adoperare il controllo sull'incremento e sul residuo.

Costo computazionale

Rimane invariato rispetto al metodo di bisezione ovvero $n + 2$ dove n è il numero di iterazioni.

5.2.5 Metodi di Newton, Corde e Secanti

Vogliamo introdurre ora una nuova classe di metodi per calcolare gli zeri di una funzione. Data una funzione $f(x)$ definita in un intervallo in cui è possibile individuare al massimo una sua radice vogliamo scegliere un valore di innesco x_0 tramite il quale sarà possibile localmente approssimare la funzione di partenza attraverso la retta di equazione $y = f(x_0) + m_0(x - x_0)$ in cui $m_0 \neq 0$ è il coefficiente angolare della funzione.



Costruisco una successione di valori x_{i+1} con $i \geq 0$ in cui ciascuno di questi è l'intersezione tra l'asse delle ascisse e la retta $y = f(x_0) + m_0(x - x_0)$:

$$x_{i+1} = x_i - \frac{f(x_i)}{m_i}$$

A seconda della scelta dei valori m_i otteniamo :

1. Metodo di Newton : $m_i = f^{'}(x_i)$, ovvero all' i-esimo passo la retta che andrò a costruire è la tangente in quel punto.

2. Metodo delle corde : $m_i = m$, scelgo un coefficiente costante ad ogni iterazione. Le condizioni necessarie per l'utilizzo del metodo delle corde sono le seguenti :

Data una funzione $f : [a, b] \mapsto R$ tale che esista α per la quale la funzione si annulla in quel punto e $f(x)$ sia derivabile scegliamo m in base ai seguenti tre principi

- $f^{'}(x) \neq 0$
- $mf^{'}(x) > 0$
- $m > \max_{[a,b]} \frac{1}{2} f^{'}(x)$

3. Metodo delle secanti : $m_i = \dfrac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$ ovvero approssimo il grafico della funzione con le rette passanti per i punti $(x_{i-1}, f(x_{x_{i-1}}))$ e $(x_i, f(x_i))$. In questo caso avremmo bisogno di due valori di innesco x_{-1}, x_0 per la prima iterata

5.2.6 Convergenza del metodo di Newton

Lavorando con delle funzioni a concavità fissa è possibile stabilire un valore di innesco x_0 che garantisce la convergenza del metodo di Newton.

Definizione 5.2.7. Sia data una funzione f definita continua e convessa all'interno di un intervallo $[a, b]$ nel quale assume segni discordi negli estremi (e quindi $f(a)f(b) < 0$) si dice estremo di Fourier l'estremo verso il quale la funzione rivolge la concavità.

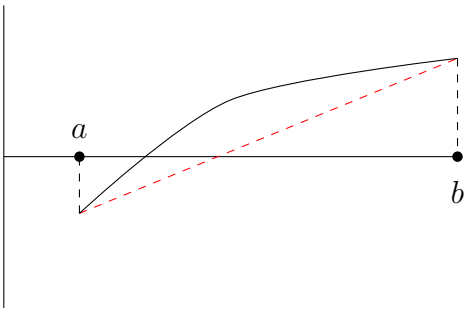


Figura 5.1: a è l'estremo di Fourier

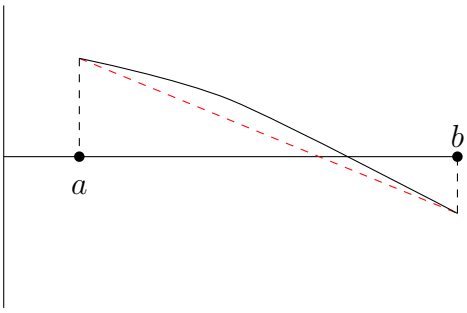


Figura 5.2: b è l'estremo di Fourier

Nell'esempio raffigurato qui sopra abbiamo tracciato una retta tra i punti $f(a)$ e $f(b)$ di entrambe le funzioni la quale divide queste in due parti. Dato che stiamo parlando di funzioni a concavità fissa una delle parti separate dalla retta passante tra $f(a)$ e $f(b)$ conterrà interamente tale concavità definiremo quindi estremo di Fourier uno dei due estremi a o b

compresi nella parte di funzione separata dalla retta $f(a)f(b)$ contenente la concavità.
Se una funzione $f[a,b] \mapsto \mathbb{R}$ rispetta le seguenti proprietà :

- $f(a)f(b) < 0$
- $f, f^{'}, f^{''}$ sono funzioni continue in $[a,b]$, ovvero $f \in C^2[a,b]$
- $f^{'}(x) \neq 0 \quad \forall x \in [a,b]$
- $f^{''}(x) \neq 0 \quad \forall x \in [a,b]$

E si sceglie come punto di innesco x_0 l'estremo di Fourier nell'intervallo $[a,b]$ allora il metodo di Newton produrrà una serie $x_{ii\geq 1}$ monotona (crescente se l'estremo di fourier è a decrescente se invece è b) che converge all'unica radice $\alpha \in [a,b]$ e tale convergenza è superlineare.

Inoltre se la funzione $f \in C^3[a,b]$ allora la convergenza è quadratica e vale inoltre :

$$\lim_{i\to\infty} \frac{|x_{i-1} - \alpha|}{(x_i - \alpha)^2} = \frac{f^{''}(\alpha)}{2f^{'}(\alpha)}$$

quindi $p = 2$ e $C = \dfrac{f^{''}(\alpha)}{2f^{'}(\alpha)}$

5.2.7 Metodi iterativi a punto fisso

I metodi iterativi a punto fisso sono algoritmi che permettono in maniera del tutto generica di calcolare gli zeri di una funzione $f(x)$ non lineare riconducendoci ad una funzione $g(x)$ detta funzione di iterazione tale per cui vale la relazione :

$$f(\alpha) = 0 \Longleftrightarrow g(\alpha) = \alpha$$

in questo modo quindi non mi interesso più a cercare il valore α per cui $f(\alpha) = 0$ ma bensì $g(\alpha) = \alpha$. Graficamente quello tentiamo di fare è individuare il punto di intersezione tra la retta di equazione $y = x$ e la funzione g. La funzione di iterazione $g(x)$ non è unica e possiamo ottenerla in svariati modi:

Esempio 5.2.2. Sia data la funzione $f(x) = x^3 + 4x - 10$ che vogliamo considerare nell'intervallo $[0,2]$ calcoliamo $\alpha \in [0,2]$ tale che $f(\alpha) = 0$.

Elenchiamo ora alcuni modi che abbiamo nel calcolare la nostra funzione $g(x)$

1. Aggiungiamo ad entrambi i membri di $x^3 + 4x - 10 = 0$ una x e ricaviamo di conseguenza la nostra $g(x)$.

$$\begin{aligned} x^3 + 4x - 10 &= 0 \\ x^3 + 4x + x - 10 &= x \end{aligned} \tag{5.9}$$

2. Dall'ultima equazione che ho trovato sposto il numero 10 a destra dell'equazione e raccolgo a primo membro una x in comune :

$$\begin{aligned} x^3 + 4x + x - 10 &= x \\ x^3 + 4x + x &= x + 10 \\ x(x^2 + 4 + 1) &= x + 10 \end{aligned} \tag{5.10}$$

   Impongo successivamente che $(x^2 + 4 + 1) \neq 0$ e divido entrambi i membri dell'equazione per questa quantità:

$$x = \frac{x + 10}{x^2 + 4 + 1}$$

Algoritmo

Prendiamo in input un valore di innesco x_0 e calcoliamo le varie iterate x_{i+1} attraverso la nostra funzione $g(x)$. Il criterio d'arresto che adopereremo questa volta sarà semplicemente quello che controlla l'incremento, non ci interessa controllare il residuo !

Algorithm 4 Algoritmo punto fisso

Require: x_0 , $g : \mathbb{R} \mapsto \mathbb{R}$
  while $COND = TRUE \vee i \geq 0$ **do**
    $x_{i+1} \leftarrow g(x_i)$
  end while

Se scelgo $g(x) = x - \dfrac{f(x)}{f'(x)}$ posso pensare al metodo di Newton, discusso in precedenza, come ad un particolare caso di un algoritmo a punto fisso.

Analisi della convergenza

Vogliamo analizzare ora le condizioni che deve rispettare la funzione g per garantire la convergenza di questo metodo iterativo ad α.

Teorema convergenza globale: Sia data una successione di punti x_i dove $xi + 1 = g(xi)$ per fare in modo che questa converga ad α deve rispettare le seguenti condizioni:

1. $g : [a, b] \mapsto [a, b]$
2. $g \in C^1[a, b]$
3. $\exists C < 1 : \ |g^{'}(x)| \leq C \ \forall x \in [a, b]$

In questo caso la nostra g ha un unico punto fisso α e qualunque valore di innesco x_0 si scelga all'interno dell'intervallo $[a, b]$ la funzione è convergente e si ha inoltre :

$$\lim_{i \to \infty} \frac{x_{i+1} - \alpha}{x_i - \alpha} = |g^{'}(\alpha)|$$

Teorema convergenza locale: Sia α un punto fisso della funzione $g \in C^1[\alpha - \rho, \alpha + \rho]$ con $\rho > 0$ se :

$$|g^{'}(x)| < 1, \forall x \in [\alpha - \rho, \alpha + \rho]$$

Allora per ciascun valore di $x_0 \in [\alpha - \rho, \alpha + \rho]$ la successione di punti generati dalla funzione $g(x_i)$ è tale che :

1. $x_i \in [\alpha - \rho, \alpha + \rho]$.
2. $\lim_{i \to \infty} x_i = \alpha$ unico punto fisso di g.

Teorema dell'ordine di convergenza: Sia $\alpha \in [I]$ il punto fisso di una funzione $g \in C^p[I]$, dove I è un opportuno intervallo, preso punto x_0 la successione x_{i+1} è convergente se :

$$g^{'}(\alpha) = g^{''}(\alpha) = g^{'''}(\alpha) = \cdots = g^{p-1}(\alpha) = 0$$

in questo caso l'ordine di convergenza è p e risulta inoltre che :

$$\lim_{i \to +\infty} \frac{|x_{i+1} - \alpha|}{|x_i - \alpha|^p} = C \qquad C = \frac{|g^p(\alpha)|}{p!}$$

Capitolo 6

Sistemi lineari

Nel seguente capitolo vogliamo studiare come risolvere sistemi lineari attraverso l'utilizzo di un calcolatore.

Definizione 6.0.1. Un sistema lineare quadrato di dimensione n è un sistema lineare costituito da n equazioni in n incognite. In forma matriciale lo possiamo scrivere come $A_{n\times n}x_{n\times 1} = b_{n\times 1}$.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & & \ddots & & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

Che possiamo rappresentare come sistema :

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & = b_2 \\ \vdots & \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{nn}x_n & = b_n \end{cases}$$

Esiste un'unica soluzione se e solo se il determinante della matrice A è diverso da zero, ovvero $\det(A) \neq 0$.

Il sistema lineare risulta un problema ben condizionato quando il numero di condizionamento della matrice A ($K(A) = \|A\|\|A^{-1}\|$) non risulta troppo elevato.

6.1 Metodi diretti

Sia dato un sistema lineare $Ax = b$ tale per cui il $\det(A) \neq 0$ e il suo numero di condizionamento $K(A)$ risulti non troppo elevato. Risolvere questo sistema tramite applicazione di metodi diretti significa trasformarlo, tramite un numero finito di passi, in un sistema lineare equivalente, ovvero avente la medesima soluzione, di più facile soluzione e tale sistema presenterà una matrice dei coefficienti triangolare (inferiore o superiore).

6.1.1 Metodi di sostituzione per la soluzione di sistemi lineari con matrice dei coefficienti in forma triangolare

Abbiamo due tipologie differenti di metodi:

1. Metodo di sostituzione **in avanti** per sistemi lineari con matrice **triangolare inferiore.**

$$\begin{pmatrix} a_{11} & 0 & 0 & 0 & \dots & 0 \\ a_{21} & a_{22} & 0 & 0 & \dots & 0 \\ \vdots & & \ddots & & & \vdots \\ a_{i1} & a_{i2} & \dots & a_{ii} & \dots & 0 \\ \vdots & & & & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & \dots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_i \\ \vdots \\ b_n \end{pmatrix}$$

   Il relativo sistema risultante ci permette di ottenere i valori delle incognite $x_1, x_2, \dots, x_n$ in una maniera più agevole poichè si presenterà nel seguente modo :

$$\begin{cases} a_{11}x_1 = b_1 \longrightarrow x_1 = \frac{b_1}{a_{11}} \\ a_{21}x_1 + a_{22}x_2 = b_2 \longrightarrow x_2 = \frac{b_2 - a_{21}x_1}{a_{22}} \\ \vdots \\ a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{ii}x_i = b_i \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{ni}x_i + \cdots + a_{nn}x_n = b_n \end{cases}$$

   Possiamo riscrivere una generica riga i del nostro sistema come una sommatoria di i fattori, ovvero :

$$\sum_{j=1}^{i} a_{ji}x_j = b_i$$

Dunque la generica soluzione di xi può essere scritta allo stesso modo in funzione di una sommatoria in quanto :

$$x_i = \frac{b_i - \sum_{j=1}^{i-1} a_{ji} x_j}{a_{ii}}$$

Questo ci permette dunque di determinare un algoritmo risolutivo :

Algorithm 5 Metodo di sostituzione in avanti

$x_1 \leftarrow \frac{b_1}{a_{11}}$
for $i = 2, 3 \dots n$ **do**
  $x_i \leftarrow \frac{b_i - \sum_{j=1}^{i-1} a_{ji} x_j}{a_{ii}}$
end for

E' da notare che i fattori $a_{11}, a_{22}, \dots, a_{nn}$ non si annullino nel nostro algoritmo poichè abbiamo richiesto che la nostra matrice non abbia determinante nullo.

Costo computazionale: ad ogni i-esimo passaggio effettuiamo $n-1$ moltiplicazioni e 1 divisione per un totale di i operazioni distinte, ripetiamo queste i operazioni per n volte.

$$\sum_{i=1}^{n} i = \frac{n(n-1)}{2} \approx \frac{n^2}{2}$$

2. Metodo di sostituzione **all'indietro** per sistemi lineari con matrice **triangolare superiore.**

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & \dots & a_{1n} \\ 0 & a_{22} & a_{23} & \dots & \dots & a_{2n} \\ \vdots & & \ddots & & & \vdots \\ 0 & 0 & \dots & a_{ii} & \dots & a_{in} \\ \vdots & & & & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_i \\ \vdots \\ b_n \end{pmatrix}$$

Stiamo lavorando con un caso analogo a quello visto in precedenza dove avremmo ancora una matrice dei coefficienti avente determinante diverso da zero. In questo caso, per la risoluzione partiremo dalla n-esima equazione invece che dalla prima.

$$\begin{cases} a_{nn}x_1 = b_n \longrightarrow x_1 = \frac{b_n}{a_{nn}} \\ a_{n-1n-11}x_1 + a_{n-1n}x_2 = b_{n-1} \longrightarrow x_2 = \frac{b_{n-1}-(a_{n-1n-1}x_1)}{a_{n-1n}} \\ \vdots \\ a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{ii}x_i = b_i \\ \vdots \\ a_{11}x_1 + a_{12}x_2 + \cdots + a_{1i}x_i + \cdots + a_{1n}x_n = b_1 \end{cases}$$

In questo caso possiamo scrivere l'equazione di una generica riga i nel seguente modo:

$$\sum_{j=1}^{i} a_{ij}x_j = b_i$$

$$x_i = \frac{b_i - \sum_{j=1}^{i-1} a_{ij}x_j}{a_{ii}}$$

Il relativo pseudocodice :

Algorithm 6 Metodo di sostituzione all'indietro

$x_1 \leftarrow \frac{b_n}{a_{nn}}$
for $i = n-1, \ldots, 1$ **do**
  $x_i \leftarrow \frac{b_i - \sum_{j=i+1}^{n} a_{ij}x_j}{a_{ii}}$
end for

Costo computazionale: ad ogni i-esimo passaggio effettuiamo $n-i$ moltiplicazioni e 1 divisione per un totale di $n-i+1$ operazioni distinte, ripetiamo queste $n-i+1$ operazioni per n volte.

$$\sum_{i=1}^{n} n-i+1 = \sum_{j=1}^{n} j = \frac{n(n-1)}{2} \approx \frac{n^2}{2}$$

Da notare che abbiamo posto $j = n-i+1$ ottenendo lo stesso costo computazionale del primo caso.

Come useremo i metodi diretti per risolvere un sistema lineare ?
Dato un sistema del tipo $Ax = b$ con A matrice $n \times n$ e $\det(A) \neq 0$ ci ricondurremo ad un sistema $Cx = y$ equivalente e tale per cui C è

triangolare superiore. Una caratteristica comune che avranno i metodi che studieremo è quella della fattorizzazione, ovvero, per ottenere il sistema $Cx = y$ a partire da $Ax = b$ moltiplicheremo la matrice triangolare superiore C per una matrice B tale che :

$$A = BC$$

dove A, B, C sono matrici con dimensioni $n \times n$ e B può essere :

- triangolare inferiore.
- ortogonale.

Tramite la fattorizzazione possiamo riscrivere il sistema $Ax = b$ nel relativo equivalente $BCx = b$ il quale vogliamo scomporre in due ulteriori sotto sistemi lineari ponendo $Cx = y$. Avremmo dunque :

$$\begin{cases} By = b \\ Cx = y \end{cases}$$

Conoscendo il primo posso ricavare il vettore y che andrò poi a sostituire al secondo membro per trovare x. Nel caso B sia triangolare inferiore risolverò il sistema $By = b$ tramite metodo di sostituzione all'avanti mentre nel caso in cui B è ortogonale troverò y semplicemente moltiplicando B^T con b, ovvero:

$$y = B^T b$$

Per quanto riguarda i costi derivanti dalla fattorizzazione se $A = BC$ con B triangolare inferiore e C triangolare superiore avremmo che il costo computazionale per $By = b$ e $Cx = y$ è $\frac{n^2}{2}$.

6.1.2 Metodo di gauss o eliminazione gaussiana

Sia data la matrice $A = LU$ avente :

- L triangolare inferiore e diagonale 1
- U triangolare superiore

La condizioni che dobbiamo richiedere affinché le matrici L ed U esistano è :

$$\det(A(1:k, 1:k)) \neq 0 \quad \forall k = 1, \ldots, n-1$$

$A(1:k, 1:k)$ è detta sotto matrice principale di testa e si ottiene attraverso il ritaglio delle prime k righe e k colonne. Di seguito viene mostrato un esempio:
Data la matrice

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & a_{24} & \dots & a_{2n} \\ \vdots & & \ddots & & & \vdots \\ a_{i1} & a_{i2} & \dots & a_{ii} & \dots & a_{in} \\ \vdots & & & & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & \dots & a_{nn} \end{pmatrix}$$

Le sue matrici di testa saranno

$$A(1:1, 1:1) = a_{11}, \quad A(1:2, 1:2) = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, \dots,$$

Se i determinanti delle $n-1$ matrici principali di testa di testa di A sono diversi da zero allora esiste ed è unica la fattorizzazione LU della matrice A dove :

- L è matrice diagonale inferiore con elementi diagonali 1.
- U è matrice triangolare superiore.

Esempio 6.1.1.

$$A = \begin{pmatrix} -5 & 2 & 1 & 8 \\ 20 & -5 & -3 & -28 \\ -30 & 18 & 7 & 54 \\ -15 & 27 & 5 & 51 \end{pmatrix}, \quad n = 4$$

Vogliamo trasformare la matrice A di partenza in una matrice triangolare superiore U applicando un metodo diretto che il quale permette di arrivare al nostro scopo in un numero finito di passi i quali sono esattamente $n-1$. In questo caso quindi ci basteranno 3 passi per trasformare A in U creando le matrici elementari di gauss da applicare ad A.

1. Nominiamo la matrice in input in $A^{(1)}$, quindi $A = A^{(1)}$
2. Costruisco la prima matrice elementare di Gauss di dimensione $n \times n$ conformata in questo modo:

$$M^{(1)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -m_{21}^{(1)} & 1 & 0 & 0 \\ -m_{31}^{(1)} & 0 & 1 & 0 \\ -m_{41}^{(1)} & 0 & 0 & 1 \end{pmatrix}$$

Notiamo subito anzitutto che la matrice ha sulla sua diagonale tutti elementi uguali ad 1 e i restanti uguali a zero fatta eccezione per la prima colonna di questa la quale contiene gli elementi $m_{21}^{(1)}, m_{31}^{(1)}, m_{41}^{(1)}$ aventi segno negativo. Questi vengono detti moltiplicatori della radice gaussiana al passo 1 e vengono calcolati nel seguente modo :

$$m_{21}^{(1)} = \frac{a_{21}^{(1)}}{a_{11}^{(1)}} = -4 \quad m_{31}^{(1)} = \frac{a_{31}^{(1)}}{a_{11}^{(1)}} = 6 \quad m_{41}^{(1)} = \frac{a_{41}^{(1)}}{a_{11}^{(1)}} = 3$$

La nostra matrice sarà allora :

$$M^{(1)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 4 & 1 & 0 & 0 \\ -6 & 0 & 1 & 0 \\ -3 & 0 & 0 & 1 \end{pmatrix}$$

3. Costruisco la matrice $A^{(2)}$:

$$A^{(2)} = M^{(1)}A^{(1)} = \begin{pmatrix} -5 & 2 & 1 & 8 \\ 0 & 3 & 1 & 4 \\ 0 & 6 & 1 & 6 \\ 0 & 21 & 2 & 27 \end{pmatrix}$$

Ho azzerato gli elementi della prima colonna della mia matrice di partenza al di sotto della diagonale.

4. Costruisco la seconda matrice elementare di Gauss $M^{(2)}$ usando gli stessi passi adoperati per la prima.

$$M^{(2)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -m_{32}^{(2)} & 1 & 0 \\ 0 & -m_{42}^{(2)} & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & -7 & 0 & 1 \end{pmatrix}$$

$$m_{32}^{(2)} = \frac{a_{32}^{(2)}}{a_{22}^{(2)}} = 2 \quad m_{42}^{(2)} = \frac{a_{42}^{(2)}}{a_{22}^{(2)}} = 7$$

5. Costruisco la matrice $A^{(3)}$:

$$A^{(3)} = M^{(2)}A^{(2)} = \begin{pmatrix} -5 & 2 & 1 & 8 \\ 0 & 3 & 1 & 4 \\ 0 & 0 & -1 & -2 \\ 0 & 0 & -5 & 1 \end{pmatrix}$$

Come previsto anche in questo caso, abbiamo che gli elementi della colonna sotto la diagonale si sono azzerati, procediamo dunque gli ultimi passaggio.

6. Costruisco la terza, e ultima, matrice elementare di Gauss $M^{(3)}$ usando gli stessi passi adoperati per la prima e la seconda.

$$M^{(2)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -m_{43}^{(3)} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -5 & 1 \end{pmatrix}$$

$$m_{43}^{(3)} = \frac{a_{43}^{(3)}}{a_{33}^{(3)}} = 5$$

7. Costruisco la matrice $A^{(4)}$:

$$A^{(4)} = M^{(3)}A^{(4)} = \begin{pmatrix} -5 & 2 & 1 & 8 \\ 0 & 3 & 1 & 4 \\ 0 & 0 & -1 & -2 \\ 0 & 0 & 0 & 9 \end{pmatrix}$$

La matrice trovata nell'ultimo passaggio è la nostra U che, da come possiamo notare, è triangolare superiore.

In questo esempio abbiamo potuto sicuramente notare quanto sia fondamentale il fatto che i nostri vari elementi pivotali $a_{11}^{(1)}, a_{22}^{(2)}, a_{33}^{(3)}$ siano diversi da zero. La condizione che il determinante di tutte le matrici di testa sia diverso da zero è necessaria per il rapporto che sussiste tra tali elementi in quanto :

$$a_{22}^{(2)} = \frac{\det(A[1:2,1:2])}{\det(A[1:1,1:1])} \quad a_{33}^{(3)} = \frac{\det(A[1:3,1:3])}{\det(A[1:2,1:2])}$$

A questo punto possiamo ricavare il nostro pseudocodice :

Algorithm 7 Metodo di eliminazione Gaussiana

Require: A, matrice $n \times n$
Ensure: $U := A^{(n)}$
  $A^{(1)} \leftarrow A$
  for $k = 1, \dots, n-1$ **do**
    for $r = k+1, \dots, n$ **do**
      $m_{rk}^{(k)} \leftarrow \frac{a_{rk}^{(k)}}{a_{kk}^{(k)}}$
    end for

$$M^{(k)} \leftarrow \begin{pmatrix} 1 & 0 & 0 & \dots & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & \dots & 0 & 0 \\ \vdots & & \ddots & \dots & \dots & 0 & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 & 0 \\ 0 & 0 & 0 & -m_{k+1,k}^{(k)} & \ddots & 0 & 0 \\ \vdots & & & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & -m_{k+1,n}^{(k)} & \dots & 0 & 1 \end{pmatrix}$$

$$A^{(K+1)} \leftarrow A^{(k)} M^{(k)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & \dots & \dots & a_{1n-1}^{(1)} & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \dots & \dots & a_{2n-1}^{(2)} & a_{2n}^{(2)} \\ \vdots & & \ddots & & & & \vdots \\ 0 & 0 & 0 & a_{kk}^{(k)} & \dots & a_{kn-1}^{(k)} & a_{kn}^{(k)} \\ 0 & 0 & 0 & 0 & a_{k+1,k+1}^{(k)} & \dots & a_{k+1,n}^{(k+1)} \\ \vdots & & & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & 0 & a_{n,k+1}^{(n)} & \dots & a_{n,n}^{(n)} \end{pmatrix}$$

  end for

possiamo scrivere $M^{(k)}$ come $I - m^{(k)} e_k^T$ dove

$$m^{(k)} = \begin{pmatrix} 0_1 \\ \vdots \\ 0_k \\ m_{k+1,k}^k \\ \vdots \\ m_{n,k}^k \end{pmatrix} \quad e_k = \begin{pmatrix} 0_1 \\ \vdots \\ 0_{k-1} \\ 1 \\ 0_{k+1} \\ \vdots \\ 0_n \end{pmatrix}$$

Rappresentare in questo modo la generica matrice di Gauss al passo k ci permette di rappresentare ciò che è la matrice L tale per cui $A = LU$.

L si presenta nella seguente forma :

$$L = \begin{pmatrix} 1 & 0 & \dots & 0 & 0 \\ m_{21}^{(1)} & 1 & \dots & 0 & 0 \\ m_{31}^{(1)} & m_{32}^{(2)} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{n1}^{(1)} & m_{n2}^{(2)} & \dots & m_{n,n-1}^{(n-1)} & 1 \end{pmatrix}$$

Tramite l'algoritmo di prima abbiamo visto che :

$$U := A^{(n)} = M^{(n-1)} M^{(n-2)} \dots M^{(2)} M^{(1)} A$$

Tutte le matrici $M^{(1)}, M^{(2)}, \dots, M^{(n-1)}$ sono invertibili in quanto sono triangolari inferiori aventi come diagonale proprio 1. Si vuole dimostrare ora che vale la relazione :

$$(M^{(k)})^{-1} = I + m^{(k)} e_k^T$$

Per verificare ciò occorrerà semplicemente vedere $M^{(k)}(M^{(k)})^{-1} = I$

$$\begin{aligned} M^{(k)}(M^{(k)})^{-1} &= (I - m^{(k)} e_k^T)(I + m^{(k)} e_k^T) \\ &= I - m^{(k)} e_k^T + m^{(k)} e_k^T - m^{(k)} e_k^T m^{(k)} e_k^T \\ &= I \end{aligned} \tag{6.1}$$

questo è dovuto al fatto che $e_k^T m^{(k)} = 0$ poiché l'unico valore 1 del vettore e_k^T moltiplica lo 0 alla posizione k del vettore $m^{(k)}$ e i valori restanti sono tutti degli zeri. Quindi il risultato dell'operazione $m^{(k)} e_k^T m^{(k)} e_k^T = 0$.

$$U := A^{(n)} = M^{(n-1)} M^{(n-2)} \dots M^{(2)} M^{(1)} A = (M^{(1)})^{-1} (M^{(2)})^{-1} \dots (M^{(n-2)})^{-1} (M^{(n-1)})^{-1} A$$

$$L := (M^{(1)})^{-1} (M^{(2)})^{-1} \dots (M^{(n-2)})^{-1} (M^{(n-1)})^{-1}$$

Calcoliamo infine gli elementi a_{ij}^{k+1} della matrice A con $i, j = k+1, \dots, n$.

$$a_{i,j}^{k+1} = M^k(i,:) \times A^k(:,j)$$

dove $M^k(i,:)$ corrisponde al vettore dell'i-esima riga della matrice M^k e $A^k(:,j)$ al vettore della j-esima colonna della matrice A^k.

Quindi:

$$a_{i,j}^{(k+1)} = \begin{pmatrix} 0, & \dots & 0, & -m_{ik}^k, & \dots & 1, & \dots & 0 \end{pmatrix} \times \begin{pmatrix} \dots \\ \dots \\ a_{kj}^k \\ \dots \\ a_{ij}^k \\ \dots \\ \dots \end{pmatrix} = a_{ij}^{(k)} - m_{ik}^{(k)} a_{kj}^{(k)}$$

Tramite questo risultato possiamo riscrivere in una maniera più agile il nostro di eliminazione gaussiana

Algorithm 8 Metodo di eliminazione Gaussiana v2

Require: A, matrice $n \times n$
Ensure: L, U
  $A^{(1)} \leftarrow A$
  for $k = 1, \dots, n-1$ **do**
    for $r = k+1, \dots, n$ **do**
      $m_{rk}^{(k)} \leftarrow \frac{a_{rk}^{(k)}}{a_{kk}^{(k)}}$
    end for
    for $i = k+1, \dots, n$ **do**
      for $j = k+1, \dots, n$ **do**
        $a_{i,j}^{k+1} \leftarrow a_{ij}^{(k)} - m_{ik}^{(k)} a_{kj}^{(k)}$
      end for
    end for
  end for

di cui il costo computazionale :

$$\sum_{k=1}^{n-1} \left[(n-k) + (n-k)^2 \right] \approx \sum_{k=1}^{n-1} (n-k)^2 = \sum_{j=1}^{n-1} j^2 = \frac{(n-1)n(2n-1)}{6} \approx \frac{n^3}{3}$$

6.1.3 Errori algoritmici nel metodo di eliminazione gaussiana

Si consideri il seguente esempio:

Esempio 6.1.2. Risoluzione del seguente sistema lineare attraverso l'algoritmo di Gauss:

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1.0001 & 2 \\ 1 & 2 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$$

Esattamente come prima definiamo la nostra matrice di partenza $A^{(1)}$ e costruiamo di conseguenza $M^{(1)}$.

$$M^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}$$

Applichiamo $M^{(1)}$ ad $A^{(1)}$

$$A^{(2)} = M^{(1)}A^{(1)} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 10^{-4} & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

Costruiamo $M^{(2)}$

$$M^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -10^4 & 1 \end{pmatrix}$$

Applichiamo $M^{(2)}$ ad $A^{(2)}$

$$A^{(3)} = M^{(2)}A^{(2)} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 10^{-4} & 1 \\ 0 & 0 & (1 - 10^{-4}) \end{pmatrix}$$

Una volta ottenuta una matrice triangolare applichiamo anche ai vettori dei termini noti le matrici elementari di Gauss :

$$b^{(1)} = b = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$$

$$b^{(2)} = M^{(1)}b^{(1)} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \quad b^{(3)} = M^{(2)}b^{(2)} = \begin{pmatrix} 1 \\ 1 \\ -10^4 \end{pmatrix} = y$$

A questo punto possiamo riscrivere il sistema che abbiamo ottenuto :

$$Ax = b \Longleftrightarrow \begin{pmatrix} 1 & 1 & 1 \\ 0 & 10^{-4} & 1 \\ 0 & 0 & (1 - 10^{-4}) \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ -10^4 \end{pmatrix}$$

Con il metodo di sostituzione all'indietro troviamo che i seguenti risultati

- $\tilde{x_3} = fl_A\left(-\frac{10^4}{1-10^{-4}}\right) = fl_A(0.10001\cdots \times 10) = 1$

- $\tilde{x_2} = fl_A\left(\frac{1-\tilde{x_3}}{10^{-4}}\right) = fl_A(\frac{1-1}{10^{-4}}) = 0$
- $\tilde{x_1} = fl_A\left(1 - \tilde{x_2} - \tilde{x_3}\right) = fl_A(1 - 0 - 1) = 0$

Risultato :

$$\tilde{x} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Quanto ottenuto non è preciso in quanto il risultato esatto è :

$$\tilde{x} = \begin{pmatrix} 1 \\ -1.0001\ldots \\ 1.0001\ldots \end{pmatrix}$$

Si può notare quindi un errore di approssimazione dei termini $\tilde{x_1}, \tilde{x_2}$ dovuto dalla presenza di un valore molto piccolo in $a_{22}^{(2)} = 10^{-4}$ il quale produce un moltiplicatore molto elevato.

6.1.4 Pivoting parziale

Il pivoting parziale è una tecnica che ci permette di ridurre l'errore algoritmico generato dal metodo di eliminazione gaussiana. Consiste nello scambiare il valore dell'elemento pivotale $a_{kk}^{(k)}$ della matrice $A^{(k)}$, al passo k con l'elemento di valore assoluto massimo collocato sotto colonna di elementi $a_{kk}^{(k)} \ldots a_{nk}^{(k)}$ effettuando uno scambio di righe attraverso una matrice di permutazione P^k.

Riprendiamo l'esempio precedente questa volta sfruttando la tecnica del pivoting parziale e vediamo come i risultati cambino :

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1.0001 & 2 \\ 1 & 2 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$$

$$M^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \quad A^{(2)} = M^{(1)}A^{(1)} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 10^{-4} & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

I primi due passaggi rimangono invariati, non abbiamo la necessità di applicare una matrice di permutazione ci troviamo tuttavia da questa trasformazione l'elemento $a_{22}^{(2)}$ avente il valore 10^{-4}. Procediamo in questo modo:

- Dobbiamo creare anzitutto una matrice di permutazione, per farlo partiamo dalla matrice identità e scambiamo l'ordine delle righe che vogliamo invertire pure nella matrice A. In questo caso scambiamo la 2 e la 3.

$$P^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

- Moltiplichiamo ad $A^{(2)}$ la matrice $P^{(2)}$ facendo scambiare le righe.

$$P^{(2)}A^{(2)} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 10^{-4} & 1 \end{pmatrix}$$

Continuiamo quindi con i relativi calcoli fino ad ottenere una matrice triangolare superiore :

$$M^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -10^{-4} & 1 \end{pmatrix} \quad A^{(3)} = M^{(2)}A^{(2)} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & (1 - 10^{-4}) \end{pmatrix}$$

Applichiamo anche a b le stesse matrici applicate ad A.

$$b = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}, \quad b^{(2)} = M^{(1)}b^{(1)} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \quad P^{(2)}b^{(2)} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \quad b^{(3)} = M^{(2)}P^{(2)}b^{(2)} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = y$$

Ancora una volta applichiamo il metodo di sostituzione all'indietro ottenendo come risultato :

$$\tilde{x} = \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix}$$

Il quale, da come si può notare, è molto più preciso rispetto al primo ottenuto in precedenza.
Come differisce dunque il metodo di eliminazione gaussiana tramite pivoting parziale dalla versione vista in precedenza? Semplicemente ad ogni passo vogliamo costruire la matrice $P^{(k)}$ che ci permette di scambiare due righe di A^k in modo tale da immettere nella posizione del pivot l'elemento con valore assoluto maggiore.

Si procederà ovviamente poi a trovare la nostra matrice $U = A^{(n)}$ triangolare superiore ed L triangolare inferiore usando gli stessi altri

passaggi visti precedentemente con la sola differenza che ora per trovare $A^{(k+1)}$ dovremmo moltiplicare anche $P^{(k)}$:

$$A^{(k+1)} = M^{(k)} A^{(k)} P^{(k)}$$

La matrice L trovata avrà il seguente aspetto :

$$L = \begin{pmatrix} 1 & 0 & \dots & 0 & 0 \\ \widetilde{m}_{21}^{(1)} & 1 & \dots & 0 & 0 \\ \widetilde{m}_{31}^{(1)} & \widetilde{m}_{32}^{(2)} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \widetilde{m}_{n1}^{(1)} & \widetilde{m}_{n2}^{(2)} & \dots & \widetilde{m}_{n,n-1}^{(n-1)} & 1 \end{pmatrix}$$

La parte triangolare inferiore di L è ottenuta memorizzando gli elementi del vettore $\widetilde{m}_{ik}^{(k)}$ con $i = k+1, \dots, n$ ottenuti come :

$$\widetilde{m}^{(k)} = P^{(n-1)} P^{(n-2)} \dots P^{(k+1)} P^{(k)} m^{(k)}$$

$$\widetilde{m}^{(n-1)} = m^{(n-1)}$$

Procediamo con un osservazione:
Applicando i passaggi dell'algoritmo otteniamo la matrice $U = A^{(n)}$

$$\begin{aligned} U = A^{(n)} &= M^{(n-1)} P^{(n-1)} A^{(n-1)} \dots M^{(2)} P^{(2)} A^{(2)} M^{(1)} P^{(1)} A^{(1)} \\ &= \widetilde{M}^{(n-1)} \widetilde{M}^{(n-2)} \widetilde{M}^{(n-3)} \dots P^{(n-1)} P^{(n-2)} \dots P^{(1)} A \end{aligned}$$

dove $\widetilde{M}^{(i-1)} = S^{(i)} M^{(i-1)} (S^{(i)})^{-1}$ e

$$S^{(n-1)} = P^{(n-1)}$$

$$S^{(i)} = P^{(n-1)} P^{(n-2)} \dots P^{(i)}$$

Ricordando che $M^{(k)} = I - m^{(k)} e_k^T$ osserviamo quanto segue :

$$\begin{aligned} \widetilde{M}^{(i-1)} &= S^{(i)} M^{(i-1)} (S^{(i)})^{-1} \\ &= S^{(i)} (I - m^{(i-1)} e_{i-1}^T)(S^{(i)})^{-1} \\ &= S^{(i)} (S^{(i)})^{-1} - S^{(i)} m^{(i-1)} e_{i-1}^T (S^{(i)})^{-1} \end{aligned}$$

Sappiamo che :

- $S^{(i)} (S^{(i)})^{-1} = I$
- $S^{(i)} m^{(i-1)} = \widetilde{m}^{(i-1)}$
- $e_{i-1}^T (S^{(i)})^{-1} = e_{i-1}^T$

e quindi $\widetilde{M}^{(i-1)} = I - \widetilde{m}^{(i-1)} e_{i-1}^T$ è anch'essa una matrice elementare di Gauss.

6.1.5 Metodo di Householder

Il metodo di Householder è un'ulteriore metodo diretto con il quale possiamo risolvere un sistema $Ax = b$.

Applicazione a matrici quadrate:

Theorem 1. $\exists Q$ matrice triangolare $n \times n$ $(QQ^T = Q^TQ = I)$ ed una matrice R, triangolare superiore e non singolare $(rg(R) = n)$ tale per cui vale l'eguaglianza :

$$A = QR$$

Quindi anche questo caso studiamo un metodo che, per risolvere un determinato sistema, riduce la matrice iniziale dei coefficienti in una equivalente mediante delle fattorizzazioni le quali avranno i seguenti effetti:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} H_1A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & a_{32} & a_{33} \end{pmatrix} H_2(H_1A) = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{pmatrix}$$

Ad ogni applicazione di H_i vengono azzerati i coefficienti della matrice che risiedono sotto l'i-esimo elemento pivotale. Dimostriamo il teorema appena mostrato definendo :

$$H_{n-1}H_{n-2} \dots H_2H_1A = L$$

Tali matrici di trasformazione sono simmetriche ed ortogonali, da definizione, pertanto vale la seguente catena di eguaglianze :

$$\begin{aligned} A &= (H_{n-1}H_{n-2} \dots H_2H_1)^{-1}L \\ &= H_{n-1}^{-1}H_{n-2}^{-1} \dots H_2^{-1}H_1^{-1}L \\ &= H_1^TH_2^T \dots H_{n-2}^TH_{n-1}^TL \\ &= H_1H_2 \dots H_{n-2}H_{n-1}L \end{aligned}$$

Dove $H_1H_2 \dots H_{n-2}H_{n-1} = Q$
Il costo computazionale di questo metodo è $\frac{2}{3}n^3$ quindi il doppio rispetto al metodo di eliminazione Gaussiana, quale vantaggio ha dunque applicare Householder?
Visto che abbiamo effettuato una fattorizzazione possiamo riscrivere il nostro sistema $Ax = b$ come $QRx = b$. Ponendo $Rx = y$ possiamo ricondurci, come già visto, a risolvere due sistemi lineari:

- $Rx = y$

- $Qy = b$

A questo punto ricordandoci le proprietà della matrice Q possiamo moltiplicare entrambi i membri di $Qy = b$ per la trasposta ottenendo in una maniera più agile il termine y.
Un grande vantaggio che ci offre questo metodo è anche la stabillità che presenta, infatti, è possibile dimostrare che il metodo di Householder è più stabile rispetto all'eliminazione Gaussiana con pivoting parziale e, inoltre, la fattorizzazione QR non è unica. Sia infatti data una matrice diagonale S:

$$S = \begin{pmatrix} \pm 1 & 0 & \dots & 0 & 0 \\ 0 & \pm 1 & \dots & 0 & 0 \\ 0 & 0 & \pm 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & \pm 1 \end{pmatrix}$$

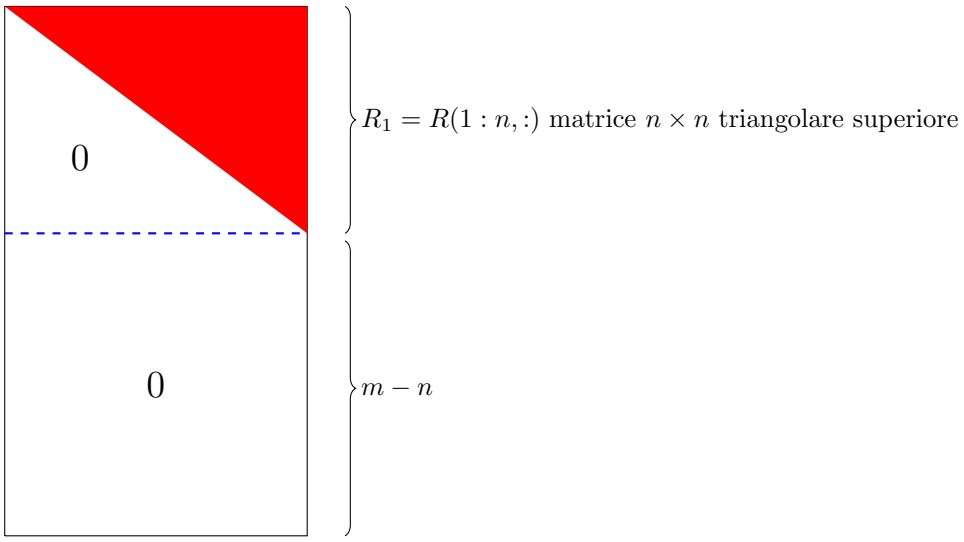
Sappiamo che $SS^T = I$, data quindi l'eguaglianza $A = QR$ possiamo moltiplicare entrambi i membri per I ottenendo che $A = QSS^TR$. Da qui possiamo ottenere quindi due ulteriori matrici $QS = \overline{Q}$ e $S^TR = \overline{R}$ tali per cui vale la relazione $A = \overline{QR}$; si noti inoltre che $\overline{Q}$ rimane una matrice ortogonale e $\overline{S}$ rimane triangolare superiore.

Applicazione a matrici triangolari:

Guardiamo ora il caso dell'applicazione di Householder a matrici triangolari, in particolare vogliamo studiare la soluzione di sistemi con matrici dei coefficienti della forma $m \times n$ con $m \geq n$:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \\ a_{51} & a_{52} & a_{53} \end{pmatrix}$$

Anche in questo caso vogliamo ricondurci ad una fattorizzazione QR in cui Q è sempre una matrice ortogonale $m \times m$ ed R è, invece, "trapezoidale superiore".



Theorem 2. Sia A una matrice $m \times n$ avente $rg(A) = n$ allora $\exists Q$ matrice $m \times m$ ortogonale e R $m \times n$ trapezoidale superiore tali che $A = QR$.

Focalizziamo la nostra attenzione ora sull'applicazione delle matrici di fattorizzazione $H_1, H_2, \ldots, H_{n-2}, H_{n-1}$. Partiamo da una matrice A di dimensione 5×3 :

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \\ a_{51} & a_{52} & a_{53} \end{pmatrix} \quad H_1A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & a_{32} & a_{33} \\ 0 & a_{42} & a_{43} \\ 0 & a_{52} & a_{53} \end{pmatrix} \quad H_2(H_1A) = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \\ 0 & 0 & a_{43} \\ 0 & 0 & a_{53} \end{pmatrix}$$

$$H_3(H_2H_1A) = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

La differenza rispetto al caso quadrato è che qui i passaggi che andiamo a fare non sono più $n-1$ ma bensì n. Il numero di matrici di trasformazione r da applicare sarà dunque $\min(n-1, n)$ in cui $n-1$ è il caso in cui la matrice A è quadrata mentre n quando abbiamo, come appena visto, una matrice A di tipo $m \times n$ con $m > n$.

$$R = H_r H_{r-1} \ldots H_1 A$$

Ricaviamo A:

$$A = H_1 H_2 \dots H_r R$$

Definiamo $Q = H_1 H_2 \dots H_r$. Il costo computazionale di questo metodo è $mn^2 - \frac{n^3}{3}$, si nota che se $m = n$ allora questo è esattamente il costo computazionale del caso quadrato.

6.2 Risoluzione di sistemi sovradeterminati

Quando risolviamo un sistema del tipo :

$$Ax = b \quad A = m \times n \quad x = n \times 1 \quad b = m \times 1$$

con $m > n$ abbiamo che il numero di incognite è superiore al numero di equazioni; questo è quindi un problema mal posto di cui potremmo non avere una soluzione. Per procedere dovremmo dunque riformulare tale problema in un ulteriore maniera, ovvero ricercando x^* tale per cui :

$$x^* = \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \|b - Ax\|_2^2$$

In altri termini :

$$\|b - Ax^*\|_2^2 \leq \|b - Ax\|_2^2 \quad \forall x \in \mathbb{R}^n$$

A questo punto possiamo definire una funzione :

$$\mathbb{Q} : \mathbb{R}^m \mapsto \mathbb{R}$$
$$x \mapsto \mathbb{Q} = \|b - Ax\|_2^2$$

la quale ha matrice Hessiana definita positiva e quindi è strettamente convessa (con l'unico punto di minimo x^*). Cerchiamo di calcolare la nostra x^* adoperando la fattorizzazione QR di A ricordando anzitutto la seguente catena di uguaglianza :

$$\|y\|_2^2 = y^T y = y^T I y = y^T Q Q^T y = (Q^T y)^T (Q^T y) = \|Q^T y\|_2^2$$

con Q matrice ortogonale. Ponendo a questo punto $y = b - Ax^*$:

$$\begin{aligned} \|b - Ax^*\|_2^2 &= \|Q^T b - Q^T A x^*\|_2^2 \\ &= \left\| \begin{pmatrix} \tilde{b_1} \\ \tilde{b_2} \end{pmatrix} - \begin{pmatrix} R_1 \\ 0 \end{pmatrix} \right\|_2^2 \\ &= \left\| \begin{pmatrix} \tilde{b_1} - R_1 \\ \tilde{b_2} \end{pmatrix} \right\|_2^2 \end{aligned}$$

in cui $Q^T A$, come abbiamo già visto, può essere scritto come R. **Nota:**

- $\tilde{b_1}$ rappresenta le prime n componenti del vettore $Q^T b$, $\tilde{b_1}$
- R_1 è la parte superiore del vettore trapezzoidale superiore R e coincide con le sue prime n componenti.

Ricordiamo ora la seguente proprietà legata alla norma 2 di un vettore $\underline{y} \in R^m$:

$$\left\|\underline{y}\right\|_2^2 = \sum_{i=0}^{n} y^2 = \sum_{i=0}^{n-1} y^2 + \sum_{i=n-1}^{m} y^2$$

questo ci permette di risolvere $\text{argmin}\|b - Ax\|_2^2$ nella seguente maniera :

$$\begin{aligned} \text{argmin}\|b - Ax\|_2^2 &= \text{argmin}(\|\tilde{b_1} - R_1\|_2^2 + \|\tilde{b_2}\|_2^2) \\ &= \text{argmin}(\|\tilde{b_1} - R_1\|_2^2) + \|\tilde{b_2}\|_2^2 \end{aligned}$$

Se scelgo allora il vettore x^* tale che risolva $R_1 x = \tilde{b_1}$ ed otteniamo quindi la soluzione:

$$x^* = \text{argmin}\|b - Ax\|_2^2$$

e :

$$\min\|b - Ax\|_2^2 = \|b - Ax^*\|_2^2 = \|\tilde{b_2}\|_2^2.$$

Nota: Se la matrice A è ben condizionata allora lo è anche R_1 in quanto vale la seguente relazione :

$$A^T A = A^T Q Q^T A = (Q^T A)^T Q^T A = \begin{pmatrix} R_1^T & 0 \end{pmatrix} \begin{pmatrix} R_1 \\ 0 \end{pmatrix} = R_1^T R_1$$

Da cui possiamo dire $K_2(A^T A) = K_2(A) = K_2(R_1) = K_2(R_1^T R_1)$