

Calcolo Numerico Appunti

Alex Bastianini

Contents

Chapter 1

I numeri finiti

Page

- 1.1 Base-N and Binary
Addizione e moltiplicazione in binario — • Conversione fra basi — • Floating point —
- 1.2 Errori di rappresentazione

Chapter 2

Risoluzione di equazioni

Page

- 2.1 Calcolo dello zero di una funzione
Algoritmo di bisezione —
- 2.2 Iterazione di punto fisso
Mappe contrattive — • Velocita' del miglioramento dell'approssimazione — • Metodo di Newton —
- 2.3 Risolvere sistemi lineari
Fattorizzazione LU — • Fattorizzazione di Cholesky —
- 2.4 Approssimazione
SVD —

Chapter 1

I numeri finiti

1.1 Base-N and Binary

$$147.3 = 1 \cdot 10^2 + 4 \cdot 10^1 + 7 \cdot 10^0 + 3 \cdot 10^{-1}.$$

Since each digit is associated with a power of 10, the decimal system is also known as **base10** because it is based on 10 digits (0 to 9). However, there is nothing special about base10 numbers except perhaps that you are more accustomed to using them. For example, in base3 we have the digits 0, 1, and 2 and the number $121(\text{base } 3) = 1 \cdot 3^2 + 2 \cdot 3^1 + 1 \cdot 3^0 = 9 + 6 + 1 = 16(\text{base } 10)$

1.1.1 Addizione e moltiplicazione in binario

1.1.2 Conversione fra basi

1.1.3 Floating point

Data la finitezza della memoria di un calcolatore, e' impossibile rappresentare esattamente un numero che ha precisione infinita. Per riuscire a rappresentare in modo piu' efficiente possibile si usano i **floating point** numbers:

Definition 1.1.1: Floating Point

Alloca:

- Un indicatore per il segno (s)
- L'esponente (e)
- La mantissa (f)

Nello standard **IEEE754** double precision, che occupa 64 bit, un numero e' rappresentato nel seguente modo:

$$n = (-1)^s 2^{e-1023} (1 + f). \text{ (for 64-bit)}$$

In generale, ogni numero reale $n \in \mathbb{R}$ e' definito da:

$$n = \pm \beta^p \cdot d_0.d_1 \dots d_t$$

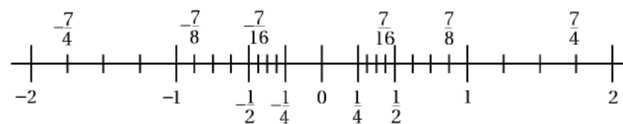
Quindi l'insieme dei numeri floating point caratterizzati dai valori $\beta, t, +L, U$ sono:

$$\mathbb{F}(\beta, t, L, U) = \left\{ \phi \cup m = \pm \beta^p \cdot d_0 \cdot d_1 \cdot \dots \cdot d_t \mid -L \leq p \leq U \right\}$$

Se un numero $x \in \mathbb{R}$ ha un numero di cifre decimali maggiore di t , allora $x \notin \mathcal{F}$ e deve essere approssimato (solitamente col troncamento).

1.2 Errori di rappresentazione

Dato che stiamo usando lo stesso numero di bit, il numero di valori rappresentabili rimane uguale, ma la codifica *floating point* fa in modo che il **gap**, ovvero la differenza, fra due numeri successivi (che esistono sempre dato che lavoriamo con valori discreti) sia relativo al valore assoluto dei numeri rappresentati. Questo è utile dato che riusciamo ad essere molto precisi con numeri piccoli, riuscendo comunque a rappresentare valori enormi con un errore che in confronto rimane minuscolo.



Abbiamo due modi per indicare l'accuratezza con la quale possiamo rappresentare un valore $x \in \mathbb{R}$:

- L'errore **assoluto**, che è più grande più è grande l'ordine di x

$$|fl(x) - x| < \beta^{p-t}$$

- E quello **relativo**, che dipende solo da t

$$\frac{|fl(x) - x|}{|x|} < \frac{1}{2} \beta^{1-t} = \text{eps}$$

Il valore **eps** è detto **precisione macchina**, ed è il numero macchina più piccolo positivo tale che

$$fl(1 + \text{eps}) > 1$$

Quindi, definite due operazioni generiche, una per i reali e l'altra per i floating-point:

- $\cdot : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$
- $\circ : \mathbb{F} \times \mathbb{F} \rightarrow \mathbb{F}$

$$x \circ y = fl(x \cdot y)$$

Abbiamo che ogni operazione fra floating-point genera un errore (di **arrotondamento**):

$$\left| \frac{(x \circ y) - (x \cdot y)}{x \cdot y} \right| < \text{eps}$$

Nel caso di somma di numeri di grandezza molto diversa, le cifre più significative del numero più piccolo possono essere perse. Per una sola operazione questo non causa un errore troppo grande, ma se l'operazione viene eseguita molte volte l'errore si **amplifica** e può diventare catastrofico.

Chapter 2

Risoluzione di equazioni

2.1 Calcolo dello zero di una funzione

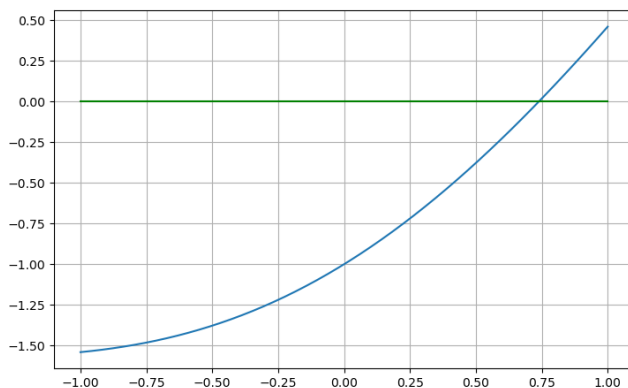
Come primo metodo per risolvere (approssimativamente) equazioni, usiamo il metodo della **bisezione**. Questo sfrutta il teorema degli zeri per trovare uno dei possibili zeri di una funzione continua su un intervallo chiuso. Come esempio, prendiamo questa equazione:

$$x = \cos x$$

Che possiamo riscrivere come:

$$(f(x) = x - \cos x), x \in \mathbb{R}. f(x) = 0$$

Dato l'intervallo $[-1, 1]$, otteniamo il seguente grafico:



2.1.1 Algoritmo di bisezione

L'algoritmo di bisezione usa un approccio simile alla ricerca binaria per avvicinarsi sempre di più al valore dello zero. Dato in input la funzione continua $f : \mathbb{R} \rightarrow \mathbb{R}$ e un intervallo $[a, b]$, $f(a)f(b) < 0$, viene calcolato il valore della funzione nel punto di mezzo c . L'intervallo viene poi ristretto, prendendo c come uno dei limiti e scegliendo l'altro fra a e b in modo che i valori della funzione agli estremi abbiano sempre segno opposto. In pseudocodice:

Algorithm 1: Bisezione semplice

Input: $f : \mathbb{R} \rightarrow \mathbb{R}$ continua su $[a, b] \subset \mathbb{R}$, tale che $f(a)f(b) < 0$. $N > 0$ (numero di iterazioni)

Output: Valore approssimato di uno zero di f su $[a, b]$

```
1  $c \leftarrow 0$ ;  
2 for  $N$  times do  
3    $c \leftarrow \frac{a+b}{2}$ ;  
4   if  $f(a)f(c) < 0$  then  
5      $b \leftarrow c$   
6   else  
7      $a \leftarrow c$   
8   end  
9 end  
10 return  $c$ 
```

Se vogliamo assicurarci che il risultato r' che otteniamo abbia al massimo un errore E_{tol} rispetto al valore reale r della radice, basta porre una condizione d'arresto nel ciclo di bisezione. Sapendo che ad ogni ciclo lo zero si trova fra a e b , scegliendo r' come punto di mezzo rende l'errore massimo $E_{max} = \frac{b-a}{2}$, quindi:

Algorithm 2: Bisezione con errore

Input: $f : \mathbb{R} \rightarrow \mathbb{R}$ continua su $[a, b] \subset \mathbb{R}$, tale che $f(a)f(b) < 0$. E_{tol}

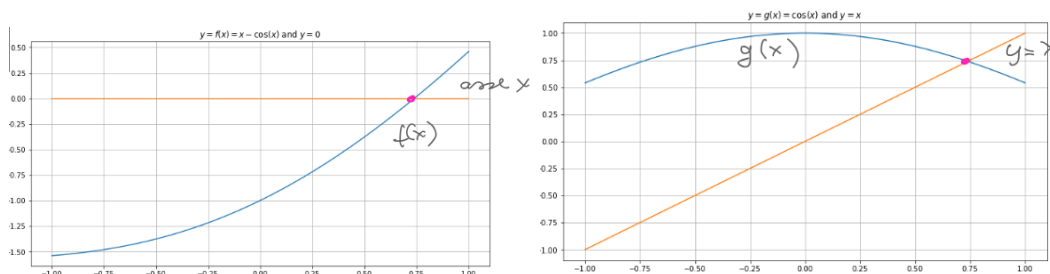
Output: Valore approssimato di uno zero di f su $[a, b]$ con errore assoluto al massimo E_{tol}

```
1  $c \leftarrow 0$ ;  
2  $E_{max} = \frac{b-a}{2}$ ;  
3 while  $E_{max} > E_{tol}$  do  
4    $c \leftarrow \frac{a+b}{2}$ ;  
5   if  $f(a)f(c) < 0$  then  
6      $b \leftarrow c$   
7   else  
8      $a \leftarrow c$   
9   end  
10   $E_{max} = \frac{b-a}{2}$   
11 end  
12 return  $c$ 
```

2.2 Iterazione di punto fisso

Preso la funzione f di cui vogliamo trovare lo zero, possiamo definire una funzione $g(x) = x - w(x)f(x)$ dove w e' una funzione peso *positiva e limitata*. Quindi

$$f(x) = 0 \iff g(x) = x$$



Quindi siamo passati dal cercare uno zero di una funzione a trovare il **punto fisso**:

Definition 2.2.1: Punto fisso

Data $g : D \rightarrow C$, si chiamano **punti fissi** tutti i punti $x \in D$:

$$g(x) = x$$

Per trovare il punto fisso, bisogna quasi sempre usare un metodo *iterativo*, ovvero di approssimazioni successive, seguendo questa procedura:

- Iniziare con una prima approssimazione x_0
- Iterare con la seguente formula: $x_{k+1} = g(x_k)$

Proposition 2.2.1

Data una funzione $g : D \rightarrow C$ continua e una successione $x_n \in D$. $x_{k+1} = g(x_k)$ tale che $x_n \xrightarrow{n \rightarrow +\infty} p$, allora:

$$g(p) = p$$

Ovvero p e' un **punto fisso** di g .

Dimostrazione: Per proprieta' delle successioni e per ipotesi, sappiamo che $\lim_{k \rightarrow +\infty} x_k = p$. Quindi, per continuita' di g :

$$\lim_{k \rightarrow +\infty} g(x_k) = g(p)$$

Inoltre, per costruzione della successione x_n , sappiamo che $g(x_k) = x_{k+1}$, quindi:

$$\lim_{k \rightarrow +\infty} x_{k+1} = \lim_{k \rightarrow +\infty} g(x_k)$$

$$p = g(p)$$

☺

2.2.1 Mappe contrattive

Ma come facciamo ad assicurarci che la successione converga? Ecco che entrano in gioco le **mappe contrattive**.

Definition 2.2.2: Mappa

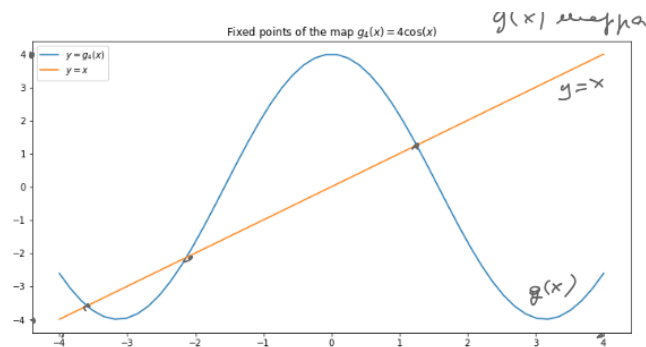
Una **mappa** e' una funzione $g : D \rightarrow D$ dove D e' un intervallo chiuso $[a, b]$.

Proposition 2.2.2

Presa una mappa continua su un intervallo $[a, b]$, allora:

$$\exists p \in [a, b]. p \text{ e' un punto fisso}$$

Dimostrazione: Consideriamo la funzione $f(x) = x - g(x)$, dove g e' una mappa continua sull'intervallo $[a, b]$. Quindi $f(a) = a - g(a) \leq 0$ e $f(b) = b - g(b) \geq 0$, dato che per definizione di mappa $g(a), g(b) \in [a, b]$. Quindi per teorema degli zeri $\exists p \in [a, b]. f(p) = 0$. Quindi $g(p) = p$ e' un punto fisso. ☺



Come si vede dal grafico della mappa continua $g(x)$ su $[-4, 4]$, e' possibile che una mappa abbia piu' di un punto fisso. Per assicurarci l'esistenza di una soluzione unica, dobbiamo aggiungere un'ultima condizione:

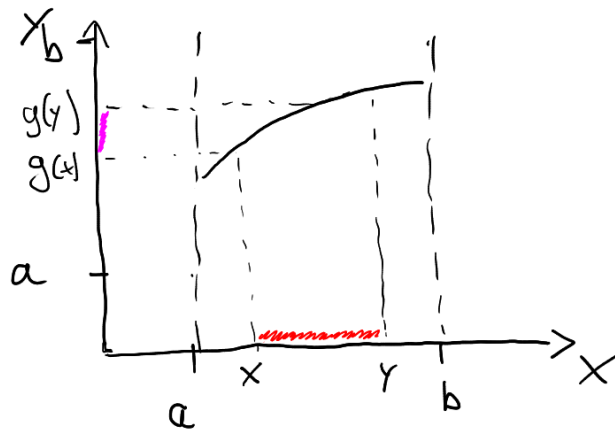
Definition 2.2.3: Mappa contrattiva

Una mappa $g : D \rightarrow D$ si dice **contrattiva** se esiste $C < 1$ tale che:

$$\forall x, y \in D. |g(x) - g(y)| \leq C|x - y|$$

C e' detta costante contrattiva

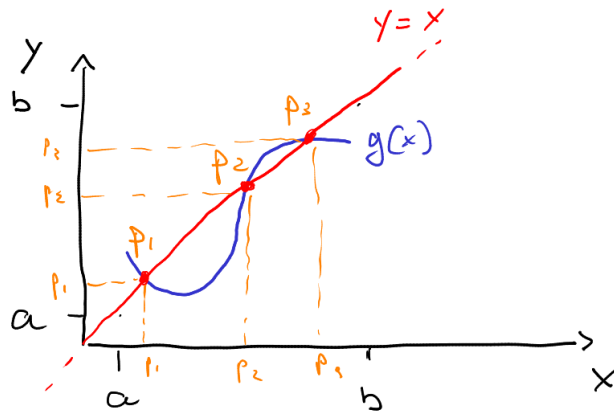
Questo significa che per ogni due punti che prendiamo dal dominio, la loro distanza sara' sempre maggiore (o uguale) alla distanza fra i due punti corrispondenti nel codominio. Graficamente (in \mathbb{R}^2):



Theorem 2.2.1

Data una mappa contrattiva g su un intervallo chiuso D , esiste un solo punto fisso p a cui converge la successione x_n dove $x_{k+1} = g(x_k)$, scegliendo x_0 come qualunque punto in D .

Dimostrazione (singolo punto fisso): Assumiamo che g abbia piu' di un punto fisso e dimostriamo l'assurdo. Consideriamo il seguente grafico:



Prendendo due dei punti fissi, p_1, p_2 , notiamo che $\frac{|g(p_1) - g(p_2)|}{|p_1 - p_2|} = \frac{|p_1 - p_2|}{|p_1 - p_2|} = 1$, ma dato che la costante C deve essere minore di 1, abbiamo dimostrato l'assurdo. ☹

La seconda parte della dimostrazione la vediamo più avanti.

Come controllare se una funzione differenziabile e' una mappa contrattiva

Nella dimostrazione precedente e' apparsa una forma che somigliava molto a una derivata. Infatti, se la funzione g e' derivabile possiamo usare il seguente teorema:

Theorem 2.2.2

Se una funzione $g : [a, b] \rightarrow [a, b]$ e' differenziabile e esiste una costante $C < 1$ tale che $\forall x \in [a, b]. |g'(x)| < C$, allora g e' una **mappa contrattiva**

Dimostrazione: Per il teorema di Lagrange, $\forall x < y \in [a, b]. \exists c \in [x, y]. g(y) - g(x) = g'(c)(y - x)$. Quindi, aggiungendo valori assoluti, $|g(y) - g(x)| = |g'(c)| \cdot |y - x| \leq C \cdot |y - x|$ per ipotesi. Quindi, per definizione, g e' una mappa contrattiva. ☹

2.2.2 Velocita' del miglioramento dell'approssimazione

Chiamiamo l'errore assoluto $|E_k|$ la differenza assoluta fra x_k e il valore effettivo del punto fisso p .

Proposition 2.2.3

$|E_{k+1}| \leq C|E_k|$, quindi

$$|E_k| = C^k |E_0|$$

Dove $E_0 = x_0 - p$.

Quindi per ogni iterazione, l'errore iniziale diminuisce almeno di un fattore C (che ricordo e' sempre < 1 quindi e' una cosa buona). Cio' implica che piu' e' piccolo C , piu' ci avviciniamo in fretta a p .

Dimostrazione: Per definizione di errore $E_{k+1} = x_{k+1} - p$, che possiamo riscrivere come $g(x_k) - g(p)$, quindi per definizione di mappa contrattiva $|E_{k+1}| = |g(x_k) - g(p)| \leq C|x_k - p| = C|E_k|$. Se sostituiamo $k = 0$ abbiamo il caso base:

$$|E_1| = C|x_0 - p|$$

Da cui ricorsivamente:

$$|E_k| = C \cdot C \cdot \dots \cdot C|x_0 - p| = C^k|x_0 - p|$$

☹

Ora siamo in grado di finire la dimostrazione di prima:

Diostrazione (convergenza): Usando la proposizione precedente, $\lim_{k \rightarrow +\infty} |E_k| = \lim_{k \rightarrow +\infty} C^k |x_0 - p| = 0$. Quindi:

$$x_k \xrightarrow{k \rightarrow +\infty} p$$

☺

2.2.3 Metodo di Newton

Vediamo ora il metodo di Newton, che rispetto alla bisezione e' generalmente piu' veloce (ma non abbiamo la garanzia). Puo' essere dimostrato come una specifica mappa contrattiva o come approssimazione a linee tangenti, vediamo la prima:

Prendiamo una funzione differenziabile f e costruiamole una mappa contrattiva g . Per migliorare l'efficienza, vogliamo fare in modo che la costante C abbia valore minimo dato un intorno della radice r di f , come facciamo? Dato che C assume il valore della derivata massima della mappa, vogliamo fare in modo che la derivata di g sia minima per quell'intorno. Un modo per fare cio' e' assicurarci che $g'(r) = 0$, vediamo come:

$$g'(r) = 1 - w'(r)f(r) - w(r)f'(r) = 1 - w(r)f'(r)$$

dato che $f(r) = 0$ per definizione, quindi:

$$w(r) = \frac{1}{f'(r)}$$

Dato che non sappiamo il valore di r , poniamo $w(x) = \frac{1}{f'(x)}$ per ogni x nel dominio. Da notare che $w(x)$ non esiste quando $f'(x) = 0$. Facendo cosi', la formula di iterazione diventa:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

che e' la formula per il metodo di Newton.

2.3 Risolvere sistemi lineari

- Metodi Diretti: +accurati, -efficienti
- Metodi Iterativi: -accurati, +efficienti

2.3.1 Fattorizzazione LU

Per risolvere sistemi lineari, il metodo diretto piu' efficiente risulta essere la **fattorizzazione LU**. Purche' abbia un numero maggiore di operazioni floating-point (flops) rispetto al Gaussian-Jordan, permette di calcolare multiple soluzioni di un sistema $Ax = b$ decomponendo una sola volta la matrice A .

Algoritmo

Dobbiamo fattorizzare la matrice A e ottenere due matrici triangolari, una superiore (U) e l'altra inferiore (L). In piu', L deve avere tutti 1 sulla diagonale principale.

$$A = LU = \begin{pmatrix} 1 & 0 & 0 & 0 \\ * & 1 & 0 & 0 \\ * & * & 1 & 0 \\ * & * & * & 1 \end{pmatrix} \cdot \begin{pmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{pmatrix}$$

In modo da essere fattorizzabile, A non deve essere *singolare* ($\det A \neq 0$) e deve avere i minori principali diversi da 0. Per fare cio', moltiplichiamo la matrice A per $n-1$ matrici E_1, E_2, \dots, E_{n-1} (che per costruzione sono tutte lower) in modo che A diventi triangolare superiore (sara' il valore di U), mentre L sara' l'inversa del prodotto $E_{n-1} \cdot \dots \cdot E_2 \cdot E_1$ (dato che $L^{-1}A = U$). Quindi per prop dell'inversa $L = E_1^{-1} \cdot E_2^{-1} \cdot \dots \cdot E_{n-1}^{-1}$:

$$A = \begin{bmatrix} 9 & 3 & 6 \\ 3 & 4 & 6 \\ 0 & 8 & 8 \end{bmatrix} \quad l_{12} = -\frac{a_{12}}{a_{11}}$$

step 1 $l_{21} = -\frac{a_{21}}{a_{11}}$
 $l_{31} = -\frac{a_{31}}{a_{11}}$

$$\begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{3} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 9 & 3 & 6 \\ 3 & 4 & 6 \\ 0 & 8 & 8 \end{bmatrix} = \begin{bmatrix} 9 & 3 & 6 \\ 0 & 3 & 4 \\ 0 & 8 & 8 \end{bmatrix}$$

E_1 triangolare inf, elementi sulla diagonale, solo la colonna 1 diversa da 0
 step 2 $l_{22} = -\frac{a_{22}}{a_{22}}$
 $l_{32} = -\frac{a_{32}}{a_{22}}$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{8}{3} & 1 \end{bmatrix} \begin{bmatrix} 9 & 3 & 6 \\ 0 & 3 & 4 \\ 0 & 8 & 8 \end{bmatrix} = \begin{bmatrix} 9 & 3 & 6 \\ 0 & 3 & 4 \\ 0 & 0 & -\frac{8}{3} \end{bmatrix}$$

E_2 triangolare inf, elementi sulla diagonale, solo la colonna 2 diversa da 0
 step 3 $l_{33} = -\frac{a_{33}}{a_{33}}$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 9 & 3 & 6 \\ 0 & 3 & 4 \\ 0 & 0 & -\frac{8}{3} \end{bmatrix} = \begin{bmatrix} 9 & 3 & 6 \\ 0 & 3 & 4 \\ 0 & 0 & -\frac{8}{3} \end{bmatrix}$$

E_3 triangolare inf, elementi sulla diagonale, solo la colonna 3 diversa da 0
 step 4 $l_{33} = -\frac{a_{33}}{a_{33}}$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 9 & 3 & 6 \\ 0 & 3 & 4 \\ 0 & 0 & -\frac{8}{3} \end{bmatrix} = \begin{bmatrix} 9 & 3 & 6 \\ 0 & 3 & 4 \\ 0 & 0 & -\frac{8}{3} \end{bmatrix}$$

E_4 triangolare inf, elementi sulla diagonale, solo la colonna 4 diversa da 0

$$\rightarrow L^{-1} = E_2 E_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{8}{3} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{3} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{3} & 1 & 0 \\ \frac{8}{9} & -\frac{8}{3} & 1 \end{bmatrix}$$

Dato che E_1, E_2, \dots, E_{n-1} sono triangolari con solo 1 sulla diagonale, per trovare l'inversa basta cambiare di segno i loro valori (tranne gli 1 diagonali). Per evitare di dividere per numeri troppo piccoli durante la costruzione delle matrici lower, e' possibile usare una matrice P di cambio di righe:

$$PA = LU \implies A = P^{-1}LU \implies A = PLU$$

($P = P^{-1} = P^T$). Infine ci resta risolvere il sistema:

$$Ax = b \implies LUX = b \implies \begin{cases} UX = y \\ LY = b \end{cases}$$

Per cui usiamo il metodo di sostituzione avanti, dato che sono sistemi triangolari:

SISTEMA TRIANGOLARE INFERIORE

$$\begin{cases} l_{11} x_1 = b_1 \\ l_{21} x_1 + l_{22} x_2 = b_2 \\ \vdots \\ l_{m1} x_1 + l_{m2} x_2 + \dots + l_{mn} x_n = b_n \end{cases}$$

$$Lx = b \quad L = \begin{pmatrix} l_{11} & & \\ l_{21} & l_{22} & \\ & \ddots & \ddots \\ & & l_{nn} \end{pmatrix}$$

metodo di sostituzione avanti

$$\begin{aligned} x_1 &= b_1 / l_{11} \\ x_2 &= (b_2 - l_{21} x_1) / l_{22} \\ &\vdots \\ x_n &= \dots \end{aligned}$$

2.3.2 Fattorizzazione di Cholesky

Si puo' usare solo nel caso speciale in cui A sia definita semi-positiva:

Definition 2.3.1: Matrice semi-definita positiva

A $n \times n$ simmetrica e' **semi-definita positiva** se $\forall x \in \mathbb{R}^n$:

$$x^T A x \geq 0$$

Per cui vale:

Proposition 2.3.1 Autovalori di matrici semi-definite positive

Data una matrice A semi-definita positiva, ogni autovalore di A e' positivo o nullo

Grazie a queste proprieta', possiamo dire che $\exists L$:

$$A = L \cdot L^T$$

e trovarlo e' circa due volte piu' efficiente rispetto a LU.

2.4 Approssimazione

2.4.1 SVD

Prima di iniziare ad introdurre metodi di approssimazione, e' necessario esplorare uno dei teoremi finali dell'algebra lineare: la **SVD** (Single Value Decomposition). Questo teorema sara' fondamentale per descrivere vari modelli di ottimizzazione e approssimazione per motivi che vedremo dopo.

Theorem 2.4.1 Single Value Decomposition

Data una qualunque matrice reale A $m \times n$ (per comodita' stabiliamo $m \geq n$) di rango k , allora esistono:

- U ortonormale $m \times m$
- V ortonormale $n \times n$
- Σ rettangolare diagonale $m \times n$

Tali che:

$$A = U \Sigma V^T$$

Quindi **qualunque** matrice puo' essere espressa come una matrice della stessa dimensione diagonale, moltiplicata per due matrici ortonormali (di cui una trasposta). Proviamo a vedere perche':

:

