

Corso di Calcolo Numerico

Corso di Introduzione al Calcolo Scientifico

Silvia Bonettini



UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

Dipartimento di Scienze Fisiche,
Informatiche e Matematiche
Università di Modena e Reggio Emilia



Optimization Algorithms and Software
for Inverse problems
www.oasis.unimore.it

Corso di Laurea in Informatica
Corso di Laurea in Matematica

a.a. 2019/2020

Introduzione

Sommario della lezione:

- riferimenti web
- struttura del corso
- modalità d'esame per il Corso di Laurea in Informatica
- modalità d'esame per il Corso di Laurea in Matematica

- silvia.bonettini@unimore.it
- Ricevimento su appuntamento
- Materiale didattico disponibile su Dolly
- Orario e aule su
[http://www.orariolezioni.unimore.it/Orario/
Dipartimento_di_Scienze_Fisiche-_Informatiche_
e_Matematiche/2019-2020/1384/index.html](http://www.orariolezioni.unimore.it/Orario/Dipartimento_di_Scienze_Fisiche-_Informatiche_e_Matematiche/2019-2020/1384/index.html)

Struttura del corso

Le lezioni del corso sono strutturate in

- una parte teorica in aula, ($\frac{2}{3}$ del corso)
- una parte applicativa in laboratorio, ($\frac{1}{3}$ del corso)

$$\begin{array}{rcl} 72 \text{ ore} & = & 48 \text{ ore in aula} & + & 24 \text{ ore in laboratorio} \\ 9 \text{ CFU} & = & 6 \text{ CFU} & + & 3 \text{ CFU} \end{array}$$

- Nelle lezioni in laboratorio verrà utilizzato l'ambiente di programmazione MATLAB e il relativo linguaggio.
- Link alla pagina <https://www.unimore.it/servizistudenti/licenzamatlab.html> per scaricare gratuitamente il software sul proprio PC con la licenza di Ateneo.
- Per il Corso di Laurea in Matematica, le ore di lezione in laboratorio corrispondono ai 3 CFU di "Introduzione al Calcolo Scientifico".

L'esame consiste in una **prova orale** in cui verranno poste 3 domande:

- **2 domande** sugli argomenti svolti nelle ore di lezione in aula
- **1 domanda** in cui si chiede l'implementazione di un semplice algoritmo in linguaggio **MATLAB**

Valutazione:

- L'esame viene valutato con un unico voto complessivo che tiene conto delle risposte ai tre quesiti proposti.

Il corso partecipa alla sperimentazione di Ateneo relativa alla didattica in modalità *Team Based Learning (TBL)*

- 4 esercitazioni in modalità TBL nel corso del semestre, durante l'orario di lezione.
- la partecipazione alle attività TBL è facoltativa
- **Incentivi e bonus** per i partecipanti al TBL: in base alla valutazione media ottenuta nelle 3 prove migliori, è possibile ottenere **da 1 a 3 punti aggiuntivi** sul voto finale dell'esame e, per i punteggi nella fascia più alta, anche **l'esenzione dalla domanda di Matlab della prova orale.**

Per ulteriori informazioni:

Vedi sezione TBL sulla pagina Dolly del corso.

L'esame relativo ai 6 CFU di "Calcolo Numerico" consiste in una **prova orale**

- **2 domande** sugli argomenti svolti nelle ore di lezione in aula

L'esame relativo ai 3 CFU di "Introduzione al Calcolo Scientifico" consiste in una **prova scritta**

- implementazione di un semplice algoritmo in linguaggio **MATLAB**

Valutazione:

- La valutazione in trentesimi della prova orale costituisce il voto corrispondente ai crediti di "Calcolo Numerico".
- La valutazione positiva della prova scritta costituisce l'idoneità corrispondente ai crediti di "Introduzione al Calcolo Scientifico".

- Verranno pubblicate su ESSE3 (una decina di appelli in totale nelle tre sessioni annuali)
- Non è possibile sostenere l'esame in date diverse da quelle pubblicate
- Possibilità di appelli fuori sessione solo per gli studenti fuori corso

Regola da rispettare

Chi non supera la prova orale oppure si iscrive ad un appello e non si presenta il giorno dell'esame senza essersi cancellato per tempo dalla lista **NON** può iscriversi all'appello immediatamente successivo della stessa sessione.

Sommario della lezione:

- Definizione della materia “Analisi Numerica” di cui questo corso fornisce le basi
- Quattro elementi fondamentali che tratteremo durante tutto il corso

Di cosa si occupa l'Analisi Numerica?

- "trovare gli algoritmi che risolvono un problema matematico nel minor tempo e con la massima accuratezza"
- "dare una risposta numerica ad un problema matematico mediante un calcolatore"

(V. Comincioli, *Analisi Numerica, Metodi, Modelli, Applicazioni*)

Quattro elementi chiave:

- ① Problemi matematici
- ② Algoritmi
- ③ Calcolatore
- ④ Tempo e accuratezza

Problemi matematici

Esempio: un problema semplice

Problema matematico: data la funzione $f(x) = \sin(x^2)$, calcolarne la derivata.

Risposta (soluzione) analitica

Applicando i teoremi e le definizioni dell'analisi matematica, si vuole ricavare l'espressione analitica della funzione derivata:

$$f'(x) = 2x \cos(x^2)$$

Il dato del problema analitico è una funzione, $f(x)$, il risultato è una funzione, $f'(x)$.

Risposta (soluzione) numerica

Applicando i teoremi e i principi dell'analisi numerica, si vuole ricavare un algoritmo che, dato un numero $x \in \mathbb{R}$, calcola un numero y che approssima bene la derivata prima di f in x

$$y \simeq f'(x)$$

Il dato del problema *numerico* è un numero reale, x , il risultato è un numero reale, y

Definizione

Un algoritmo è una procedura che, a partire da un insieme di dati in ingresso, produce un insieme di risultati in uscita mediante un numero finito di passi definiti in modo univoco.

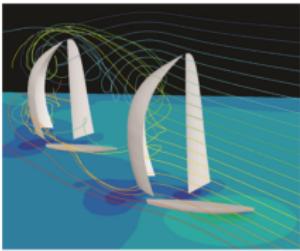
Negli algoritmi *numerici*,

- i dati in input e le soluzioni in output sono numeri reali.
- i passi sono costituiti da successioni delle quattro operazioni aritmetiche fondamentali, per poter essere eseguiti da un calcolatore.

Con algoritmi numerici avanzati si possono risolvere tanti problemi...matematici!

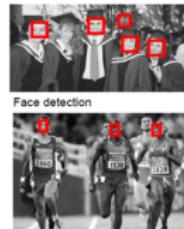


riconoscere cifre
scritte a mano



simulare l'interazione aerodinamica tra due barche da regata in navigazione

sequenze genomiche



classificare
immagini, testi, ...



3. Cerca con Google o digita un URL.

progettare un motore di ricerca

- Prenderemo in esame alcuni problemi matematici fondamentali
- Studieremo gli algoritmi numerici opportuni per risolverli, analizzando le loro proprietà teoriche
- Realizzeremo un'implementazione degli algoritmi in ambiente Matlab

Tempo e accuratezza

Due limitazioni fondamentali

Limiti di tempo

- Un algoritmo efficace deve produrre le soluzioni in un tempo non solo finito, ma “ragionevole”.
- Il tempo di calcolo è direttamente proporzionale al numero di operazioni aritmetiche elementari eseguite.



Complessità computazionale

Limiti di spazio

- Si dispone di uno spazio di memoria finita
 - Non tutti i numeri possono essere rappresentati al calcolatore
 - Il risultato delle operazioni è diverso da quello usuale



Condizionamento dei problemi e Stabilità degli algoritmi

I limiti spaziali impongono l'analisi dei problemi e degli algoritmi numerici da un punto di vista molto speciale.

Analisi del condizionamento dei problemi

I problemi numerici verranno valutati tenendo conto del fatto che, per le limitazioni dello spazio di memoria, i dati in ingresso potrebbero non essere rappresentati esattamente dal calcolatore.

Analisi della stabilità degli algoritmi

Gli algoritmi numerici verranno valutati tenendo conto del fatto che dovranno essere eseguiti dal calcolatore che utilizza un'aritmetica diversa da quella usuale.

Esempio

```
#include <iostream>
using namespace std;
int main()
{
    int i;
    double u;

    u = 1;
    i = 0;
    while (1 + u > 1)
    {
        u = u/2;
        i = i + 1;
    };
    printf("i=%i\n",i);
}
```

Risultato:

Compilatore C online: <http://www.italiancpp.org/compiler/>

L'output del programma è la stampa

$$i = 53$$

Significa che per il calcolatore è vera l'uguaglianza

$$1 + \frac{1}{2^{53}} = 1$$



Nella prima parte del corso vedremo come questo sia una diretta conseguenza di come il calcolatore *rappresenta* i numeri reali e come esegue le operazioni aritmetiche.

- ① Rappresentazione dei numeri al calcolatore
 - Effetti dell'aritmetica di macchina
- ② Algebra lineare numerica: algoritmi per la soluzione di sistemi lineari
 - Metodi diretti (fattorizzazioni)
 - Metodi iterativi
- ③ Equazioni non lineari
- ④ Problemi di punto fisso
- ⑤ Interpolazione
 - Polinomio di interpolazione
 - Splines
- ⑥ Approssimazione con il criterio dei Minimi quadrati

Suggerimento per uno studio efficace...

Per ogni problema considerato:

- Quali sono i dati? Cosa si vuole calcolare?
- Discussione delle proprietà della/delle soluzioni:
 - ① Esistenza e unicità, ipotesi...
 - ② Condizionamento
- Presentazione degli algoritmi, delle loro proprietà
 - ① Complessità computazionale
 - ② Stabilità
 - ③ Confronto tra algoritmi
- Implementazione in Matlab

La rappresentazione dei numeri

Sommario della lezione:

- Richiami sulla notazione posizionale
- Rappresentazione di interi e reali in basi non decimali
 - Algoritmo delle divisioni successive
 - Algoritmo delle moltiplicazioni successive

La notazione posizionale

distinguiamo il concetto di quantità espressa da un numero e la sua rappresentazione.

NUMERO

ENTITA' ASTRATTA

quantità "sette"

univocamente determinata

RAPPRESENTAZIONE

(7, VII, $(111)_2$, ...)

molteplice a seconda dei criteri di rappresentazione adottati.

La convenzione da noi usata per esprimere un numero è detta *notazione posizionale*, in cui ogni cifra assume un valore diverso a seconda della sua posizione all'interno della stringa usata per rappresentarlo.

$$\begin{array}{lllll} \text{Cifre} & 7 & 5 & 9 & = \\ \text{Posizione} & 2 & 1 & 0 & \\ \text{Base} & 10 & & & \end{array} \quad 7 \cdot 10^2 + 5 \cdot 10^1 + 9 \cdot 10^0$$

- La notazione posizionale decimale è caratterizzata da una *base* $\beta = 10$, dalle posizioni delle cifre che costituiscono gli *esponenti* della base e da un insieme di *simboli*, le cifre 0,1,2,3,... , ciascuna delle quali indica un valore compreso tra 0 e 9.
- Mantenendo la convenzione posizionale, possiamo estendere il discorso per qualsiasi base $\beta > 1$, introducendo un insieme di simboli

BASE SIMBOLI

$$2 \quad \mathcal{S} = \{0, 1\}$$

$$8 \quad \mathcal{S} = \{0, 1, 2, 3, 4, 5, 6, 7\}$$

$$16 \quad \mathcal{S} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$$

- L'insieme dei simboli contiene le cifre corrispondenti ai valori $0, 1, \dots, \beta - 1$.

Rappresentazione posizionale dei naturali in base β

- Data una base β e il corrispondente insieme di simboli S , ogni numero naturale N si può rappresentare in modo univoco con una stringa di cifre

$$(c_p c_{p-1} \dots c_1 c_0)_{\beta}$$

dove $p \in \mathbb{N}$.

- Se, con un piccolo abuso di notazione, indichiamo con c_i anche il valore della cifra e identifichiamo il numero con la sua rappresentazione (univoca), allora il valore del numero è

$$N = (c_p c_{p-1} \dots c_1 c_0)_{\beta} = c_p \beta^p + c_{p-1} \beta^{p-1} + \dots + c_1 \beta + c_0$$

Cifre significative della rappresentazione

$$N = (c_p c_{p-1} \cdots c_1 c_0)_\beta = c_p \beta^p + c_{p-1} \beta^{p-1} + \cdots + c_1 \beta + c_0$$

Supponiamo $c_p \neq 0$.

Dato $t \leq p$,

- le t cifre **meno significative** della rappresentazione in base β del numero $N \in \mathbb{N}$ sono i pesi delle potenze più piccole della base:

$$(c_p c_{p-1} \cdots c_t \underbrace{c_{t-1} c_1 c_0}_{})_\beta$$

Si contano da destra;

- le t cifre **più significative** nella rappresentazione in base β del numero $N \in \mathbb{N}$ sono i pesi delle potenze più grandi della base:

$$(\underbrace{c_p c_{p-1} \cdots c_{p-t+1}}_{\cdots} \cdots c_1 c_0)_\beta$$

Si contano da sinistra partendo dalla prima cifra non nulla della rappresentazione.

Esempi

"trecentosettantadue" =

$$(372)_{10} = 3 \cdot 10^2 + 7 \cdot 10^1 + 2 \cdot 10^0$$

$$(174)_{16} = 1 \cdot 16^2 + 7 \cdot 16^1 + 4 \cdot 16^0$$

$$(564)_8 = 5 \cdot 8^2 + 6 \cdot 8^1 + 4 \cdot 8^0$$

$$\begin{aligned}(1011110100)_2 &= 1 \cdot 2^8 + 0 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + \\ &\quad + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0\end{aligned}$$

"duecentoottantasette" =

$$(287)_{10} = (100011111)_2 = (10133)_4$$

$$(437)_8 = (11F)_{16} = (8V)_{32}.$$

Osservazione

All'aumentare della base, aumenta il numero dei simboli. Al contrario, con una base più grande sono in genere necessarie meno cifre per rappresentare uno stesso valore.

Aritmetica in base 2

Le tabelline di somma e prodotto

Somma		
+	0	1
0	0	1
1	1	10

Prodotto		
.	0	1
0	0	0
1	0	1

Somma con riporto				
c_1	c_0	riporto	s	riporto
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Osservazione

Una base bassa corrisponde a un minor numero di combinazioni di cifre e, di conseguenza, ad un'aritmetica più semplice.

La rappresentazione posizionale di numeri reali < 1 in base β

Un numero $\alpha \in \mathbb{R}$, $0 \leq \alpha < 1$, si rappresenta in una base fissata $\beta > 1$ come

$$\alpha = (0.a_1a_2\dots a_t a_{t+1}\dots)_{\beta} = a_1\beta^{-1} + a_2\beta^{-2} + \dots + a_t\beta^{-t} + a_{t+1}\beta^{t+1} + \dots$$

dove “.” è il *punto radice* e a_i sono cifre appartenenti all'insieme dei simboli associati alla base β .

Lo zero a sinistra del punto radice può essere omesso.

Tutto ciò che è a destra del punto radice è il peso di una potenza negativa della base.

Esempi:

β	rappresentazione	valore
10	$(0.1)_{10}$	$1/10$
10	$(0.11)_{10}$	$1/10 + 1/100$
2	$(0.1)_2$	$1/2$
2	$(0.11)_2$	$1/2 + 1/4$
16	$(0.1)_{16}$	16^{-1}

La somma delle potenze negative si può scrivere anche come

$$\alpha = (0.a_1a_2\dots a_t a_{t+1}\dots)_{\beta} = \sum_{i=1}^{\infty} a_i \beta^{-i}.$$

Si osservi che per rappresentare certi valori (numeri periodici, irrazionali,...) è necessario un numero infinito di cifre.

Rappresentazione di un qualsiasi numero reale

Un numero reale α si può ottenere come somma della sua parte intera più la sua parte frazionaria. Quindi, mettendo insieme i due contributi e tenendo conto del segno si ha

$$\begin{aligned}\alpha &= \text{segno}(\alpha) a_1 a_2 \dots a_p . a_{p+1} a_{p+2} a_{p+3} \dots \\ &= \text{segno}(\alpha) (a_1 \beta^{p-1} + a_2 \beta^{p-2} + \dots + a_p + a_{p+1} \beta^{-1} + a_{p+2} \beta^{-2} + \dots)\end{aligned}$$

Raccogliendo un fattore β^p tra tutti i termini, si può scrivere

$$\begin{aligned}\alpha &= \text{segno}(\alpha) (a_1 \beta^{-1} + a_2 \beta^{-2} + a_3 \beta^{-3} + \dots) \beta^p \\ &= \text{segno}(\alpha) \sum_{i=1}^{\infty} (a_i \beta^{-i}) \beta^p \\ &= \text{segno}(\alpha) m \beta^p\end{aligned}$$

dove

- **segno** vale $+1$ o -1 a seconda del **segno** di α ;
- $m = \sum_{i=1}^{\infty} (a_i \beta^{-i}) = (0.a_1 a_2 \dots)_{\beta}$ è un numero reale < 1 chiamato **mantissa**;
- β^p è un intero chiamato **parte esponente** di α .

Rappresentazione di numeri reali

Forma scientifica normalizzata

Le rappresentazioni del tipo segno–mantissa–esponente

$$\alpha = \text{segno}(\alpha)(0.a_1a_2\dots)\beta^p$$

si dicono in *forma scientifica*; se $a_1 \neq 0$ si parla di forma scientifica *normalizzata*.
ESEMPI

$(372)_{10}$	mista
$.372 10^3$	normalizzata
$.0372 10^4$	scientificia
$(3.141592\dots)_{10}$	mista
$.3141592 10^1$	normalizzata
$.3243F\dots 16^1$	normalizzata
$(3.243F\dots)_{16}$	mista

L'unicità della rappresentazione si ha nel caso della forma scientifica normalizzata.

Algoritmo delle divisioni successive

Serve per rappresentare un intero positivo $\alpha \in \mathbb{N}$ da base 10 ad una diversa base β .

L'algorithmo consiste nell'eseguire la divisione intera (quoziente e resto) del numero α per β finché non si giunge ad un quoziente nullo.

Esempio

Rappresentazione del numero $(1972)_{10}$ in base 2.

base 2

1972 : 2 = 986	resto 0
986 : 2 = 493	resto 0
493 : 2 = 246	resto 1
246 : 2 = 123	resto 0
123 : 2 = 61	resto 1
61 : 2 = 30	resto 1
30 : 2 = 15	resto 0
15 : 2 = 7	resto 1
7 : 2 = 3	resto 1
3 : 2 = 1	resto 1
1 : 2 = 0	resto 1

$$(1972)_{10} = (11110110100)_2$$

Algoritmo delle moltiplicazioni successive

Serve per rappresentare un reale decimale $\alpha \in [0, 1)$ in una base $\beta > 1$. Si tratta di determinare le cifre della rappresentazione

$$\alpha = (.a_1 a_2 a_3 \dots)_{\beta} = a_1 \beta^{-1} + a_2 \beta^{-2} + a_3 \beta^{-3} + \dots$$

moltiplicando il numero α per la base β .

Esempio

$(0.1)_{10}$

base 2

$$\begin{array}{ll} 0.1 \times 2 = 0.2 & \text{p. intera } 0 \\ 0.2 \times 2 = 0.4 & \text{p. intera } 0 \\ 0.4 \times 2 = 0.8 & \text{p. intera } 0 \\ 0.8 \times 2 = 1.6 & \text{p. intera } 1 \\ 0.6 \times 2 = 1.2 & \text{p. intera } 1 \\ 0.2 \times 2 = 0.4 & \text{p. intera } 0 \end{array}$$

...

$$(0.1)_{10} = (0.000\overline{1100})_2$$

Conversione di un reale da base 10 a base β

Ricordando che ogni numero si esprime come somma della sua parte intera e della sua parte frazionaria, gli algoritmi delle divisioni e delle moltiplicazioni successive possono essere impiegati per calcolare la conversione di base di un qualunque reale.

- ① Determinare $|\alpha|$, ricordando il segno.
- ② Determinare la parte intera $\lfloor |\alpha| \rfloor$ ed eseguire la conversione con l'algoritmo delle divisioni successive.
- ③ Determinare la parte frazionaria $|\alpha| - \lfloor |\alpha| \rfloor$ ed eseguire la conversione con l'algoritmo delle moltiplicazioni successive.
- ④ Scrivere il segno, la conversione della parte intera, il punto radice, la conversione della parte frazionaria.

Esempio

$\alpha = -(25.375)_{10}$. Convertire in base 2.

- ① $|\alpha| = 25.375$; segno = '-'.
- ② $\lfloor |\alpha| \rfloor = 25$; $(25)_{10} = (11001)_2$.
- ③ $|\alpha| - \lfloor |\alpha| \rfloor = .375$; $(.375)_{10} = (.011)_2$.
- ④ $\alpha = -(11001.011)_2$.

I numeri di macchina - Numeri Fixed Point

Rappresentazione in formato fixed point

Sommario della lezione:

- Rappresentazione degli interi in formato fixed point
- Operazioni con i numeri fixed point

Rappresentazione fixed point degli interi

Base 2, $t + 1$ bit

Il formato fixed point è un insieme di regole che, fissato un numero $t + 1$ di cifre, associa ad ogni numero dell'insieme

$$fi(\mathbb{Z}) = \mathbb{Z} \cap [-2^t, 2^t - 1]$$

una stringa di $t + 1$ cifre binarie

$$fi(N) = \boxed{c_{t+1} \quad c_t \quad \dots \quad \dots \quad \dots \quad c_2 \quad c_1}$$

Rappresentazione binaria degli elementi di $fi(\mathbb{Z})$

$$fi(\mathbb{Z}) = \mathbb{Z} \cap [-2^t, 2^t - 1]$$

Osserviamo che, per ogni $N \in fi(\mathbb{Z})$,

- se $N \geq 0$, allora N si rappresenta su t cifre binarie come

$$\begin{aligned} N &= (d_t d_{t-1} \cdots d_2 d_1)_2 \\ &= d_1 + d_2 2 + \cdots + d_t 2^{t-1}, \quad d_i \in \{0, 1\} \end{aligned}$$

- il massimo valore è $2^t - 1 = (\underbrace{11 \cdots 1}_t)_2$;

- se $N < 0$, allora N si rappresenta su $t + 1$ cifre binarie come

$$\begin{aligned} N &= -(d_{t+1} d_{t-1} \cdots d_2 d_1)_2 \\ &= -(d_1 + d_2 2 + \cdots + d_t 2^{t-1} + d_{t+1} 2^t), \quad d_i \in \{0, 1\} \end{aligned}$$

- il minimo valore è $-2^t = -(\underbrace{100 \cdots 0}_{t+1})_2$.

NB

A seconda del valore di N , alcune delle cifre con indice più alto potrebbero essere nulle. Esempio: $t = 7$, $N = 8$, $N = (00001000)_2$.

Rappresentazione fixed point degli interi in $fi(\mathbb{Z})$

Sia $N \in fi(\mathbb{Z})$. La stringa $fi(N)$ si definisce nel modo seguente.

- se $N \geq 0$, $N = (d_t d_{t-1} \cdots d_2 d_1)_2$.

$fi(N) =$	0	d_t	d_{t-1}	\cdots	\cdots	\cdots	d_2	d_1
-----------	---	-------	-----------	----------	----------	----------	-------	-------

- se $N < 0$, $N = -(d_{t+1} d_{t-1} \cdots d_2 d_1)_2$, si calcola la rappresentazione in *complemento a 2* nel seguente modo:

$$\tilde{d}_i = \begin{cases} 1 & \text{se } d_i = 0 \\ 0 & \text{se } d_i = 1 \end{cases} \Rightarrow \frac{\tilde{d}_{t+1} \quad \tilde{d}_t \quad \cdots \quad \tilde{d}_1 \quad +}{c_{t+1} \quad c_t \quad \cdots \quad c_1} =$$

(dove la somma è in aritmetica binaria)

$fi(N) =$	c_{t+1}	c_t	\cdots	\cdots	\cdots	c_2	c_1
-----------	-----------	-------	----------	----------	----------	-------	-------

NB

Se $N \in [-2^t, 0)$, la regola del complemento a 2 implica che $c_{t+1} = 1$

Esempi

$$t + 1 = 16$$

- $N = 1235 = (10011010011)_2$

$fi(N) =$

0	0	0	0	0	1	0	0	1	1	0	1	0	0	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

- $N = 4695 = (1001001010111)_2$

$fi(N) =$

0	0	0	1	0	0	1	0	0	1	0	1	0	1	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

- $N = -1235 = -(10011010011)_2$

- ① Rappresentazione binaria su 16 bit: $|N| = 0000010011010011$
- ② Scambio dei bit: 1111101100101100
- ③ Somma dell'unità: 1111101100101101

$fi(N) =$

1	1	1	1	1	0	1	1	0	0	1	0	1	1	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Valore di una rappresentazione fixed point

$$fi(N) = \begin{array}{ccccccccc} & c_{t+1} & c_t & \cdots & \cdots & \cdots & & c_2 & c_1 \end{array}$$

- Se $c_{t+1} = 0$, viene considerato come un numero positivo, il cui valore si ottiene mediante la regola della notazione posizionale

$$fi(N) = c_1 + c_2 \cdot 2 + \dots + c_t 2^{t-1}$$

- Se $c_{t+1} = 1$, viene interpretato come un numero negativo,

$$fi(N) = -(\bar{c}_1 + \bar{c}_2 2 + \cdots + \bar{c}_t 2^{t-1} + \bar{c}_{t+1} 2^t)$$

dove le cifre binarie \bar{c}_i si ottengono invertendo i passi dell'algoritmo di rappresentazione degli interi negativi:

$$\Rightarrow \frac{\begin{matrix} c_{t+1} & c_t & \cdots & c_1 & - \\ \tilde{c}_{t+1} & \tilde{c}_t & \cdots & \tilde{c}_1 \end{matrix}}{\begin{matrix} 1 & & & & \\ \tilde{c}_1 & & & & \end{matrix}} = \Rightarrow \bar{c}_i = \begin{cases} 1 & \text{se } \tilde{c}_i = 0 \\ 0 & \text{se } \tilde{c}_i = 1 \end{cases}$$

Esempi

- Caso $c_{t+1} = 0$:

$$fi(N) = \boxed{0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0}$$

$$fi(N) = (0111100011000010)_2 = 30914$$

- Caso $c_{t+1} = 1$:

$$fi(N) = \boxed{1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1}$$

$$1111000110000101 \xrightarrow{-1} 1111000110000100 \Rightarrow 0000111001111011$$

$$fi(N) = -(111001111011)_2 = -3707$$

Insieme dei numeri fixed point in base 2 con $t + 1 = 4$

\mathbb{Z}	$fi(N)$	N
1111		-1
1110		-2
1101		-3
1100		-4
1011		-5
1010		-6
1001		-7
1000		-8
0111		7
0110		6
0101		5
0100		4
0011		3
0010		2
0001		1
0000		0

Rappresentazione degli interi al calcolatore

- Il calcolatore adotta le regole della rappresentazione fixed point per gli interi.
- I formati standard sono:

tipo	$t + 1$	-2^t	$2^t - 1$
short (int)	16	-32768	32767
long int	32	$-2.147483648 \cdot 10^9$	$2.147483647 \cdot 10^9$

- Per gli interi che non appartengono all'insieme

$$fi(\mathbb{Z}) = \{-2^t, -(2^t - 1), \dots, 2^t - 1\}$$

si applicano le stesse regole, prendendo solo le $t + 1$ cifre più a destra della stringa ottenuta.

Sorpresa!

```
#include <iostream>
using namespace std;

int main()
{
    short i;

    i = 70231;
    printf("i=%i",i);
}
```

Esempi

base 2, $t + 1 = 16$

- $N = 70231 = (10001001001010111)_2$, 17 cifre: **overflow intero**

$fi(N) = \boxed{0} \boxed{0} \boxed{0} \boxed{1} \boxed{0} \boxed{0} \boxed{1} \boxed{0} \boxed{0} \boxed{1} \boxed{0} \boxed{1} \boxed{0} \boxed{1} \boxed{1} \boxed{1} \boxed{1}$

$fi(N) = fi(N_1)$, con $N_1 = (001001001010111)_2 = 4695$

- $N = -34622 = -(1000011100111110)_2$

$1000011100111110 \Rightarrow 0111100011000001 \Rightarrow \underbrace{0111100011000010}_{16 \text{ cifre}}$

underflow intero

$fi(N) = \boxed{0} \boxed{1} \boxed{1} \boxed{1} \boxed{1} \boxed{0} \boxed{0} \boxed{0} \boxed{1} \boxed{1} \boxed{0} \boxed{0} \boxed{0} \boxed{0} \boxed{1} \boxed{0}$

$fi(N) = fi(N_1)$, con $N_1 = (0111100011000010)_2 = 30914$

Esempi

base 2, $t + 1 = 16$

- $N = 61829 = (1111000110000101)_2$, 16 cifre: **overflow intero**

$$fi(N) = \boxed{1} \quad \boxed{1} \quad \boxed{1} \quad \boxed{1} \quad \boxed{0} \quad \boxed{0} \quad \boxed{0} \quad \boxed{1} \quad \boxed{1} \quad \boxed{0} \quad \boxed{0} \quad \boxed{0} \quad \boxed{0} \quad \boxed{0} \quad \boxed{1} \quad \boxed{0} \quad \boxed{1}$$

$$fi(N) = fi(N_1), \text{ con } N_1 = -(111001111011)_2 = -3707$$

- $N = -66771 = -(10000010011010011)_2$

$$10000010011010011 \Rightarrow 01111101100101100 \Rightarrow \underbrace{01111101100101101}_{17 \text{ cifre}}$$

underflow intero

$$fi(N) = \boxed{1} \quad \boxed{1} \quad \boxed{1} \quad \boxed{1} \quad \boxed{1} \quad \boxed{0} \quad \boxed{1} \quad \boxed{1} \quad \boxed{0} \quad \boxed{0} \quad \boxed{1} \quad \boxed{0} \quad \boxed{1} \quad \boxed{1} \quad \boxed{0} \quad \boxed{1}$$

$$fi(N) = fi(N_1), \text{ con } N_1 = -1235$$

- Il calcolatore estende la rappresentazione fixed point a tutti gli interi, utilizzando le $t + 1$ cifre più a destra.
- Ogni numero dell'insieme $fi(\mathbb{Z}) = \{-2^t, \dots, 2^t - 1\}$ rappresenta più valori interi
- Nella rappresentazione di interi non appartenenti a $fi(\mathbb{Z})$, si verifica overflow o underflow, ossia una perdita di informazione sul valore del numero.

Insieme dei numeri fixed point in base 2 con $t + 1 = 4$

\mathbb{Z}	$fi(N)$	N
	1111	-1
	1110	-2
	1101	-3
	1100	-4
	1011	-5
	1010	-6
	1001	-7
-9	1000	-8
7	→ 0111	7
23	↗ 0110	6
	0101	5
	0100	4
	0011	3
	0010	2
	0001	1
	0000	0

- L'insieme $fi(\mathbb{Z})$ non è chiuso rispetto alle quattro operazioni aritmetiche intere (addizione, sottrazione, moltiplicazione e divisione intera)
 - anche se i dati appartengono a $fi(\mathbb{Z})$, il risultato potrebbe non appartenere a $fi(\mathbb{Z})$ (overflow o underflow).
- Le operazioni del calcolatore vengono ridefinite in modo che il risultato sia un numero fixed point
 - La rappresentazione fixed point del risultato si ottiene prendendone le $t + 1$ cifre meno significative.

Sorpresa!

```
#include <iostream>
using namespace std;

int main()
{
    short i,j,s;
    i = 3900;
    j = 30000;
    s = i + j;
    printf("i %i j %i s %i",i,j,s);
}
```

Esempi: somma fixed point

Base 2, $t + 1 = 6$

- $N_1 = 010010 (= 18_{10}), N_2 = 000101 (= 5_{10}) \Rightarrow fi(N_1 + N_2) = 010111 (= 23_{10}).$

$$\begin{array}{r} 010010 \\ 000101 \\ \hline 010111 \end{array}$$

- **OVERFLOW** $N_1 = 010011 (= 19_{10}), N_2 = 001110 (= 14_{10}) \Rightarrow fi(N_1 + N_2) = 100001 = fi(-31_{10})$ invece di $33_{10}.$

$$\begin{array}{r} 010011 \\ 001110 \\ \hline 100001 \end{array}$$

- $N_1 = (-7)_{10}, N_2 = (-1)_{10} \Rightarrow fi(N_1 + N_2) = 111000 = fi(-8_{10}).$

$$\begin{array}{r} 1 & 1 & 1 & 0 & 0 & 1 & + \\ 1 & 1 & 1 & 1 & 1 & 1 & = \\ \hline 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{array}$$

- **UNDERFLOW** $N_1 = (-19)_{10}, N_2 = (-14)_{10} \Rightarrow fi(N_1 + N_2) = 011111 = fi(31_{10})$ invece di $-33_{10}.$

$$\begin{array}{r} 1 & 0 & 1 & 1 & 0 & 1 & + \\ 1 & 1 & 0 & 0 & 1 & 0 & = \\ \hline 1 & 0 & 1 & 1 & 1 & 1 & 1 \end{array}$$

- Il calcolatore rappresenta gli interi in formato fixed point.
- Solo un sottoinsieme limitato di interi è rappresentato esattamente al calcolatore.
- Numeri troppo grandi o troppo piccoli (overflow o underflow) vengono rappresentati con una perdita di informazione, ossia con un *errore*.
- I risultati delle operazioni aritmetiche tra numeri fixed point vengono rappresentati all'interno dell'insieme $fi(\mathbb{Z})$ tramite le $t + 1$ cifre meno significative.
- In caso di underflow o di overflow, anche il risultato di un'operazione viene rappresentato con un errore

La rappresentazione dei numeri reali: i numeri floating point

- L'insieme \mathbb{R} dei numeri reali è *simulato* al calcolatore mediante un insieme discreto e finito di numeri reali.
- Ogni numero reale $\alpha \in \mathbb{R}$ viene rappresentato con un numero in *virgola mobile* o *floating point*, indicato con $fl(\alpha)$.
- La funzione fl associa ad ogni numero reale il suo rappresentante floating point.
- Nel seguito descriviamo le regole che definiscono la funzione fl e l'insieme dei numeri floating point

In generale

Fissata una base β , ogni numero reale α si può rappresentare in modo univoco nella forma

$$\alpha = \pm m\beta^p$$

dove

- $m \in [0, 1)$ è la mantissa di α :

$$m = (0.a_1a_2\dots a_t a_{t+1}\dots)_\beta = \sum_{i=1}^{\infty} a_i \beta^{-i}$$

- $a_i, i = 1, 2, \dots$ sono le cifre della rappresentazione della mantissa nella base β
- $a_1 \neq 0$
- $p \in \mathbb{Z}$ è l'esponente.

$$\alpha = \pm(0.a_1a_2\dots a_ta_{t+1}\dots)_{\beta} \beta^r$$

Osserviamo che

- Le cifre della rappresentazione della mantissa potrebbero essere infinite
- L'esponente r può essere un qualsiasi numero intero.

L'insieme dei numeri floating point si ottiene ponendo una limitazione al numero di cifre della mantissa e limiti inferiore e superiore per la scelta dell'esponente.

L'insieme dei numeri floating point

$$F(\beta, t, L, U) =$$

$$\{\pm(0.a_1a_2...a_t)_\beta \beta^r : a_i \in \{0, \dots, \beta - 1\}, \forall i = 1, \dots, t, a_1 \neq 0, r \in \mathbb{Z}, L \leq r \leq U\}$$

- E' un insieme discreto e finito di numeri reali.
- E' simmetrico rispetto all'origine.
- La sua cardinalità è data da $2(\beta - 1)\beta^{t-1}(U - L + 1)$, che si ottiene contando tutte le possibili combinazioni di mantissa ed esponente.

Caso $\beta = 2$

La normalizzazione nel caso binario implica necessariamente $a_1 = 1$. In altri termini, ogni numero reale ha un'unica rappresentazione binaria nella forma

$$\alpha = \pm(1.a_2\dots a_t a_{t+1}\dots)_2 2^r$$

In questo caso l'insieme dei numeri floating point si scrive come

$$F(2, t, L, U) = \{\pm(1.a_2\dots a_t)_2 2^r : a_i \in \{0, 1\}, \forall i = 2, \dots, t, r \in \mathbb{Z}, L \leq r \leq U\}$$

- Il numero più grande dell'insieme è

$$(1.1\dots 1)_2 2^U = (2 - 2^{1-t})2^U$$

- Il numero più piccolo in valore assoluto è

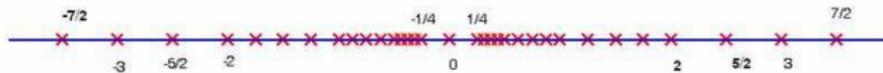
$$(1.0\dots 0)_2 2^L = 2^L$$

Esempio: $\beta = 2, t = 3, L = -2, U = 1$

Gli elementi dell'insieme F sono:

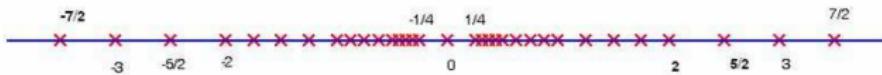
$$\pm 1.a_2a_3 \cdot 2^r, \quad r = -2, -1, 0, 1; a_2 = 0, 1; a_3 = 0, 1$$

$m \setminus r$	-2	-1	0	1
1.00	$\pm 1/4$	$\pm 1/2$	± 1	± 2
1.01	$\pm 5/16$	$\pm 5/8$	$\pm 5/4$	$\pm 5/2$
1.10	$\pm 6/16$	$\pm 3/4$	$\pm 3/2$	± 3
1.11	$\pm 7/16$	$\pm 7/8$	$\pm 7/4$	$\pm 7/2$



Osservazioni

$$\pm(1.a_2\dots a_t)_\beta \beta^r$$



- Attorno allo 0 c'è un intervallo $[-2^L, 2^L]$ di numeri reali che vengono rappresentati come 0 (underflow floating point)
- Per numeri più grandi di $(2 - 2^{1-t})2^U$ o più piccoli di $-(2 - 2^{1-t})2^U$ si incorre in overflow floating point
- Numeri piccoli sono meglio rappresentati.

Formati standard

Convenzioni del formato IEEE (Institute of Electrical and Electronic Engineering) documento 754 ANSI

	SEMPLICE PRECISIONE	DOPPIA PRECISIONE
N. totale di bit	32 (4 byte)	64 (8 byte)
segno s	1 bit	1 bit
$t-1$	23	52
l	8	11
bias	127 (01111111)	1023 (011111111111)
U	127	1023
L	-126	-1022

$$\pm(1.a_2\dots a_t)_2 \beta^p$$



Esempio

Rappresentazione floating point di $\alpha = 12.3125$.

$$(12.3125)_{10} = (1100.0101)_2 = (1.\textcolor{red}{1000101})_2 \cdot 2^3$$

$$p = 3 \Rightarrow p + \text{bias} = 130$$

$$(130)_{10} = (\textcolor{blue}{10000010})_2$$

0 **10000010** **1000101**000000000000000000000000

Rappresentazione di un reale in virgola mobile

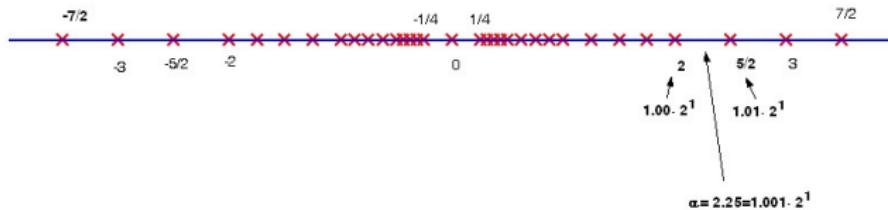
Un numero reale che non appartiene all'insieme F viene rappresentato all'interno di esso per *arrotondamento* oppure per *troncamento* della sua mantissa alla t -esima cifra significativa.

Esempio: $\beta = 2$, $t = 3$, $L = -2$, $U = 1$

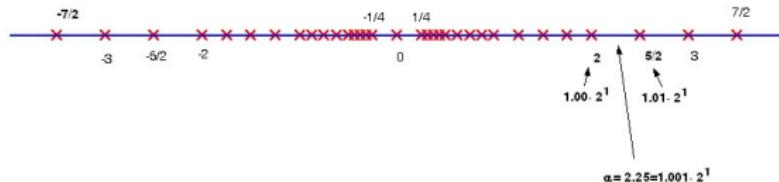
$$2.25 = (1.001) \cdot 2^1$$

↗
↘

troncamento: $1.00 \cdot 2^1$
arrotondamento: $1.01 \cdot 2^1$



Esempio: $\beta = 2$, $t = 3$, $L = -2$, $U = 1$



Troncamento

Tutti i numeri dell'intervallo $[2, 5/2)$ vengono rappresentati come 2.

$$2 = \begin{array}{l} 1.00\ 2^1 \\ 1.001\ 2^1 \\ 1.0011\ 2^1 \\ \dots \end{array} \left. \right\} \xrightarrow{fl} 1.00\ 2^1$$

$$\frac{5}{2} = 1.01\ 2^1 \xrightarrow{fl} 1.01\ 2^1$$

Arrotondamento

Tutti i numeri dell'intervallo $[15/8, 9/4)$ vengono rappresentati come 2.

$$\frac{15}{8} = \begin{array}{l} 1.111\ 2^0 \\ 1.1111\ 2^0 \\ 1.111101\ 2^0 \\ 1.00\ 2^1 \\ 1.0001\ 2^1 \\ 1.00011\ 2^1 \\ \dots \end{array} \left. \right\} \xrightarrow{fl} 1.00\ 2^1$$

$$\frac{9}{4} = 1.001\ 2^1 \xrightarrow{fl} 1.01\ 2^1$$

Esempio

$\alpha = 0.1$, precisione semplice.

$$\begin{aligned}(0.1)_{10} &= (0.0001100110011001100110011001100...)_2 \\&= (1.\textcolor{red}{100110011001100110011001}\textcolor{blue}{1}100...)_2 2^{-4} \\p &= -4 \Rightarrow p + \text{bias} = 123 \\(123)_{10} &= (\textcolor{blue}{1111011})_2\end{aligned}$$

Troncamento:

0 $\textcolor{blue}{01111011} \textcolor{red}{10011001100110011001100}$

Arrotondamento:

0 $\textcolor{blue}{01111011} \textcolor{red}{10011001100110011001101}$

Osservazione importante

Errore di rappresentazione

Ogni volta che un numero reale viene rappresentato in virgola mobile con precisione finita (t fissato), se questo non appartiene all'insieme $F(\beta, t, L, U)$ si ha una perdita di informazione, ossia si commette un *errore*. La motivazione fondante dell'analisi numerica è la necessità di valutare problemi matematici e relativi algoritmi tenendo conto di questi errori.

Errore assoluto ed errore relativo

Sia α un numero reale. L'errore assoluto commesso approssimando α con un altro numero reale α^* è definito come

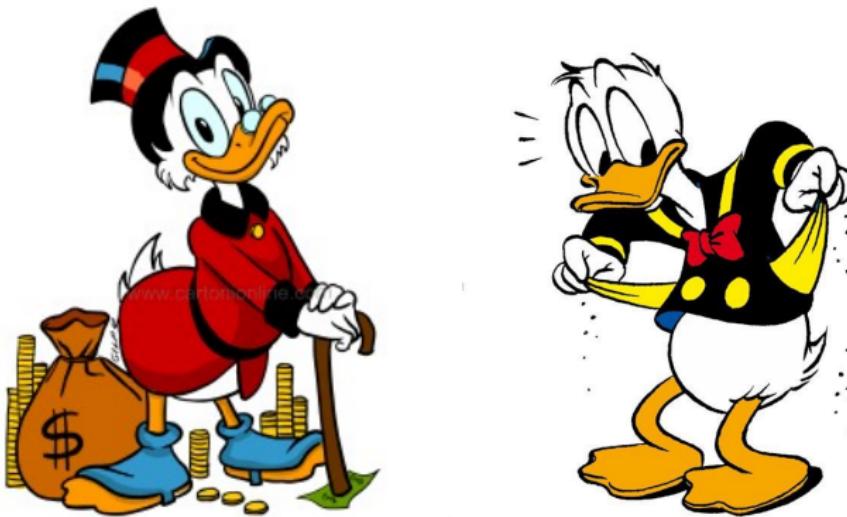
$$E_a = |\alpha - \alpha^*|.$$

Se $\alpha \neq 0$, si definisce l'errore relativo come

$$E_r = \frac{E_a}{|\alpha|}.$$

Errori assoluti e relativi

Esempi



$$E_a = |\alpha - \alpha^*| \quad E_r = \frac{E_a}{|\alpha|}$$

α	α^*	E_a	E_r
$0.3 \cdot 10^1$	$0.31 \cdot 10^1$	0.1	$0.3333\dots \cdot 10^{-1}$
$0.3 \cdot 10^{-3}$	$0.31 \cdot 10^{-3}$	$0.1 \cdot 10^{-4}$	$0.3333\dots \cdot 10^{-1}$
$0.3 \cdot 10^4$	$0.31 \cdot 10^4$	$0.1 \cdot 10^3$	$0.3333\dots \cdot 10^{-1}$

Teorema dell'errore di rappresentazione dei numeri reali

Sia α un numero reale e $fl(\alpha)$ la sua rappresentazione in virgola mobile in base β , con t cifre di precisione per la mantissa. L'errore relativo commesso approssimando α con $fl(\alpha)$ soddisfa la seguente disegualanza

$$\frac{|fl(\alpha) - \alpha|}{|\alpha|} \leq k\beta^{1-t},$$

dove $k = \frac{1}{2}$ oppure $k = 1$ a seconda che la rappresentazione sia ottenuta per arrotondamento o per troncamento, rispettivamente.

- La quantità $u = k\beta^{1-t}$ che si trova al lato destro della disegualanza si chiama **precisione di macchina**.
- Un modo equivalente per esprimere la stessa stima è

$$fl(\alpha) = \alpha(1 + \epsilon) \quad |\epsilon| \leq k\beta^{1-t}.$$

Proprietà caratteristica della precisione di macchina

La precisione di macchina u può essere definita anche come il più piccolo numero reale tale che $fl(1+u) \neq 1$.

Esempi: $\beta = 2$, $t = 3$, troncamento

$$u = 2^{-2} = (0.01)_2, \quad \begin{array}{rcl} 1 & + & 0.01 \\ 1 & + & 0.001 \end{array} = \begin{array}{rcl} 1.01 \\ 1.001 \end{array} \xrightarrow{fl} \begin{array}{l} 1.01 \\ 1.00 \end{array}$$

Esempi: $\beta = 2$, $t = 3$, arrotondamento

$$u = \frac{1}{2}2^{-2} = (0.001)_2, \quad \begin{array}{rcl} 1 & + & 0.001 \\ 1 & + & 0.0001 \end{array} = \begin{array}{rcl} 1.001 \\ 1.0001 \end{array} \xrightarrow{fl} \begin{array}{l} 1.01 \\ 1.00 \end{array}$$

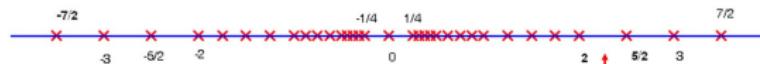
Operazioni con i numeri floating point

Osservazione

L'insieme F non è chiuso rispetto alle operazioni aritmetiche. Rispetto ai numeri in virgola fissa, oltre ai casi di overflow e underflow si può verificare la situazione seguente.

Esempio: $\beta = 2, t = 3, L = -2, U = 1$

$m \setminus r$	-2	-1	0	1
1.00	$\pm 1/4$	$\pm 1/2$	± 1	± 2
1.01	$\pm 5/16$	$\pm 5/8$	$\pm 5/4$	$\pm 5/2$
1.10	$\pm 6/16$	$\pm 3/4$	$\pm 3/2$	± 3
1.11	$\pm 7/16$	$\pm 7/8$	$\pm 7/4$	$\pm 7/2$



$$2 + \frac{1}{4} = (1.001)_2 \cdot 2^1 \notin F$$

Le operazioni di macchina

(o operazioni floating point)

- Abbiamo a che fare con un altro tipo di aritmetica, in cui le operazioni vengono ridefinite in modo che il risultato appartenga all'insieme F
- Il risultato di un'operazione tra numeri dell'insieme F è un numero reale: il risultato dell'operazione di macchina corrispondente è la rappresentazione floating point di esso.

$$F \times F \xrightarrow{+-\diagup} \mathbb{R} \xrightarrow{fl} F$$

- Conseguenza dell'ultimo passaggio che definisce le operazioni di macchina è l'introduzione di un potenziale errore che capita ogni volta che il risultato dell'operazione non appartiene ad F e deve quindi essere arrotondato o troncato alla t -esima cifra della mantissa.

Errore di rappresentazione del risultato delle operazioni

Teorema

Siano $x, y \in F(\beta, t, L, U)$. Indichiamo genericamente con \bullet una qualsiasi delle quattro operazioni aritmetiche. L'errore relativo commesso approssimando il risultato esatto dell'operazione $x \bullet y$ con la sua rappresentazione floating point soddisfa la diseguaglianza

$$\frac{|fl(x \bullet y) - (x \bullet y)|}{|x \bullet y|} \leq k\beta^{1-t}$$

dove $k = \frac{1}{2}$ oppure $k = 1$ a seconda che la rappresentazione sia ottenuta per arrotondamento o per troncamento rispettivamente.

- Si tratta del teorema dell'errore di rappresentazione dei numeri reali applicato nel caso particolare $\alpha = x \bullet y$.
- Si può scrivere in forma equivalente come

$$fl(x \bullet y) = (x \bullet y)(1 + \epsilon) \quad |\epsilon| \leq k\beta^{1-t}$$

Take home message

- Il calcolatore ha a disposizione un insieme limitato e finito di numeri reali.
 - Ogni volta che si assegna un valore reale ad una variabile in semplice o doppia precisione si commette un potenziale errore.
- Il calcolatore utilizza un'aritmetica diversa dalla nostra
 - Ogni volta che si esegue una operazione aritmetica al calcolatore si commette un potenziale errore
- Entrambi questi errori sono maggiorati dalla precisione di macchina.



Operazioni di macchina in dettaglio

Somma algebrica

$$x, y \in F(\beta, t, L, U) \rightarrow z = fl(x \pm y)$$

$$\begin{aligned} x &= xm \beta^{xe} \\ y &= ym \beta^{ye} \end{aligned} \Rightarrow z = zm \beta^{ze}$$

- ① **Normalizzazione degli addendi.** Si confrontano xe e ye . Si scala il numero con l'esponente più basso in modo che gli addendi abbiano lo stesso esponente (quello più alto)
- ② **Somma delle mantisse.** Si esegue la somma delle mantisse normalizzate.
- ③ **Troncamento o arrotondamento.** Si memorizzano le prime t cifre significative (per arrotondamento o troncamento) in zm
- ④ **Normalizzazione del risultato.** Si normalizza il risultato aggiustando l'esponente in modo che la mantissa sia < 1 .

NB

Le prime t cifre significative della mantissa si contano a partire dalla prima non nulla a sinistra andando verso destra.

Esempi

Negli esempi seguenti, per semplificare i calcoli, utilizziamo la base 10. L'insieme dei numeri floating point è quindi definito come

$$F(10, t, L, U) = \{0.a_1a_2\dots a_t 10^p : a_i \in \{0, \dots, 9\}, a_1 \neq 0, p \in \mathbb{Z}, L \leq p \leq U\}$$

Ricordiamo anche le regole generali

- Gli operandi di un'operazione floating point devono essere numeri floating point
- Il risultato di un'operazione floating point deve essere un numero floating point

Esempio, $\beta = 10$, $t = 5$, arrotondamento

Somma di macchina

$$x = .64932 \cdot 10^7; y = .53726 \cdot 10^4$$

- ➊ L'esponente più grande è $xe = 7$; si deve quindi riscrivere y portandolo all'esponente 7. Si tratta di spostare il punto radice nella mantissa a destra di $xe - ye = 3$ posizioni.

$$y = .00053726 \cdot 10^7$$

- ➋ Somma delle mantisse normalizzate al punto precedente

$$R = .64932 + .00053726 = .64985726$$

- ➌ Le prime t cifre significative con arrotondamento di R danno la mantissa del risultato

$$zm = .64986$$

- ➍ Il numero è già normalizzato (cioè la prima cifra a destra del punto radice è diversa da zero), quindi l'esponente è $ze = 7$.

Il risultato della somma è il numero floating point $z = .64986 \cdot 10^7$.

Esempio, $\beta = 10, t = 5$, arrotondamento

Somma di macchina

$$x = .64932 \cdot 10^7; y = .53726 \cdot 10^7$$

- ① In questo caso gli addendi hanno già lo stesso esponente ($xe - ye = 0$) e non è necessaria nessuna operazione di normalizzazione iniziale;
- ② Si procede a sommare le mantisse

$$R = .64932 + .53726 = 1.18658$$

- ③ Si pongono le prime t cifre significative di R in zm

$$zm = .11866$$

- ④ Si deve tenere conto della normalizzazione finale, poiché $R > 1$. L'esponente è $ze = 7 + 1$.

Il risultato della somma è il numero floating point $z = .11866 \cdot 10^8$

Esempio, $\beta = 10, t = 5$, arrotondamento

Somma di macchina

$$x = .75869 \cdot 10^2; y = -.75868 \cdot 10^2$$

- ① Non è necessaria la normalizzazione iniziale ($xe - ye = 0$);
- ② La somma delle mantisse fornisce

$$R = .75869 - .75868 = .00001 \quad h = -4.$$

- ③ Le prime t cifre significative di R danno luogo a $zm = .1$
- ④ Per riportare il risultato alla normalizzazione scientifica si deve tenere conto della divisione per 10^4 del passo precedente. Si ha quindi $ze = 2 - 4$

Il risultato della somma è il numero floating point $z = .1 \cdot 10^{-2}$.

Esempio, $\beta = 10$, $t = 5$, arrotondamento

Somma di macchina

$$x = .62379 \cdot 10^7; y = .32881 \cdot 10^1$$

- ① L'esponente più grande è quello di x e $xe - ye = 6$;
- ② Somma delle mantisse $R = .62379 + .00000032881 = .62379032881$
- ③ Arrotondamento alle prime 5 cifre significative $zm = .62379$
- ④ Esponente del risultato $ze = 7$

Abbiamo ottenuto $x + y = z = .62379 \cdot 10^7$, ossia

$$x + y = x \text{ anche se } y \neq 0.$$

Errore di incolonnamento

La somma di due addendi entrambi non nulli ha dato come risultato il più grande, in valore assoluto, dei due. Questa situazione di verifica quando

$$|y| \leq \frac{u}{\beta} |x|.$$

L'errore di incolonnamento significa che per la somma floating point *non c'è un unico elemento neutro*.

Operazioni di macchina in dettaglio

Prodotto

$$x, y \in F(\beta, t, L, U) \rightarrow z = fl(x \cdot y)$$

$$\begin{aligned} x &= xm \beta^{xe} \\ y &= ym \beta^{ye} \end{aligned} \Rightarrow z = zm \beta^{ze}$$

- ① Si esegue il prodotto tra le mantisse $xm \cdot ym$;
- ② Si esegue troncamento o arrotondamento alla t -esima cifra significativa, normalizzando la mantissa in modo che la prima cifra significativa dopo il punto radice sia non nulla;
- ③ Si sommano gli esponenti, tenendo conto di un'eventuale normalizzazione della mantissa.

Esempio, $\beta = 10, t = 5$, arrotondamento

Prodotto di macchina

$$x = .11111 \cdot 10^7; y = .10202 \cdot 10^{-2}$$

- ① Prodotto delle mantisse $.11111 * .10202 = .0113354422$
- ② Arrotondamento e normalizzazione della mantissa $zm = .11335, h = 1$;
- ③ Somma degli esponenti, tenendo conto della normalizzazione della mantissa

$$ze = 7 - 2 - \underbrace{1}_{h} = 4$$

$$z = .11335 \cdot 10^4.$$

Operazioni di macchina in dettaglio

Quoziente

$$x, y \in F(\beta, t, L, U) \rightarrow z = fl(x/y)$$

$$\begin{array}{rcl} x & = & xm \beta^{xe} \\ y & = & ym \beta^{ye} \end{array} \Rightarrow z = zm \beta^{ze}$$

- ① Se $xm < ym$ si pone $h = 0$; altrimenti si divide xm per β e si pone $h = 1$;
- ② Si esegue $(xm/\beta^h)/ym$ e si pongono le t cifre più significative in zm ;
- ③ Si calcola l'esponente $ze = xe - ye + h$.

Esempio, $\beta = 10$, $t = 5$, arrotondamento

Quoziente di macchina

$$x = .62500 \cdot 10^0; y = .12500 \cdot 10^{-2}$$

- ① Scalatura del dividendo $.062500 \quad h = 1$
- ② Divisione delle mantisse $.06250/.12500 = .5;$
- ③ Calcolo dell'esponente $ze = 0 + 2 + 1 = 3$
 $z = .5 \cdot 10^3.$

Non validità delle proprietà formali delle operazioni aritmetiche

Le operazioni di macchina non godono delle proprietà fondamentali che valgono per le operazioni usuali. In particolare, negli esempi precedenti abbiamo verificato che

- L'elemento neutro della somma (e del prodotto) non è unico;
- L'opposto di un numero non è unico

Nelle prossime slides vedremo alcuni esempi che mostrano la **non validità** di

- associativa di somma e prodotto
- distributiva
- legge di annullamento del prodotto

Non validità della proprietà associativa della somma

Aritmetica di macchina con $\beta = 10, t = 7$

Calcoliamo la somma $x + y + z$ con

$$x = .1234567 \cdot 10^0; y = .6666325 \cdot 10^4; z = -.6666325 \cdot 10^4$$

Ricordiamo che, quando si eseguono più operazioni in sequenza, i risultati intermedi devono essere rappresentati in formato floating point.

1 $fl(fl(x + y) + z) = 0.123 \cdot 10^0$

$$\begin{aligned} fl(x + y) &= fl((.6666325 + .00001234567) \cdot 10^4) = \\ &= .6666448 \cdot 10^4 \end{aligned}$$

$$\begin{aligned} fl(fl(x + y) + z) &= fl((.6666448 - .6666325) \cdot 10^4) = \\ &= .123 \cdot 10^0 \end{aligned}$$

2 $fl(x + fl(y + z)) = .1234567 \cdot 10^0$

$$\begin{aligned} fl(y + z) &= 0 \\ fl(x + fl(y + z)) &= .1234567 \cdot 10^0 \end{aligned}$$

Conclusione:

$$fl(x + fl(y + z)) \neq fl(fl(x + y) + z)$$

Osservazione importante

L'esempio precedente può anche essere visto come un confronto tra due algoritmi diversi per la somma di tre numeri

ALGORITMO 1	ALGORITMO 2
$s \leftarrow x + y$	$s \leftarrow y + z$
$s \leftarrow s + z$	$s \leftarrow s + x$

Nell'esempio precedente, il risultato esatto dell'operazione è $x+y+z = .1234567 \cdot 10^0$, che coincide con il risultato calcolato in aritmetica finita dall'algoritmo 2, mentre il risultato calcolato in aritmetica finita dall'algoritmo 1 ha un errore assoluto rispetto al risultato esatto pari a $E_{alg1} = 0.4567 \cdot 10^{-4}$.

In generale

- Il risultato del calcolo di un'espressione, intesa come successione di operazioni aritmetiche, dipende dall'**algoritmo** usato per calcolarla.
- Siccome il risultato di operazioni di macchina contiene un errore, l'errore nel risultato finale dipende dall'algoritmo utilizzato.

Non validità della proprietà distributiva della somma rispetto al prodotto

$\beta = 10, t = 2$, troncamento

$$x = .91 \cdot 10^1; y = .92 \cdot 10^1; z = .10 \cdot 10^0$$

e mostriamo che

$$fl(x \cdot fl(y + z)) \neq fl(fl(xy) + fl(xz))$$

1 $fl(y + z) = fl((.92 + .010) \cdot 10^1) = .93 \cdot 10^1$

$$fl(x \cdot fl(y + z)) = fl(.91 \cdot 10^1 * .93 \cdot 10^1) = fl(0.8463 \cdot 10^1) = .84 \cdot 10^2$$

2 $fl(xy) = .83 \cdot 10^2$

$$fl(xz) = .91 \cdot 10^0$$

$$fl(fl(xy) + fl(xz)) = fl(.83 \cdot 10^2 + .91 \cdot 10^0) = fl((.83 + .0091) \cdot 10^2) = .83 \cdot 10^2$$

Non validità della legge di annullamento del prodotto

$\beta = 10, t = 7, L = -50, U = 49$

$$x = .2 \cdot 10^{-27}; y = .1 \cdot 10^{-26}.$$

e mostriamo che

$$fl(xy) = 0 \text{ anche se } x, y \neq 0$$

$$fl(xy) = fl(.2 \cdot 10^{-54}) = 0 \text{ UNDERFLOW!!}$$

Take home message

- Nell'aritmetica del calcolatore, non valgono molte delle proprietà delle operazioni usuali
- L'aritmetica finita fa sì che gli algoritmi eseguiti al calcolatore diano un risultato che, rispetto a quello che vorremo calcolare, presenta un errore
- Algoritmi diversi possono fornire risultati con errori diversi
- Vogliamo valutare gli algoritmi per poter selezionare quelli che danno gli errori più piccoli.



Analisi degli errori

Gli algoritmi numerici

- Vengono applicati per calcolare le soluzioni di problemi matematici.
- I dati in input, che vengono elaborati tramite una successione di operazioni, e i risultati in output sono numeri reali.
- Vengono eseguiti dal calcolatore.

!!!Attenzione!!!

- I dati di input sono affetti dall'errore di rappresentazione (dati perturbati);
- I risultati di ogni singola operazione sono affetti dall'errore di rappresentazione (aritmetica di macchina)
- Ogni risultato intermedio può esserne soggetto e influenzare i risultati di tutte le operazioni successive.
- L'accumulo di questi errori viene chiamato propagazione degli errori

Compiti dell'analisi numerica sono

- analizzare i problemi matematici per capire quanto le relative soluzioni variano in presenza di perturbazioni dei dati = Analisi del **condizionamento dei problemi**
- progettare algoritmi numerici che risentano poco degli effetti dell'aritmetica di macchina = Analisi della **stabilità degli algoritmi**

Due concetti fondamentali

Condizionamento e stabilità

I concetti di condizionamento dei problemi e stabilità degli algoritmi sono uno dei fili conduttori del nostro corso e di tutta l'analisi numerica.

- L'analisi del condizionamento di un problema permette una opportuna interpretazione delle soluzioni calcolate mediante algoritmi numerici
- L'analisi della stabilità degli algoritmi costituisce un criterio di scelta e classificazione di essi: inoltre suggerisce buone strategie di programmazione per l'elaborazione di numeri reali.

Nelle prossime slides presenteremo questi concetti.

Esempio

Problema: somma $x + y + z$. Algoritmo: $(x + y) + z$

- Quello che vorremmo calcolare $\alpha = x + y + z \dots$
- ...e la soluzione del calcolatore $\alpha^* = fl(fl(fl(x) + fl(y)) + fl(z))$
- $fl(s) = s(1 + \epsilon)$, con $|\epsilon| \leq u$

Dobbiamo considerare un errore per ogni dato, $\epsilon_x, \epsilon_y, \epsilon_z$ ed un errore per ogni operazione, ϵ_1, ϵ_2 . Tutti gli ' ϵ ' sono maggiorati dalla precisione di macchina.

- Primo passo

$$\begin{aligned} x + y fl(x) + &= fl(x(1 + \epsilon_x) + y(1 + \epsilon_y)) \\ fl(y) fl(\textcolor{red}{fl(x)} + \textcolor{blue}{fl(y)}) &= (x(1 + \epsilon_x) + y(1 + \epsilon_y))(1 + \epsilon_1) \end{aligned}$$

- Secondo passo

$$\begin{aligned} fl(fl(fl(x) + fl(y)) + \textcolor{red}{fl(z)}) &= fl(((x(1 + \epsilon_x) + y(1 + \epsilon_y))(1 + \epsilon_1) + \textcolor{red}{z(1 + \epsilon_z)})) \\ &= ((x(1 + \epsilon_x) + y(1 + \epsilon_y))(1 + \epsilon_1) + z(1 + \epsilon_z)) \cdot \\ &\quad \cdot (1 + \epsilon_2) \end{aligned}$$

Risultato:

$$\alpha^* = ((x(1 + \epsilon_x) + y(1 + \epsilon_y))(1 + \epsilon_1) + z(1 + \epsilon_z))(1 + \epsilon_2)$$

Esempio

Problema: somma $x + y + z$. Algoritmo: $(x + y) + z$

Calcoliamo l'errore relativo:

$$\begin{aligned}\alpha^* &= ((x(1 + \epsilon_x) + y(1 + \epsilon_y))(1 + \epsilon_1) + z(1 + \epsilon_z))(1 + \epsilon_2) \\ &\simeq x + y + z + x\epsilon_x + y\epsilon_y + z\epsilon_z + (x + y)\epsilon_1 + (x + y + z)\epsilon_2\end{aligned}$$

Nel calcolo vengono trascurati i termini del tipo $\epsilon \cdot \epsilon$

$$\begin{aligned}E_r &= \frac{\alpha^* - \alpha}{\alpha} \\ &\simeq \underbrace{\frac{x}{x+y+z}\epsilon_x + \frac{y}{x+y+z}\epsilon_y + \frac{z}{x+y+z}\epsilon_z}_{\text{errore inerente}} + \underbrace{\frac{x+y}{x+y+z}\epsilon_1 + \epsilon_2}_{\text{errore algoritmico}} \\ &= E_{in} + E_{alg}\end{aligned}$$

Errore inerente ed errore algoritmico

$$E_r \quad \simeq \quad \underbrace{\frac{x}{x+y+z} \epsilon_x + \frac{y}{x+y+z} \epsilon_y + \frac{z}{x+y+z} \epsilon_z}_{\text{errore inerente}} + \underbrace{\frac{x+y}{x+y+z} \epsilon_1 + \epsilon_2}_{\text{errore algoritmico}}$$

L'errore inerente

è l'errore che si commetterebbe se le operazioni fossero eseguite in aritmetica esatta ($\epsilon_1 = \epsilon_2 = 0$)

- NON dipende dall'algoritmo ma solo dai dati da come essi sono legati alle soluzioni del problema.

L'errore algoritmico

è l'errore che si commetterebbe se i dati fossero rappresentati esattamente ($\epsilon_x = \epsilon_y = \epsilon_z = 0$) ma le operazioni fossero eseguite in precisione finita

- dipende dall'algoritmo e dai dati.

Indice di condizionamento e indice algoritmico

$$\begin{aligned}|E_{in}| &= \left| \frac{x}{x+y+z} \epsilon_x + \frac{y}{x+y+z} \epsilon_y + \frac{z}{x+y+z} \epsilon_z \right| \\&\leq \left| \frac{x}{x+y+z} \right| |\epsilon_x| + \left| \frac{y}{x+y+z} \right| |\epsilon_y| + \left| \frac{z}{x+y+z} \right| |\epsilon_z| \\&\leq \left(\left| \frac{x}{x+y+z} \right| + \left| \frac{y}{x+y+z} \right| + \left| \frac{z}{x+y+z} \right| \right) u\end{aligned}$$

$$\begin{aligned}|E_{alg}| &= \left| \frac{x+y}{x+y+z} \epsilon_1 + \epsilon_2 \right| \\&\leq \left| \frac{x+y}{x+y+z} \right| |\epsilon_1| + |\epsilon_2| \\&\leq \left(\left| \frac{x+y}{x+y+z} \right| + 1 \right) u\end{aligned}$$

$$I_{cond} = \left| \frac{x}{x+y+z} \right| + \left| \frac{y}{x+y+z} \right| + \left| \frac{z}{x+y+z} \right|, \quad I_{alg} = \left| \frac{x+y}{x+y+z} \right| + 1$$

Amplificazione degli errori

Anche se abbiamo un “ \leq ”, nella pratica possiamo assumere

$$\frac{|E_{in}|}{|E_{alg}|} \simeq \frac{I_{cond}u}{I_{alg}u} \Rightarrow |E_r| \simeq (I_{cond} + I_{alg})u$$

- L'indice di condizionamento è il coefficiente di proporzionalità **AMPLIFICAZIONE** tra l'errore sui dati iniziali (u) e l'errore inerente
- L'indice algoritmico è il coefficiente di proporzionalità **AMPLIFICAZIONE** tra l'errore introdotto dalle singole operazioni di macchina (u) e l'errore algoritmico.
- La somma $I_{cond} + I_{alg}$ è il coefficiente di proporzionalità **AMPLIFICAZIONE** tra l'errore sui dati e sulle singole operazioni (u) e l'errore totale.

Osservazione:

La precisione di macchina u è piccola ($u \simeq 10^{-16}$ in doppia precisione), ma se anche solo uno tra I_{cond} e I_{alg} è grande, l'errore tra il risultato esatto e quello calcolato potrebbe essere grande.

$$|E_r| \quad \simeq \quad (I_{cond} + I_{alg})u$$

- In teoria, se conoscessimo l'indice di condizionamento del problema che vogliamo risolvere e l'indice algoritmico dell'algoritmo che vogliamo applicare, potremmo stimare l'errore relativo e quindi l'accuratezza e affidabilità del risultato calcolato
- In realtà, calcolare esplicitamente gli indici non è possibile per problemi e algoritmi mediamente complessi

Definizione di condizionamento

Un problema si dice ben condizionato se a piccole perturbazioni dei dati corrispondono altrettanto piccole variazioni delle soluzioni. Viceversa, un problema si dice mal condizionato se a piccole perturbazioni dei dati corrispondono (relativamente) grandi variazioni delle soluzioni.

- Il condizionamento è una proprietà della funzione che lega i dati alla soluzione di un problema, non dipende da come queste vengano calcolate.
- L'indice di condizionamento fornisce una quantificazione del condizionamento del problema: più è piccolo più il problema è ben condizionato.

Esempio

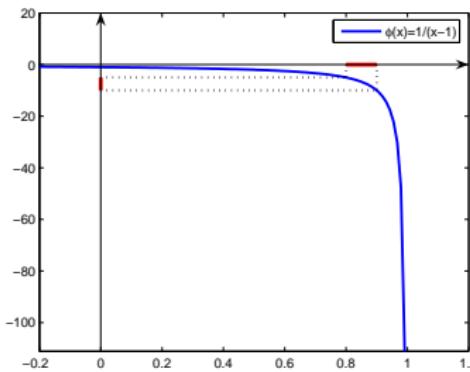
Calcolo di un'espressione

Problema: calcolare la soluzione dell'equazione lineare

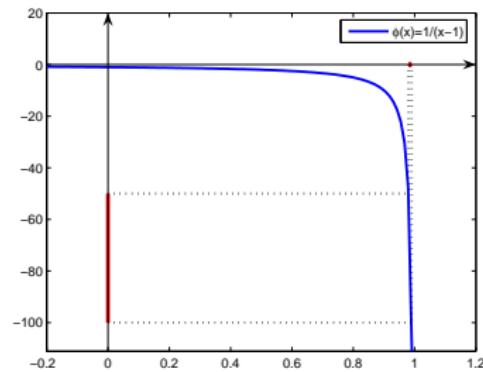
$$(x - 1)\alpha = 1$$

Dati: x . Soluzione $\alpha = \frac{1}{x-1}$.

Grafico della funzione che lega dato e soluzione del problema è



Ben condizionato per valori lontani da 1



Mal condizionato per valori vicini ad 1

Esempio

Calcolo delle radici di un polinomio di 2° grado

Supponiamo di voler calcolare le soluzioni dell'equazione

$$x^2 - 4x + c = 0.$$

Dati: 1, 4, c . Soluzioni:

$$x_{1,2} = 2 \pm \sqrt{4 - c}$$

Per semplicità studiamo come variano le soluzioni al variare del solo termine noto c e prendiamo la soluzione con il “-”.

$$c = 4 \quad \rightarrow \quad \alpha = 2$$

$$c^* = 4 - 10^{-6} \quad \rightarrow \quad \alpha^* = 2 - 10^{-3}$$

Variazioni sul dato: $\frac{c - c^*}{c} = -\frac{1}{4}10^{-6}$

Variazioni sulle soluzioni: $\frac{\alpha - \alpha^*}{\alpha} = \frac{1}{2}10^{-3}$.

Il problema è mal condizionato per valori dei dati vicino a 4

La variazione delle soluzioni è di 3 ordini di grandezza superiore alla perturbazione del dato: 10^{-3} vs. 10^{-6} .

Definizione di stabilità di un algoritmo

Un algoritmo si dice stabile se non è troppo sensibile agli errori di rappresentazione introdotti dalle operazioni in aritmetica finita. Viceversa, un algoritmo si dice instabile se tende ad amplificare gli errori dovuti all'aritmetica finita.

- La stabilità dipende dal tipo, dall'ordine delle operazioni che costituiscono l'algoritmo.
- L'indice algoritmico fornisce una quantificazione della stabilità: quanto più è vicino ad 1, tanto più l'algoritmo è stabile.
- Un algoritmo è più stabile di un altro se il suo indice è inferiore (dipende anche dai dati).

Esempio

Soluzione di un'equazione lineare.

Risolvere l'equazione lineare

$$7\alpha = 21$$

operando in aritmetica finita in base 10 con 7 cifre di precisione, arrotondamento.

Soluzione: $\alpha = 3$

ALGORITMO 1

$$\begin{aligned} t &\leftarrow fl\left(\frac{1}{7}\right) \\ \alpha^* &\leftarrow fl(21 \cdot t) \end{aligned}$$

ALGORITMO 2

$$\alpha^* \leftarrow fl\left(\frac{21}{7}\right)$$

$$\begin{aligned} t &= fl(0.142857142857143) \\ &= 0.1428571 \cdot 10^0 \\ \alpha^* &= fl(21 * 0.142857) \\ &= fl(2.9999991) \\ &= 2.999999 \end{aligned}$$

$$\alpha^* = 3$$

Conclusioni:

Il primo algoritmo produce una perdita di precisione sulla settima cifra: è meno stabile del secondo, che invece calcola la soluzione esatta. Il primo algoritmo ha anche una complessità maggiore, poiché è composto da due operazioni.

Esempio: somma di n numeri

$\beta = 10, t = 7$, arrotondamento

$$x_1 = 0.1 \cdot 10^1, x_i = 0.1 \cdot 10^{-6}, i = 2, \dots, 10$$

Risultato esatto $\sum_{i=1}^{10} x_i = 1 + 9 \cdot 10^{-7} = 1.0000009$

```
s ← x1;
for i = 2, ..., 10
    s ← s + xi;
```

$$\begin{array}{ll} i = 2 & s = fl(x_1 + x_2) = fl(0.1 \cdot 10^1 + 0.1 \cdot 10^{-6}) = .1 \cdot 10^1 = y_1 \\ i = 3 & s = fl(y_1 + x_3) = fl(0.1 \cdot 10^1 + 0.1 \cdot 10^{-6}) = .1 \cdot 10^1 = y_2 \end{array}$$

⋮

$$i = 10 \quad s = fl(y_{n-1} + x_n) = fl(0.1 \cdot 10^1 + 0.1 \cdot 10^{-6}) = .1 \cdot 10^1$$

$$\epsilon_r = 8.999.. \cdot 10^{-7} \sim 10^{-6}$$

Esempio: somma di n numeri

$\beta = 10, t = 7$, arrotondamento

$$x_1 = 0.1 \cdot 10^1, x_i = 0.1 \cdot 10^{-6}, i = 2, \dots, 10$$

Risultato esatto $\sum_{i=1}^{10} x_i = 1 + 9 \cdot 10^{-7} = 1.0000009$

```
s ← x10;
for i = 9, ..., 1 : -1
    s ← s + xi;
```

$$\begin{array}{lllll} i = 9 & s = fl(x_{10} + x_9) & = & fl(.1 \cdot 10^{-6} + .1 \cdot 10^{-6}) & = .2 \cdot 10^{-6} = z_1 \\ i = 8 & s = fl(z_1 + x_8) & = & fl(.2 \cdot 10^{-6} + .1 \cdot 10^{-6}) & = .3 \cdot 10^{-6} = z_2 \end{array}$$

⋮

$$\begin{array}{lllll} i = 2 & s = fl(z_7 + x_2) & = & fl(.8 \cdot 10^{-6} + .1 \cdot 10^{-6}) & = .9 \cdot 10^{-6} = z_8 \\ i = 1 & s = fl(z_8 + x_1) & = & fl(.9 \cdot 10^{-6} + 1) & = .100001 \cdot 10^1 \end{array}$$

$$\epsilon_r = 9.999.. \cdot 10^{-8} \sim 10^{-7}$$

L'errore relativo è 10 volte più piccolo

Esempio

Valutazione di un polinomio in un punto

Dati: x, a_0, \dots, a_n , calcolare il numero

$$\begin{aligned} p_n(x) &= a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0 \\ &= a_0 + x(a_1 + a_2 x + \dots + a_n x^{n-1}) \\ &= a_0 + x(a_1 + x(a_2 + a_3 x + \dots + a_n x^{n-2})) \\ &= a_0 + x(a_1 + x(a_2 + x(a_3 + \dots + x(a_{n-1} + a_n x)))) \end{aligned}$$

Forma canonica

```
s ← a0
p ← x
FOR i = 1, ..., n
    s ← s + aip
    p ← px
```

Schema di Ruffini-Horner

```
s ← an
FOR i = n - 1, ..., 0
    s ← sx + ai
```

Complessità computazionale:
 n somme, $2n$ prodotti.

L'algoritmo di Horner è più efficiente (ha una minore complessità computazionale) rispetto a quello basato sulla forma canonica del polinomio. Si può mostrare che, in generale, è più stabile, anche se entrambi gli algoritmi possono diventare instabili in prossimità delle radici del polinomio.

Complessità computazionale:
 n somme, n prodotti.

- La valutazione di un algoritmo numerico deve tenere conto non solo dell'efficienza (complessità computazionale), ma anche della stabilità;
- Particolari cautele vanno adottate nell'interpretare i risultati di un algoritmo numerico applicato ad un problema mal condizionato.

Richiami di calcolo matriciale

Calcolo matriciale

Definizioni

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \dots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{pmatrix} \in \mathbb{R}^{m \times n}$$

- m numero di righe, n numero di colonne;
- se $m = n$ è quadrata di ordine (o dimensione) n ;
- a_{ij} , i indice di riga, j indice di colonna;
- se $n = 1$, è un vettore colonna di m componenti;
- se $m = 1$, è un vettore riga di n componenti;
- \mathbb{R}^n è lo spazio dei vettori colonna di n componenti;
- se $n = m = 1$, è uno scalare;
- gli elementi a_{ii} , $i = 1, \min(n, m)$ sono detti *elementi diagonali*

NB

Il numero totale degli elementi di una matrice è nm , perciò la memorizzazione di una matrice utilizzando il formato double richiede uno spazio di $8nm$ byte. La memorizzazione può avvenire per righe (C++) oppure per colonne (Matlab, Fortran).

Calcolo matriciale

Definizioni

Se $A \in \mathbb{R}^{m \times n}$, la sua *trasposta* è una matrice in $\mathbb{R}^{n \times m}$ che si indica con A^T ed è definita come

$$A^T = \begin{pmatrix} a_{11} & a_{21} & a_{31} & \cdots & a_{n1} \\ a_{12} & a_{22} & a_{32} & \cdots & a_{n2} \\ \cdots & & & & \\ a_{1m} & a_{2m} & a_{3m} & \cdots & a_{nm} \end{pmatrix} \in \mathbb{R}^{m \times n}$$

- Una matrice quadrata di ordine n si dice *simmetrica* se $A = A^T$. In questo caso $a_{ij} = a_{ji}$, $i = 1, \dots, n$.
- La matrice di ordine n che ha elementi diagonali unitari, mentre tutti gli altri elementi sono nulli è la *matrice identità* di ordine n , e si indica con I (o con I_n , nel caso in cui occorra specificarne la dimensione)

$$I = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdots & & & & \\ 0 & \cdots & & & 1 \end{pmatrix}$$

Calcolo matriciale

Definizioni

- Una matrice quadrata D di ordine n si dice *diagonale* se $d_{ij} = 0$ per $i \neq j$, $i, j = 1, \dots, n$.

$$D = \begin{pmatrix} d_{11} & 0 & 0 & \cdots & 0 \\ 0 & d_{22} & 0 & \cdots & 0 \\ 0 & 0 & d_{33} & \cdots & 0 \\ \cdots & & & & \\ 0 & \cdots & & & d_{nn} \end{pmatrix}$$

- Una matrice quadrata U di ordine n si dice *triangolare superiore* se $u_{ij} = 0$ per $i > j$. Una matrice quadrata L di ordine n si dice *triangolare inferiore* se $l_{ij} = 0$ per $i < j$

$$U = \begin{pmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ 0 & 0 & u_{33} & \cdots & u_{3n} \\ \cdots & & & & \\ 0 & \cdots & & & u_{nn} \end{pmatrix} \quad L = \begin{pmatrix} l_{11} & 0 & 0 & \cdots & 0 \\ l_{21} & l_{22} & 0 & \cdots & 0 \\ l_{31} & l_{32} & l_{33} & \cdots & 0 \\ \cdots & & & & \\ l_{n1} & \cdots & & & l_{nn} \end{pmatrix}$$

- Conoscere la struttura di una matrice (diagonale, triangolare), permette di ridurre l'occupazione di memoria, poichè non è necessario memorizzare esplicitamente gli elementi nulli.
- Una matrice diagonale di ordine n ha al più n elementi non nulli. Per memorizzarli, occorre uno spazio di memoria corrispondente a n reali floating point.
- Gli elementi non nulli di una matrice triangolare sono $\simeq \frac{n^2}{2}$

Operazioni matriciali

Somma matriciale

Date due matrici della stessa dimensione $A, B \in \mathbb{R}^{m \times n}$, la somma $A + B$ è una matrice delle stesse dimensioni $C \in \mathbb{R}^{m \times n}$ i cui elementi si ottengono facendo la somma elemento per elemento di A e B :

$$c_{ij} = a_{ij} + b_{ij}, \quad i = 1, \dots, m, j = 1, \dots, n$$

Algoritmo della somma matriciale:

```
For i = 1, ..., m
    For j = 1, ..., n
        cij ← aij + bij
```

Complessità computazionale: nm somme floating point.

Operazioni matriciali: moltiplicazione per uno scalare

Data una matrice $A \in \mathbb{R}^{m \times n}$ e un numero reale λ , si definisce la matrice λA come la matrice $C \in \mathbb{R}^{m \times n}$ i cui elementi si ottengono moltiplicando i corrispondenti elementi di A per λ :

$$c_{ij} = \lambda a_{ij}, \quad i = 1, \dots, m, j = 1, \dots, n$$

Algoritmo della moltiplicazione per scalare:

```
For  $i = 1, \dots, m$ 
  For  $j = 1, \dots, n$ 
     $c_{ij} \leftarrow \lambda a_{ij}$ 
```

Complessità computazionale: nm profotti floating point.

Prodotto tra matrici

Prodotto righe–colonne

Date due matrici $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{n \times p}$, si definisce la matrice prodotto AB come la matrice $C \in \mathbb{R}^{m \times p}$ i cui elementi sono definiti dalla formula seguente:

$$c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}, \quad i = 1, \dots, m, j = 1, \dots, p$$

$$\begin{pmatrix} c_{11} & c_{1j} & c_{1p} \\ \vdots & & \\ c_{i1} & c_{ij} & c_{ip} \\ c_{m1} & c_{mj} & c_{mp} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{1j} & a_{1n} \\ & \boxed{a_{i1}} & a_{ij} & a_{in} \\ a_{m1} & a_{mj} & a_{mn} \end{pmatrix} \begin{pmatrix} b_{11} & b_{1j} & b_{1p} \\ b_{i1} & b_{ij} & b_{ip} \\ b_{n1} & b_{nj} & b_{np} \end{pmatrix}$$

Prodotto tra matrici

Prodotto righe–colonne

Date due matrici $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{n \times p}$, si definisce la matrice prodotto AB come la matrice $C \in \mathbb{R}^{m \times p}$ i cui elementi sono definiti dalla formula seguente:

$$c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}, \quad i = 1, \dots, m, j = 1, \dots, p$$

Algoritmo della moltiplicazione matrice–matrice:

```
For i = 1, ..., m
  For j = 1, ..., p
    s ← 0
    For k = 1, ..., n
      s ← s + aik bkj
    cij ← s
```

Complessità computazionale: per ogni elemento occorrono n somme e prodotti, gli elementi sono in tutto mp , quindi la complessità è nmp . Se la matrice è quadrata, $n = m = p$, la complessità è n^3 .

Proprietà del prodotto matriciale

- Valgono la proprietà associativa e la distributiva rispetto alla somma matriciale.
- NON vale la commutativa, nemmeno se le matrici sono quadrate.
- Elemento neutro:

$$AI_n = I_m A$$

- $(AB)^T = B^T A^T$.

Prodotti con matrici con struttura

Se una delle matrici da moltiplicare ha una struttura diagonale o triangolare, la complessità dell'operazione si può ridurre notevolmente, omettendo i prodotti con gli elementi nulli.

- Moltiplicare a sinistra per una matrice diagonale, cioè calcolare $C = DA$, significa moltiplicare gli elementi della riga i -esima di A per d_{ii} , al costo di nm prodotti
- Moltiplicare a destra per una matrice diagonale, cioè calcolare $C = AD$, significa moltiplicare gli elementi della colonna j -esima di A per d_{jj} , al costo di nm prodotti
- Moltiplicare per una matrice triangolare, a destra o sinistra, permette di risparmiare circa la metà del costo computazionale. Es. $A, L \in \mathbb{R}^{n \times n}$,
 $C = LA$

$$c_{ij} = \sum_{k=1}^i l_{ik} a_{kj}, \quad i, j = 1, \dots, n$$

Prodotto matrice-vettore

E' un caso particolare del prodotto matriciale, in cui $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$ e $y = Ax \in \mathbb{R}^m$ è definito come

$$y_i = \sum_{k=1}^n a_{ik}x_k, \quad i = 1, \dots, m$$

Algoritmo della moltiplicazione matrice–vettore:

```
For i = 1, ..., m
    s ← 0
    For k = 1, ..., n
        s ← s + aikxk
    yi ← s
```

Complessità computazionale: per ogni elemento occorrono n somme e prodotti, gli elementi sono in tutto m , quindi la complessità è nm . Se la matrice è quadrata, $n = m$, la complessità è n^2 .

Prodotto scalare tra due vettori

Dati due vettori $u, v \in \mathbb{R}^n$, si definisce prodotto scalare di u e v il numero s dato da

$$s = u^T v = \sum_{k=1}^n u_k v_k$$

Algoritmo per il calcolo del prodotto scalare tra vettori:

```
s ← 0
For k = 1, ..., n
    s ← s + v_k u_k
```

Complessità computazionale: n prodotti, n somme.

Basic Linear Algebra Subroutine

- Le operazioni vettoriali e matriciali che abbiamo visto finora, hanno una corrispondente implementazione ottimizzata all'interno della libreria BLAS
- <http://www.netlib.org/blas/>
- Molti software di calcolo scientifico che richiedono la manipolazione di matrici e vettori utilizzano questa libreria per le operazioni fondamentali.

Inversa di una matrice

Una matrice quadrata A di ordine n si dice *invertibile* o *non singolare* se esiste una matrice $X \in \mathbb{R}^{n \times n}$ tale che

$$AX = XA = I$$

La matrice X viene detta *inversa* di A e si indica come $X = A^{-1}$. Proprietà dell'inversa:

- $(A^{-1})^{-1} = A$
- $(A^T)^{-1} = (A^{-1})^T = A^{-T}$
- Se A e B sono invertibili, allora anche AB lo è e

$$(AB)^{-1} = B^{-1}A^{-1}$$

Determinante di una matrice

Regola di Laplace

Sia $A \in \mathbb{R}^{n \times n}$, il *determinante* di A è un numero denotato con $\det(A)$ che si può definire in modo ricorsivo:

$$\det(A) = \begin{cases} a_{11} & \text{se } n = 1 \\ \sum_{j=1}^n a_{ij} \det(A_{ij}) & \text{se } n > 1 \end{cases}$$

dove:

- i è un qualsiasi indice $i \in \{1, \dots, n\}$
- A_{ij} è la matrice di ordine $n - 1$ che si ottiene eliminando la riga i e la colonna j di A

Complessità computazionale: il determinante di una matrice di ordine n richiede n somme, n prodotti e il calcolo di n determinanti di ordine $n - 1$, ciascuno dei quali richiede $n - 1$ somme, $n - 1$ prodotti e il calcolo di $n - 1$ determinanti di ordine $n - 2, \dots$

In totale, la complessità del calcolo del determinante tramite la definizione è dell'ordine di $n!$

Proprietà del determinante.

- $\det(A) = \det(A^T)$;
- $\det(AB) = \det(A) \det(B)$;
- $\det(A^{-1}) = \frac{1}{\det(A)}$;
- $\det(\lambda A) = \lambda^n \det(A)$;
- Se B è una matrice ottenuta scambiando due righe (o due colonne) di A , allora $\det(B) = -\det(A)$;

Inoltre

Una matrice A è nonsingolare se e solo se $\det(A) \neq 0$.

Autovalori

Sia $A \in \mathbb{R}^{n \times n}$. Un vettore $x \in \mathbb{C}^n$, $x \neq 0$, e uno scalare $\lambda \in \mathbb{C}$ si dicono rispettivamente *autovettore* e *autovalore* di A se soddisfano l'uguaglianza seguente:

$$Ax = \lambda x$$

Il vettore x si dice anche autovettore relativo all'autovalore λ .

Gli n autovalori di A sono le soluzioni dell'equazione

$$\det(A - \lambda I) = 0$$

il cui lato sinistro è chiamato *polinomio caratteristico* di A .

Proprietà:

- $\det(A) = \lambda_1 \lambda_2 \cdot \dots \cdot \lambda_n$
- Una matrice è nonsingolare se e solo se tutti i suoi autovalori sono non nulli.
- Se A è non singolare e λ è un suo autovalore, allora $1/\lambda$ è autovalore di A^{-1}
- Se A è simmetrica, allora tutti i suoi autovalori sono reali.

Definizione di raggio spettrale

Sia A una matrice di ordine n e indichiamo con $\lambda_1, \dots, \lambda_n$ i suoi autovalori. Si definisce raggio spettrale di A il numero reale positivo indicato con $\rho(A)$ definito nel modo seguente:

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

La funzione $\|\cdot\| : \mathbb{R}^n \rightarrow \mathbb{R}$ è una *norma vettoriale* se soddisfa le seguenti proprietà:

- $\|x\| \geq 0, \forall x \in \mathbb{R}^n$, e $\|x\| = 0$ se e solo se $x = 0$;
- $\|\lambda x\| = |\lambda| \|x\|, \forall x \in \mathbb{R}^n, \lambda \in \mathbb{R}$;
- Proprietà triangolare:

$$\|x + y\| \leq \|x\| + \|y\|, \quad \forall x, y \in \mathbb{R}^n$$

Norme vettoriali

Norma ∞

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

Norma 1

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

Norma 2 (o *norma Euclidea*)

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

- Una norma vettoriale associa ad ogni vettore un numero che esprime la sua misura e viene calcolata tenendo conto di tutte le sue componenti.
- E' un'informazione utile per poter confrontare tra loro i vettori.
- Per la norma Euclidea vale la seguente uguaglianza:

$$\|x\|_2^2 = x_1^2 + x_2^2 + \dots + x_n^2 = x^T x$$

- Le norme vettoriali sono *equivalenti*, nel senso che prese due qualsiasi norme $\|\cdot\|_*$, $\|\cdot\|_\bullet$, con $*, \bullet \in \{\infty, 1, 2\}$, esistono due costanti positive m, M , tali che

$$m\|x\|_* \leq \|x\|_\bullet \leq M\|x\|_*, \quad \forall x \in \mathbb{R}^n$$

La funzione $\|\cdot\| : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}$ è una *norma matriciale* se soddisfa le seguenti proprietà:

- $\|A\| \geq 0, \forall A \in \mathbb{R}^{n \times n}$, e $\|A\| = 0$ se e solo se $A = 0$;
- $\|\lambda A\| = |\lambda| \|A\|, \forall A \in \mathbb{R}^{n \times n}, \lambda \in \mathbb{R}$;
- Proprietà triangolare:

$$\|A + B\| \leq \|A\| + \|B\|, \quad \forall A, B \in \mathbb{R}^{n \times n}$$

Norme matriciali indotte

A partire da una norma vettoriale $\|\cdot\|$, è possibile definire formalmente una *norma matriciale indotta* nel modo seguente:

$$\|A\| = \sup_{x \in \mathbb{R}^n, x \neq 0} \frac{\|Ax\|}{\|x\|}$$

Si può dimostrare che:

Norma ∞

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

Norma 1

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

Norma 2 (o *norma Euclidea*)

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

Proprietà delle norme matriciali

Sia $\bullet \in \{\infty, 1, 2\}$

Proprietà submoltiplicativa

Siano $A, B \in \mathbb{R}^{n \times n}$:

$$\|AB\|_{\bullet} \leq \|A\|_{\bullet} \|B\|_{\bullet}$$

Compatibilità con le norme vettoriali

Siano $A \in \mathbb{R}^{n \times n}, x \in \mathbb{R}^n$

$$\|Ax\|_{\bullet} \leq \|A\|_{\bullet} \|x\|_{\bullet}$$

- Le norme esprimono l'idea di una misura.
- Hanno lo stesso ruolo che ha il valore assoluto per gli scalari.
- Se $x, y \in \mathbb{R}^n$, il numero $\|x - y\|$ esprime il concetto della distanza tra x e y .
- Per valutare gli errori tra vettori si utilizzano le norme:

$$\frac{\|x - y\|}{\|x\|}$$

è l'errore relativo (la distanza relativa) tra x e y

- Le stesse considerazioni valgono anche per le matrici.

Esperimento

- Problema: calcolare il valore approssimato della derivata di una funzione f in un punto.
- Strumenti a disposizione: un programma che, dato un punto a , calcola il valore di f in a , ossia il numero $f(a)$.
- Idea: approssimare la definizione di derivata con il rapporto incrementale

$$f'(a) = \lim_{h \rightarrow 0} \frac{f(a+h) - f(a)}{h} \Rightarrow f'(a) \simeq \frac{f(a+h) - f(a)}{h}$$

per h “piccolo”

- Verifica: verifichiamo la bontà dell’idea scrivendo un programma Matlab per un caso in cui conosciamo la soluzione esatta del problema, in modo da poter valutare l’accuratezza della soluzione calcolata (problema test).

Problema test

Scegliamo per esempio

$$f(x) = \sin(x) \Rightarrow f'(x) = \cos(x)$$

e $a = 1$.

Soluzione esatta	Soluzione calcolata
$\alpha = \cos(a)$	$\alpha^* = \frac{\sin(a+h) - \sin(a)}{h}$

Scriviamo un programma Matlab che calcola diversi valori di α^* corrispondenti a diverse scelte di h :

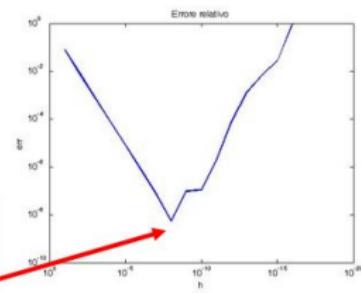
$$h \in \{10^{-1}, 10^{-2}, \dots, 10^{-16}\}$$

Codice Matlab

```
clear all
a           = 1;
h           = 10.^[-1:-1:-16];
alpha_star  = (sin(a+h)-sin(a))./h;
alpha       = cos(a);
err         = abs(alpha - alpha_star)/abs(alpha);
n           = length(h);
fprintf('%10s %6s %6s %10s\n', 'h', 'sol', 'der', 'err');
fprintf('-----\n');
for i=1:n
    fprintf(' %10.0e %6.5f %6.5f %10.4e \n',...
        h(i), alpha, alpha_star(i), err(i));
end
loglog(h,err,'LineWidth',2);
```

Risultati

h	sol	der	err
<hr/>			
1e-01	0.54030	0.49736	7.9471e-02
1e-02	0.54030	0.53609	7.8036e-03
1e-03	0.54030	0.53988	7.7887e-04
1e-04	0.54030	0.54026	7.7872e-05
1e-05	0.54030	0.54030	7.7871e-06
1e-06	0.54030	0.54030	7.7872e-07
1e-07	0.54030	0.54030	7.7415e-08
1e-08	0.54030	0.54030	5.4967e-09
1e-09	0.54030	0.54030	9.7244e-08
1e-10	0.54030	0.54030	1.0824e-07
1e-11	0.54030	0.54030	2.1631e-06
1e-12	0.54030	0.54035	8.0030e-05
1e-13	0.54030	0.53957	1.3583e-03
1e-14	0.54030	0.54401	6.8609e-03
1e-15	0.54030	0.55511	2.7409e-02
1e-16	0.54030	0.00000	1.0000e+00





- Ci saremmo aspettati che quanto più h fosse piccolo, tanto più accurata sarebbe stata la soluzione calcolata.
- Invece si ha un valore minimo dell'errore in corrispondenza di $h = 10^{-8}$ mentre per valori di h più piccoli l'errore è più grande. Infatti, se $f \in C^2$ e $|f''(y)| \leq M \forall y$, abbiamo che (teorema di Taylor)

$$\left| f'(a) - \frac{f(a+h) - f(a)}{h} \right| \leq \frac{M}{2} h \xrightarrow{h \rightarrow 0} 0$$

- Perché?

Tutta colpa dell'aritmetica finita...

- Assumiamo per semplicità di avere solo un errore di rappresentazione dei valori $f(a)$ ed $f(a + h)$ e che questi errori siano maggiorati entrambi da una costante δ :

$$|fl(y) - y| \leq \delta$$

- In realtà stiamo calcolando

$$\alpha^* = \frac{fl(f(a + h)) - fl(f(a))}{h}$$

- L'errore (assoluto) che in realtà commettiamo è

$$E_a = |\alpha - \alpha^*| = \left| f'(a) - \frac{fl(f(a + h)) - fl(f(a))}{h} \right|$$

Analizziamo l'errore

Ipotesi: $|fl(y) - y| \leq \delta$, $\left| f'(a) - \frac{f(a+h) - f(a)}{h} \right| \leq \frac{M}{2}h$

$$\begin{aligned} E_a &= \left| f'(a) - \frac{fl(f(a+h)) - fl(f(a))}{h} \right| \\ &= \left| f'(a) - \frac{f(a+h) - f(a)}{h} + \frac{f(a+h) - f(a)}{h} - \frac{fl(f(a+h)) - fl(f(a))}{h} \right| \\ &\leq \left| f'(a) - \frac{f(a+h) - f(a)}{h} \right| + \left| \frac{f(a+h) - f(a)}{h} - \frac{fl(f(a+h)) - fl(f(a))}{h} \right| \\ &\leq \frac{M}{2}h + \left| \frac{fl(f(a+h)) - f(a+h)}{h} - \frac{fl(f(a)) - f(a)}{h} \right| \\ &\leq \frac{M}{2}h + \left| \frac{fl(f(a+h)) - f(a+h)}{h} \right| + \left| \frac{fl(f(a)) - f(a)}{h} \right| \\ &\leq \frac{M}{2}h + \frac{2\delta}{h} \end{aligned}$$

NB!!!

$$E_a \simeq \frac{M}{2}h + \frac{2\delta}{h} \xrightarrow{h \rightarrow 0} \infty$$

Analizziamo ancora più in dettaglio

- Chiamiamo $\psi(h)$ la funzione che maggiora l'errore (di fatto è circa uguale)

$$\psi(h) = \frac{M}{2}h + \frac{2\delta}{h}$$

- Il minimo di ψ (dell'errore) si trova per il valore di h che risolve

$$\psi'(h) = 0 \Rightarrow \frac{M}{2} - \frac{2\delta}{h^2} = 0 \Rightarrow \bar{h} = 2\sqrt{\frac{\delta}{M}}$$

- Nel problema test dell'esempio numerico abbiamo $M = 1$ e $\delta \simeq u = 10^{-16}$

$$\bar{h} \simeq 10^{-8}$$

- Ecco la spiegazione del motivo per cui il minimo si trova proprio in prossimità del valore 10^{-8} !

Sistemi lineari

Definizione del problema

Data una matrice $A \in \mathbb{R}^{n \times n}$, detta matrice dei coefficienti, e il termine noto $b \in \mathbb{R}^n$

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{pmatrix}$$

si vuole calcolare il vettore $x \in \mathbb{R}^n$ che soddisfa l'uguaglianza

$$Ax = b$$

In forma esplicita:

$$\left\{ \begin{array}{lcl} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots + a_{1n}x_n & = & b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \dots + a_{2n}x_n & = & b_2 \\ \dots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \dots + a_{nn}x_n & = & b_n \end{array} \right.$$

Risultati di esistenza ed unicità

Definizione

Una matrice $A \in \mathbb{R}^{n \times n}$ si dice nonsingolare (o invertibile) se esiste una matrice X tale che $AX = XA = I$; la matrice X si dice *inversa* di A e si indica come $X = A^{-1}$.

Proposizione

Le seguenti proprietà sono equivalenti

- A è nonsingolare
- $\det(A) \neq 0$;
- le righe e le colonne di A formano un insieme di vettori lineamente indipendenti;
- $Ax = 0$ se e solo se $x = 0$.

Teorema di Rouché- Capelli

Se A è nonsingolare, allora esiste un'unica soluzione del sistema lineare $Ax = b$.

Condizionamento di un sistema lineare

Esempio

$$\begin{cases} x_1 + 2x_2 = 3 \\ .499x_1 + 1.001x_2 = 1.5 \end{cases} \quad \text{Soluzione esatta: } \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Consideriamo due sistemi ottenuti perturbando i dati del problema (matrice dei coefficienti e termine noto).

$$\begin{cases} x_1 + 2x_2 = 3 \\ .5x_1 + 1.002x_2 = 1.5 \end{cases} \quad \begin{cases} x_1 + 2x_2 = 3 \\ .499x_1 + 1.001x_2 = 1.4985 \end{cases}$$

Soluzione esatta: $\begin{pmatrix} 3 \\ 0 \end{pmatrix}$

Soluzione esatta: $\begin{pmatrix} 2 \\ 0.5 \end{pmatrix}$

Ad una piccola (10^{-3}) perturbazione dei dati, corrisponde una grande (10^0) variazione dei risultati.

E' un problema malcondizionato.

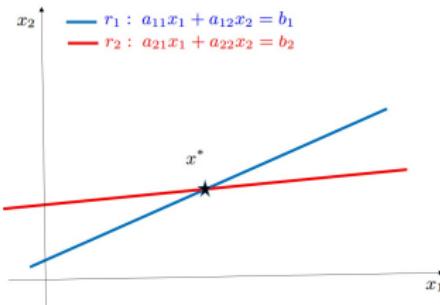
$$\begin{cases} x_1 + 2x_2 = 3 \\ .499x_1 + 1.001x_2 = 1.5 \end{cases}$$

Un sistema lineare è malcondizionato quando le equazioni del sistema - le righe della matrice dei coefficienti - sono 'quasi' linearmente dipendenti.

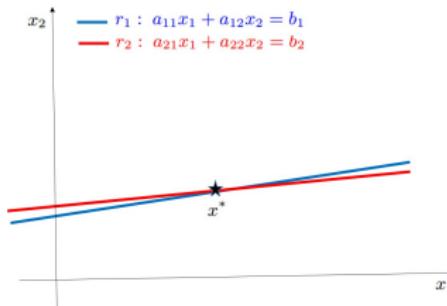
Interpretazione grafica

Sistemi lineari nel piano

Se $n = 2$, la soluzione di un sistema consiste nel punto x^* in cui si intersecano le rette r_1 ed r_2 di equazione $a_{11}x_1 + a_{12}x_2 = b_1$ e $a_{21}x_1 + a_{22}x_2 = b_2$ rispettivamente.



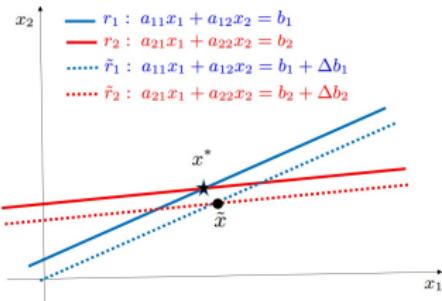
Il caso di malcondizionamento si ha quando r_1 ed r_2 sono ‘quasi’ coincidenti.



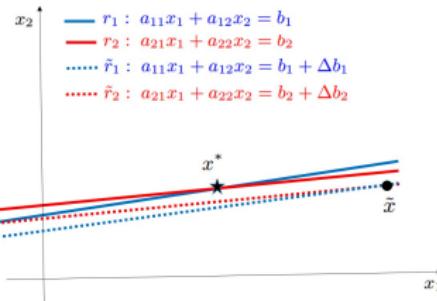
Condizionamento di un sistema lineare

Occorre studiare come variano le soluzioni rispetto a perturbazioni sui dati, che sono A, b . Per semplicità supponiamo di perturbare solo il termine noto

★ : x^* soluzione di $Ax = b$



● : \tilde{x} soluzione di $Ax = b + \Delta b$



- Il determinante della matrice non fornisce un'indicazione utile ad identificare questa situazione (si consideri la matrice $A = \frac{1}{2}I_n$: si ha $\det(A) = \frac{1}{2^n}$, ma non è un caso di malcondizionamento)
- Cerchiamo un parametro adatto a quantificare il condizionamento dei sistemi lineari.

Obiettivo: ricavare una relazione tra l'errore relativo sulle soluzioni e l'errore relativo sui dati

$$\frac{\|x^* - \tilde{x}\|}{\|x^*\|} = \boxed{?} \frac{\|\Delta b\|}{\|b\|}$$

Numero (o indice) di condizionamento di una matrice

Ipotesi: $Ax^* = b$ $A\tilde{x} = b + \Delta b$ ($b \neq 0$)

Osservazione 1:

$$A\tilde{x} = Ax^* + \Delta b \Leftrightarrow A(\tilde{x} - x^*) = \Delta b \Leftrightarrow (\tilde{x} - x^*) = A^{-1}\Delta b$$

Osservazione 2:

$$b = Ax^* \Rightarrow \|b\| = \|Ax^*\| \Rightarrow \|b\| \leq \|A\| \cdot \|x^*\| \Rightarrow \frac{1}{\|x^*\|} \leq \frac{\|A\|}{\|b\|}$$

$$\begin{aligned} \frac{\|\tilde{x} - x^*\|}{\|x^*\|} &= \frac{\|A^{-1}\Delta b\|}{\|x^*\|} \\ &\leq \frac{1}{\|x^*\|} \|A^{-1}\| \cdot \|\Delta b\| \\ &\leq \|A\| \cdot \|A^{-1}\| \frac{\|\Delta b\|}{\|b\|} \end{aligned}$$

Il numero $k(A) = \|A\| \cdot \|A^{-1}\|$ viene detto **numero di condizionamento** della matrice A .

Proprietà del numero di condizionamento

Abbiamo dimostrato

$$\frac{\|\tilde{x} - x^*\|}{\|x^*\|} \leq k(A) \frac{\|\Delta b\|}{\|b\|}$$

ma possiamo pensare che invece del simbolo ' \leq ' ci sia ' \simeq '.

- Il numero di condizionamento di A agisce come un fattore di amplificazione tra errore sui dati ed errore sulle soluzioni.
- Per ogni matrice non singolare A si ha

$$1 = \|I\| = \|AA^{-1}\| \leq \|A\| \cdot \|A^{-1}\| = k(A).$$

Se $k(A) \simeq 1$, allora ogni sistema con A come matrice dei coefficienti è ben condizionato. Se $k(A) \gg 1$, allora il sistema $Ax = b$ è mal condizionato.

Si può provare anche un risultato più generale sul condizionamento dei sistemi, nel caso più realistico in cui siano presenti perturbazioni anche sulla matrice dei coefficienti (slide successiva).

Esempio

$$A = \begin{pmatrix} 1 & 2 \\ 0.499 & 1.001 \end{pmatrix}, b = \begin{pmatrix} 3 \\ 1.5 \end{pmatrix} \quad \det(A) = 0.003$$

$$A^{-1} = \frac{1}{\det(A)} \begin{pmatrix} 1.001 & -2 \\ -0.499 & 1 \end{pmatrix}$$

$$\|A\|_\infty = 3, \|A^{-1}\|_\infty = \frac{1}{\det(A)} \cdot 3.001$$

$$\Rightarrow k_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty = \simeq 3 \cdot 10^3$$

Indichiamo con

$$x^* = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \tilde{x} = \begin{pmatrix} 2 \\ 0.5 \end{pmatrix}$$

Si ha $Ax^* = b$ e $A\tilde{x} = b + \Delta b$ con

$$b + \Delta b = \begin{pmatrix} 3 \\ 1.4985 \end{pmatrix} \Rightarrow \Delta b = \begin{pmatrix} 0 \\ -0.0015 \end{pmatrix}$$

Calcoliamo gli errori:

$$\frac{\|x^* - \tilde{x}\|_\infty}{\|x^*\|_\infty} = 1 \quad \frac{\|\Delta b\|_\infty}{\|b\|_\infty} = 5 \cdot 10^{-4}$$

Il numero di condizionamento $k_\infty(A)$ è il coefficiente di proporzionalità tra i due errori.

Stima dell'errore inerente nella soluzione di un sistema

Supponiamo che x^* sia la soluzione del sistema corrispondente ai dati (A, b) . Indichiamo con \tilde{x} la soluzione del sistema corrispondente ai dati perturbati $(A + \Delta A, b + \Delta b)$. Indichiamo inoltre con e_A ed e_b gli errori relativi presenti nei dati

$$e_A = \frac{\|\Delta A\|}{\|A\|}, \quad e_b = \frac{\|\Delta b\|}{\|b\|}$$

Si può dimostrare (sotto alcune ipotesi) che l'errore relativo inerente

$$e(\tilde{x}) = \frac{\|x^* - \tilde{x}\|}{\|x^*\|}$$

soddisfa la seguente diseguaglianza:

$$e(\tilde{x}) \leq \frac{k(A)}{1 - k(A)e_A} (e_A + e_b).$$

Suggerimento per uno studio efficace...

Per ogni problema considerato:

- Quali sono i dati? Cosa si vuole calcolare?
- Discussione delle proprietà della/delle soluzioni:
 - ① Esistenza e unicità, ipotesi...
 - ② Condizionamento
- Presentazione degli algoritmi, delle loro proprietà
 - ① Complessità computazionale
 - ② Stabilità
 - ③ Confronto tra algoritmi
- Implementazione in Matlab

Sistemi lineari

- Quali sono i dati? Cosa si vuole calcolare?

Dati: $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$. Soluzione: $x \in \mathbb{R}^n$ tale che $Ax = b$

- Discussione delle proprietà della/delle soluzioni:

- ① Esistenza e unicità, ipotesi...

Se A è non singolare, allora esiste un'unica soluzione del sistema.

- ② Condizionamento

Il condizionamento del problema dipende dalla matrice dei coefficienti: se il numero di condizionamento di A è grande, allora il sistema è mal condizionato.

- Presentazione degli algoritmi, delle loro proprietà

- ① Complessità computazionale

- ② Stabilità

- ③ Confronto tra algoritmi

- Implementazione in Matlab

Vedremo alcuni algoritmi appartenenti a due classi di metodi:

- DIRETTI: con un numero finito di operazioni si calcola la soluzione.
- ITERATIVI: si costruisce una successione di vettori che all'infinito si avvicina alla soluzione del sistema

Sistemi qualsiasi

Sistemi con matrice nonsingolare ma senza una particolare struttura

Il teorema di Rouché-Capelli implicitamente suggerisce un algoritmo per il calcolo della soluzione: $x = A^{-1}b$

- ① Calcolo dell'inversa di A , A^{-1}
- ② Calcolo della soluzione come prodotto $A^{-1}b$

E' una buona idea?



No, non è una buona idea, per almeno 2 motivi:

- Complessità computazionale
- Stabilità

Perchè calcolare l'inversa per risolvere un sistema lineare non è una buona idea

Stabilità

Fissiamo $n = 1$ e consideriamo la singola equazione lineare

$$7x = 21$$

la cui soluzione è $x = 3$. Applicare l'algoritmo dell'inversa consiste nei seguenti passi

- ① Calcolare $\frac{1}{7}$
- ② Calcolare $(\frac{1}{7}) \cdot 21$

Supponiamo di utilizzare un'aritmetica finita con $\beta = 10$, $t = 4$ e troncamento.

- $fl(1/7) = fl(0.142857142857143) = 0.1428$
- $fl(fl(1/7) \cdot 21) = fl(2.9988) = 2.998$

Si è commesso un errore dell'ordine di 10^{-4} (sulla quarta cifra significativa), mentre calcolando $fl(21/7) = 3$ si sarebbe ottenuto il risultato esatto.

$$A = \begin{pmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{pmatrix}, \quad \det(A) = d_1 \cdot d_2 \cdot \dots \cdot d_n$$

Sistema in forma esplicita:

$$d_i x_i = b_i, \quad i = 1, \dots, n \Rightarrow x_i = \frac{b_i}{d_i}, \quad i = 1, \dots, n$$

Costo computazionale: n quozienti.

Sistemi facili

Sistemi triangolari: algoritmi di sostituzione all'avanti e all'indietro

Possiamo avere due casi, a seconda che la matrice del sistema sia triangolare inferiore o superiore, ossia $a_{ij} = 0$ per $i > j$ e $a_{ij} = 0$ per $i < j$ rispettivamente.

inferiore

$$R = \begin{pmatrix} r_{11} & & & \\ r_{21} & r_{22} & & \\ & & \ddots & \\ r_{n1} & r_{n2} & \dots & r_{nn} \end{pmatrix}$$

superiore

$$R = \begin{pmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ & r_{22} & \dots & r_{2n} \\ & & \ddots & \\ & & & r_{nn} \end{pmatrix}$$

Sostituzione in avanti

for $j = 1, \dots, n$

$$\mid \quad \underline{x}_j = \frac{b_j - \sum_{i=1}^{j-1} r_{ji} \underline{x}_i}{r_{jj}}$$

Sostituzione all'indietro

for $j = n, \dots, 1$

$$\mid \quad \underline{x}_j = \frac{b_j - \sum_{i=j+1}^n r_{ji} \underline{x}_i}{r_{jj}}$$

Complessità computazionale: al passo j si calcolano j somme, $j - 1$ prodotti e un quoziente.

Somme	Prodotti	Quozienti
$\sum_{j=1}^n j = \frac{n(n+1)}{2} \simeq \mathcal{O}\left(\frac{n^2}{2}\right)$	$\sum_{j=1}^n (j-1) = \frac{n(n-1)}{2} \simeq \mathcal{O}\left(\frac{n^2}{2}\right)$	n

Sistemi qualsiasi

Sistemi con matrice nonsingolare ma senza una particolare struttura

Una possibilità consiste nell'applicare il metodo di Cramer, che è basato sul calcolo di $n + 1$ determinanti di matrici di ordine n .

Il calcolo del determinante di una matrice mediante il teorema di Laplace ha una complessità di $n!$ somme e prodotti.

Complessità del metodo di Cramer

Utilizzando il supercomputer attualmente più veloce, a giugno 2018 l'IBM Summit con una potenza di calcolo di 200 petaFLOPS (petaFLOPS= 10^5 Floating Point Operazions Per Second), risolvere un sistema di dimensione 25 con il metodo di Cramer richiederebbe circa 2 anni e mezzo di tempo.



Metodo di Gauss

Descrizione

$$Ax = b$$

Il metodo di Gauss per la soluzione di un sistema lineare è un algoritmo che si divide in due fasi:

- ① Fase 1, procedimento di eliminazione (o fattorizzazione) di Gauss: si calcola una matrice triangolare inferiore L e una matrice triangolare superiore U tali che

$$A = LU$$

- ② Fase 2: costruzione e soluzione di un sistema triangolare equivalente a quello iniziale. Dall'uguaglianza $A = LU$ si riscrive il sistema come

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

La soluzione del sistema si ottiene in due passi:

- Soluzione del sistema triangolare inferiore $Ly = b$ per sostituzione all'avanti;
- Soluzione del sistema triangolare superiore $Ux = y$ per sostituzione all'indietro.

Fase 1: procedimento di eliminazione/fattorizzazione

Descrizione

- I passi di eliminazione gaussiana sono $n - 1$;
- Ad ogni passo si ottiene una nuova matrice A_k , $k = 1, \dots, n - 1$ partendo da $A_1 = A$;
- Al passo k si ricava una nuova matrice A_{k+1} mediante opportune combinazioni lineari delle righe di A_k , in modo che gli elementi delle colonne con indice da 1 a k di A_{k+1} che si trovano sotto alla diagonale principale siano nulli.

Primo passo di eliminazione

Elemento perno (o pivot): a_{11} .

Ipotesi: $a_{11} \neq 0$.

Moltiplicatori: $m_{i1} = \frac{a_{i1}}{a_{11}}$, $i = 2, \dots, n$.

Combinazioni lineari: riga i con la riga 1 e coefficiente $-m_{i1}$, $i = 2, \dots, n$.

$$A \equiv A_1 = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix} \Rightarrow A_2 = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \dots & a_{2n}^{(2)} \\ 0 & a_{32}^{(2)} & a_{33}^{(2)} & \dots & a_{3n}^{(2)} \\ 0 & \dots & \dots & \dots & \dots \\ 0 & a_{n2}^{(2)} & a_{n3}^{(2)} & \dots & a_{nn}^{(2)} \end{pmatrix}$$

$$a_{ij}^{(2)} = a_{ij} - m_{i1}a_{1j} \quad i, j = 2, \dots, n$$

$$A_2 = \begin{pmatrix} 1 & & & & \\ -m_{21} & 1 & & & \\ -m_{31} & 0 & 1 & & \\ \vdots & & & \ddots & \\ -m_{n1} & 0 & \dots & 0 & 1 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix} = L_1 A_1$$

Secondo passo di eliminazione

Elemento perno (o pivot): $a_{22}^{(2)}$.

Ipotesi: $a_{22}^{(2)} \neq 0$.

Moltiplicatori: $m_{i2} = \frac{a_{i2}^{(2)}}{a_{22}^{(2)}}, i = 3, \dots, n$.

Combinazioni lineari: riga i con la riga 2 e coefficiente $-m_{i2}$, $i = 3, \dots, n$.

$$A_2 = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \dots & a_{2n}^{(2)} \\ 0 & a_{32}^{(2)} & a_{33}^{(2)} & \dots & a_{3n}^{(2)} \\ 0 & \dots & & & \\ 0 & a_{n2}^{(2)} & a_{n3}^{(2)} & \dots & a_{nn}^{(2)} \end{pmatrix} \Rightarrow A_3 = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \dots & a_{2n}^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \dots & a_{3n}^{(3)} \\ 0 & 0 & 0 & \dots & \\ 0 & 0 & a_{n3}^{(3)} & \dots & a_{nn}^{(3)} \end{pmatrix}$$

$$a_{ij}^{(3)} = a_{ij}^{(2)} - m_{i2} a_{2j}^{(2)} \quad i, j = 3, \dots, n$$

$$A_3 = \begin{pmatrix} 1 & & & & \\ 0 & 1 & & & \\ 0 & -m_{32} & 1 & & \\ \vdots & & & \ddots & \\ 0 & -m_{n2} & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \dots & a_{2n}^{(2)} \\ 0 & a_{32}^{(2)} & a_{33}^{(2)} & \dots & a_{3n}^{(2)} \\ \vdots & \dots & & & \\ 0 & a_{n2}^{(2)} & a_{n3}^{(2)} & \dots & a_{nn}^{(2)} \end{pmatrix} = L_2 A_2$$

k-esimo passo di eliminazione

Elemento perno (o pivot): $a_{kk}^{(k)}$.

Ipotesi: $a_{kk}^{(k)} \neq 0$.

Moltiplicatori: $m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}, i = k+1, \dots, n$.

Combinazioni lineari: riga i con la riga k e coefficiente $-m_{ik}$, $i = k+1, \dots, n$.

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}, \quad i, j = k+1, \dots, n$$

$$A_{k+1} = \begin{pmatrix} 1 & & & & & & \\ 0 & 1 & & & & & \\ 0 & 0 & 1 & & & & \\ \vdots & & & & & & \\ & & & 1 & & & \\ & & & -m_{k+1,k} & 1 & & \\ & & & \vdots & & & \\ 0 & 0 & & -m_{n,k} & & 1 & \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1k} & \dots & a_{1n} \\ a_{21}^{(2)} & a_{22}^{(2)} & a_{23}^{(2)} & \dots & a_{2k}^{(2)} & \dots & a_{2n}^{(2)} \\ a_{31}^{(3)} & a_{32}^{(3)} & a_{33}^{(3)} & \dots & a_{3k}^{(3)} & \dots & a_{3n}^{(3)} \\ \ddots & & & & & & \\ a_{kk}^{(k)} & & & & & & \\ a_{nk}^{(k)} & & & & & & \\ a_{nn}^{(k)} & & & & & & \end{pmatrix}$$

$$A_{k+1} = L_k A_k$$

Dopo $n - 1$ passi

$$U = A_n = L_{n-1}A_{n-1} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1k} & \dots & a_{1n} \\ a_{22}^{(2)} & a_{23}^{(2)} & \dots & a_{2k}^{(2)} & \dots & a_{2n}^{(2)} \\ a_{33}^{(3)} & \dots & a_{3k}^{(3)} & \dots & a_{3n}^{(3)} \\ & \ddots & & & & & \\ & & & a_{kk}^{(k)} & \dots & a_{kn}^{(k)} \\ & & & & \ddots & & \\ & & & & & a_{nn}^{(n)} & \end{pmatrix}$$

- U è la matrice triangolare ottenuta dopo $n - 1$ passi di eliminazione.
- Il procedimento di eliminazione equivale ad una successione di prodotti matriciali

$$U = L_{n-1}A_{n-1} = L_{n-1}L_{n-2}A_{n-2} = \dots = L_{n-1}L_{n-2}\dots L_2L_1A$$

- La matrice L_k è detta *k-esima trasformazione elementare di Gauss* ed è definita in funzione dei moltiplicatori del k -esimo passo.
- Il procedimento è ben posto solo se $a_{kk}^{(k)} \neq 0$ per $k = 1, \dots, n - 1$.

Algoritmo di eliminazione Gaussiana

Pseudocodice

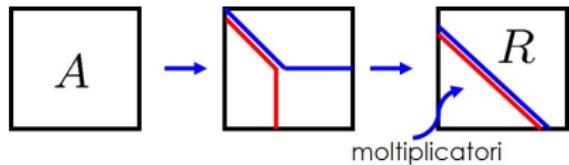
Regole di aggiornamento:

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - a_{kj}^{(k)} \cdot \underbrace{\begin{pmatrix} a_{ik}^{(k)} \\ a_{kk}^{(k)} \end{pmatrix}}_{m_{ik}} \quad k = 1, \dots, n-1, \quad i, j = k+1, \dots, n$$

Osservazione:

Un' implementazione che ottimizza l'occupazione di memoria si ottiene sovrascrivendo gli elementi della matrice aggiornata e i moltiplicatori nelle stesse locazioni inizialmente occupate dagli elementi di A .

```
for  $k = 1, n-1$ 
    for  $i = k+1, n$ 
         $a_{ik} \leftarrow a_{ik}/a_{kk};$ 
        for  $j = k+1, n$ 
             $a_{ij} \leftarrow a_{ij} - a_{ik}a_{kj}$ 
```



Algoritmo di eliminazione Gaussiana

Risultato dell'algoritmo

Dopo gli $n - 1$ passi del metodo, nelle locazioni in cui inizialmente erano memorizzati gli elementi di A si trova

a_{11}	a_{12}	a_{13}	a_{1n}
m_{21}	$a_{22}^{(2)}$	$a_{23}^{(2)}$	$a_{2n}^{(2)}$
m_{31}	m_{32}	$a_{33}^{(3)}$	$a_{3n}^{(3)}$
...					
m_{i1}	m_{i2}	..	$a_{ii}^{(i)}$...	$a_{in}^{(i)}$
...					
$m_{n-1,1}$	$m_{n-1,2}$	$a_{n-1,n-1}^{(n-1)}$	$a_{n-1,n}^{(n-1)}$
m_{n1}	m_{n2}	$m_{n,n-1}$	$a_{nn}^{(n)}$

Algoritmo di eliminazione Gaussiana

Complessità computazionale

```
for k = 1, n - 1
    for i = k + 1, n
        aik ← aik/akk;
        for j = k + 1, n
            aij ← aij - aikakj
```

	Calcolo moltiplicatori	Aggiornamento matrice
Passo	Quozienti	Somme e prodotti
1	n - 1	(n - 1) ²
2	n - 2	(n - 2) ²
...
k	n - k	(n - k) ²
...
n - 1	1	1

Totale matrice:

- somme e prodotti $\sum_{k=1}^{n-1} k^2 = \frac{1}{6}n(n-1)(2n-1) = \mathcal{O}\left(\frac{n^3}{3}\right)$
- quozienti $\sum_{k=1}^{n-1} k = \frac{1}{2}n(n-1) = \mathcal{O}\left(\frac{n^2}{2}\right)$

Fattorizzazione di Gauss

L'algoritmo di eliminazione di Gauss applicato ad una matrice A , sotto l'ipotesi che tutti i perni siano diversi da zero, fornisce una matrice U triangolare superiore e i moltiplicatori m_{ik} , $k = 1, \dots, n-1$, $i = k+1, \dots, n$ che realizzano l'uguaglianza seguente:

$$L_{n-1}L_{n-2}\cdots L_2L_1A = U$$

Osserviamo che tutte le trasformazioni elementari di Gauss sono triangolari con diagonale unitaria: in particolare sono nonsingolari.

$$A = L_1^{-1}L_2^{-1}\cdots L_{n-2}^{-1}L_{n-1}^{-1}U$$

Indichiamo con

$$L = L_1^{-1}L_2^{-1}\cdots L_{n-2}^{-1}L_{n-1}^{-1} \Rightarrow A = LU$$

- L è una matrice triangolare inferiore nonsingolare (è prodotto di matrici triangolari inferiori nonsingolari)
- Dimostriamo che

$$L = \begin{pmatrix} 1 & & & & & & \\ m_{21} & 1 & & & & & \\ m_{31} & m_{32} & 1 & & & & \\ \dots & \dots & \dots & 1 & & & \\ & & & m_{j+1j} & 1 & & \\ \dots & \dots & \dots & \dots & & 1 & \\ m_{n1} & m_{n2} & \dots & m_{nj} & & m_{n,n-1} & 1 \end{pmatrix}$$

Proprietà delle trasformazioni elementari di Gauss

$$L_k = \begin{pmatrix} 1 & & & & & \\ 0 & 1 & & & & \\ \vdots & & & & & \\ & & & 1 & & \\ & & & -m_{k+1,k} & 1 & \\ & & & \vdots & & \\ 0 & 0 & & -m_{n,k} & & 1 \end{pmatrix}$$

- L_k è triangolare con diagonale unitaria $\Rightarrow \det(L_k) = 1 \Rightarrow L_k$ è non singolare
- $L_k = I - m^{(k)} e_k^T,$

$$L_k = \begin{pmatrix} 1 & & & & & \\ 0 & 1 & & & & \\ \vdots & & & & & \\ & & & 1 & & \\ & & & & 1 & \\ 0 & 0 & & & & 1 \end{pmatrix} - \underbrace{\begin{pmatrix} 0 & & & & & \\ \dots & & & & & \\ 0 & & & & & \\ m_{k+1,k} & & & & & \\ \dots & & & & & \\ m_{nk} & & & & & \end{pmatrix}}_{m^{(k)}} \underbrace{(0 \quad \cdots \quad 1 \quad 0 \quad \cdots \quad 0)}_{e_k^T}$$

Inversa della k -esima trasformazione elementare di Gauss

$$L_k^{-1} = I + m^{(k)} e_k^T = \begin{pmatrix} 1 & & & & \\ 0 & 1 & & & \\ 0 & 0 & 1 & & \\ \vdots & & & 1 & \\ & & & m_{k+1,k} & 1 \\ & & & \vdots & \\ 0 & 0 & & m_{n,k} & 1 \end{pmatrix}$$

Dimostrazione.

$$\begin{aligned} (I - m^{(k)} e_k^T)(I + m^{(k)} e_k^T) &= I - m^{(k)} e_k^T + m^{(k)} e_k^T - m^{(k)} \underbrace{e_k^T m^{(k)} e_k^T}_{=0} \\ &= I \end{aligned}$$

Prodotto delle inverse di due trasformazioni elementari di Gauss

$$L_k^{-1} L_j^{-1} = \begin{pmatrix} 1 & & & & & & \\ 0 & 1 & & & & & \\ & m_{k+1k} & 1 & & & & \\ & \dots & 0 & 1 & & & \\ & & & m_{j+1j} & 1 & & \\ 0 & \dots & 0 & \dots & & \ddots & \\ & m_{nk} & & m_{nj} & & & 1 \end{pmatrix}$$

Dimostrazione.

$$\begin{aligned} L_k^{-1} L_j^{-1} &= (I + m^{(k)} e_k^T)(I + m^{(j)} e_j^T) \\ &= I + m^{(k)} e_k^T + m^{(j)} e_j^T + \underbrace{m^{(k)} e_k^T m^{(j)} e_j^T}_{=0} \\ &= I + m^{(k)} e_k^T + m^{(j)} e_j^T \end{aligned}$$

Teorema di fattorizzazione di Gauss

Se $a_{kk}^{(k)} \neq 0, \forall k = 1, \dots, n - 1$, si ha

$$A = LU$$

dove U è una matrice triangolare superiore e L è una matrice triangolare inferiore con diagonale unitaria.

Metodo di Gauss per la soluzione dei sistemi lineari

Complessità computazionale

$$Ax = b$$

$$L \underbrace{Ux}_y = b$$

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

Costo della fattorizzazione: $\mathcal{O}\left(\frac{n^3}{3}\right)$

Costo della soluzione dei due sistemi triangolari: $2\mathcal{O}\left(\frac{n^2}{2}\right)$

Osservazione

Applicare le regole di aggiornamento al termine noto durante il processo di fattorizzazione è equivalente alla soluzione del sistema $Ly = b$.

Osservazione

$$A_k = L_{k-1} \cdot \dots \cdot L_2 L_1 A$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1k} & \dots & a_{1n} \\ a_{22}^{(2)} & a_{23}^{(2)} & \dots & a_{2k}^{(2)} & \dots & a_{2n}^{(2)} \\ a_{33}^{(3)} & \dots & a_{3k}^{(3)} & \dots & a_{3n}^{(3)} \\ \vdots & & & & & & \\ & a_{kk}^{(k)} & \dots & a_{kn}^{(k)} \\ a_{nk}^{(k)} & \dots & a_{nn}^{(k)} \end{pmatrix} = \begin{pmatrix} 1 & & & & & \\ -m_{21} & 1 & & & & \\ -m_{31} & -m_{32} & 1 & & & \\ \vdots & \vdots & \vdots & 1 & & \\ -m_{k1} & & & -m_{k,k-1} & 1 & \\ \vdots & \vdots & \vdots & \vdots & 0 & 1 \\ -m_{n1} & -m_{n2} & \dots & -m_{n,k-1} & 0 & 0 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1k} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2k} & \dots & a_{2n} \\ a_{31} & a_{32} & \dots & a_{3k} & \dots & a_{3n} \\ \vdots & \vdots & \dots & \vdots & \dots & \vdots \\ a_{k1} & a_{k2} & \dots & a_{kk} & \dots & a_{kn} \\ a_{n1} & a_{n2} & \dots & a_{nk} & \dots & a_{nn} \end{pmatrix}$$

Grazie alla struttura triangolare di $L_{k-1} \cdot \dots \cdot L_2 L_1$ si ha anche (sottomatrici)

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1k} \\ a_{22}^{(2)} & a_{23}^{(2)} & \dots & a_{2k}^{(2)} \\ a_{33}^{(3)} & \dots & a_{3k}^{(3)} \\ \vdots & & & \\ a_{kk}^{(k)} \end{pmatrix} = \begin{pmatrix} 1 & & & & & \\ -m_{21} & 1 & & & & \\ -m_{31} & -m_{32} & 1 & & & \\ \vdots & \vdots & \vdots & 1 & & \\ -m_{k1} & & & -m_{k,k-1} & 1 & \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1k} \\ a_{21} & a_{22} & \dots & a_{2k} \\ a_{31} & a_{32} & \dots & a_{3n} \\ \vdots & \vdots & \dots & \vdots \\ a_{k1} & a_{k2} & \dots & a_{kk} \end{pmatrix}$$

Si passa all'uguaglianza dei determinanti

$$a_{11} a_{22}^{(2)} a_{33}^{(3)} \cdots a_{kk}^{(k)} = 1 \cdot A^{(k)}$$

dove $A^{(k)}$ è il k -esimo *minore principale* di A , cioè il determinante della sottomatrice formata dalle sue prime k righe e colonne.

In particolare:

$$\begin{aligned} a_{11} &= A^{(1)} \\ a_{11} \color{blue}{a_{22}^{(2)}} &= A^{(2)} \\ a_{11} \color{blue}{a_{22}^{(2)}} \color{red}{a_{33}^{(3)}} &= A^{(3)} \\ \dots \\ a_{11} \color{blue}{a_{22}^{(2)}} \color{red}{a_{33}^{(3)}} \cdot \dots \cdot \color{magenta}{a_{kk}^{(k)}} &= A^{(k)} \end{aligned}$$

Segue che

$$A^{(i)} \neq 0, i = 1, \dots, k \iff a_{ii}^{(i)} \neq 0, i = 1, \dots, k$$

Condizione necessaria e sufficiente

affinchè tutti i perni siano diversi da zero (e quindi l'algoritmo di Gauss sia ben posto) è che i minori principali di A siano tutti diversi da zero eccetto al più l'ultimo (det A può essere nullo ma l'algoritmo potrebbe essere ugualmente portato a termine).

Teorema di fattorizzazione di Gauss

Enunciato equivalente

Se tutti i minori principali di A sono diversi da zero, tranne al più l'ultimo, ($A^{(k)} \neq 0$, $k = 1, \dots, n - 1$), allora esistono una matrice triangolare inferiore L con diagonale unitaria e una matrice triangolare superiore U tali che

$$A = LU.$$

Inoltre si ha

$$A^{(k)} = u_{11}u_{22} \cdot \dots \cdot u_{kk},$$

per $k = 1, \dots, n - 1$.

Unicità della fattorizzazione di Gauss

Nell'ipotesi che A sia non singolare, allora la fattorizzazione $A = LU$ è unica.

Dimostrazione

Supponiamo che esistano due coppie di matrici con le proprietà enunciate nel teorema tali che

$$A = L_1 U_1 \text{ e } A = L_2 U_2$$

- L_1 ed L_2 sono nonsingolari
- Se A è nonsingolare, allora anche U_1 e U_2 sono nonsingolari

$$L_1 U_1 = L_2 U_2 \Rightarrow L_1^{-1} L_2 = U_1 U_2^{-1}$$

Osserviamo che:

- $L_1^{-1} L_2$ è triangolare inferiore, $U_1 U_2^{-1}$ è triangolare superiore, quindi l'uguaglianza può essere verificata solo se sono entrambe diagonali
- $L_1^{-1} L_2$ ha diagonale unitaria

$$I = L_1^{-1} L_2 = U_1 U_2^{-1} \Rightarrow L_1 = L_2 \quad U_1 = U_2$$

Matrici strettamente a diagonale dominante

Una matrice quadrata A si dice strettamente a diagonale dominante per righe (risp. per colonne) se $\sum_{j=1, j \neq i}^n |a_{ij}| < |a_{ii}|$, $\forall i = 1, \dots, n$ (risp. $\sum_{i=1, i \neq j}^n |a_{ij}| < |a_{jj}|$, $\forall j = 1, \dots, n$).

Per queste matrici vale la proprietà che i minori principali sono diversi da zero.

Fallimento dell'algoritmo

Se l'ipotesi del teorema di fattorizzazione di Gauss non è soddisfatta, l'algoritmo non può essere portato a termine.

Osservazione

Non tutte le matrici invertibili hanno tutti i minori principali diversi da zero.

Esempio:

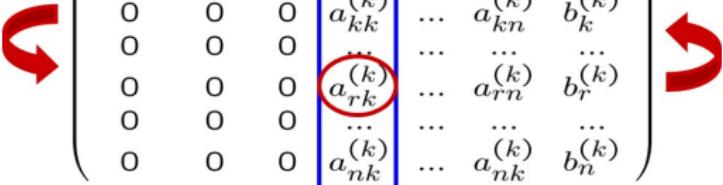
$$A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Introduciamo una modifica dell'algoritmo di Gauss che possa essere applicata a tutte le matrici non singolari.

Strategie di pivoting

- L'idea è quella di introdurre la possibilità di scambiare le righe della matrice *durante* il procedimento di fattorizzazione.
- Se gli stessi scambi vengono applicati anche alle componenti corrispondenti del termine noto, ciò equivale a scambiare due equazioni del sistema $Ax = b$

Al passo k , si scambiano la riga k e la riga r della matrice A_k . Deve essere $r \geq k$ per non distruggere la struttura triangolare che si sta costruendo.

$$\left(\begin{array}{cccccc} a_{11} & \dots & & a_{n1} & b_1 \\ 0 & a_{22}^{(2)} & \dots & a_{2n}^{(2)} & b_2^{(2)} \\ 0 & 0 & \dots & \dots & \dots \\ 0 & 0 & 0 & a_{kk}^{(k)} & \dots & a_{kn}^{(k)} & b_k^{(k)} \\ 0 & 0 & 0 & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & a_{rk}^{(k)} & \dots & a_{rn}^{(k)} & b_r^{(k)} \\ 0 & 0 & 0 & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & a_{nk}^{(k)} & \dots & a_{nk}^{(k)} & b_n^{(k)} \end{array} \right)$$


Dopo lo scambio delle righe, si procede con il calcolo dei moltiplicatori e della corrispondente trasformazione elementare di Gauss.

Matrici di permutazione elementari

Lo scambio di due righe di una matrice può essere espresso come un prodotto matriciale tramite opportune *matrici di permutazione elementari*.

$$P_{ij} = \begin{pmatrix} & i & j \\ i & \left(\begin{array}{cccccc} 1 & & & & & \\ & \dots & & & & \\ & & 1 & & & \\ & & & 0 & & 1 \\ & & & & 1 & \\ & & & & & \dots \\ & & & & & & 1 \\ j & \left(\begin{array}{cccccc} & & & & & \\ & & & & & \\ & & 1 & & & \\ & & & 1 & & 0 \\ & & & & 1 & \\ & & & & & \dots \\ & & & & & & 1 \end{array} \right) \end{pmatrix} \end{array} \right)$$

- P_{ij} è la matrice ottenuta scambiando la riga i e la riga j della matrice identità
- La matrice $P_{ij}A$ si ottiene scambiando la riga i e la riga j di A
- La matrice AP_{ij} si ottiene scambiando la colonna i e la colonna j di A

Proprietà delle matrici di permutazione elementari

- ➊ Sono matrici nonsingolari

$$\det(P_{ij}) = -\det(I) = -1$$

- ➋ Sono matrici simmetriche

$$P_{ij} = P_{ij}^T$$

- ➌ Sono matrici ortogonali

$$P_{ij}P_{ij}^T = P_{ij}P_{ij} = I$$

Un prodotto di matrici di permutazione elementari viene chiamato matrice di permutazione.

$$P = P_{ij}P_{hk} \cdot \dots \cdot P_{uv}$$

La matrice di permutazione P è ortogonale perché prodotto di matrici ortogonali.

$$P^{-1} = P^T = P_{uv}^T \cdot \dots \cdot P_{hk}^T P_{ij}^T = P_{uv} \cdot \dots \cdot P_{hk} P_{ij}$$

Fattorizzazione di Gauss con scambio di righe: primo passo

Ipotesi: A nonsingolare

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & & & \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$$

Dall'ipotesi di nonsingolarità segue che esiste almeno un elemento diverso da zero nella prima colonna di A

- ① Si definisce la matrice di permutazione elementare P_1 che scambia la riga 1 con una *qualsiasi* riga r tale che $a_{r1} \neq 0$
 - Se $a_{11} \neq 0$ è possibile scegliere $P = I$, ossia non effettuare nessuno scambio
- ② Si calcolano i moltiplicatori associati alla prima colonna della matrice permutata $P_1 A$ e di definisce la matrice elementare di Gauss L_1 corrispondente.
- ③ Si aggiornano gli elementi della matrice: $A_2 = L_1 P_1 A$.

Fattorizzazione di Gauss con scambio di righe: secondo passo

Ipotesi: A nonsingolare

$$A_2 = L_1 P_1 A = \begin{pmatrix} a_{11}^{(2)} & a_{12}^{(2)} & \dots & a_{1n}^{(2)} \\ 0 & a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ \dots & & & \\ 0 & a_{n2}^{(2)} & \dots & a_{nn}^{(2)} \end{pmatrix}$$

Definiamo la matrice \tilde{A}_2 come la sottomatrice di A_2 formata dalle sue ultime $n - 1$ righe e colonne

$$\tilde{A}_2 = \begin{pmatrix} a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ \dots & & \\ a_{n2}^{(2)} & \dots & a_{nn}^{(2)} \end{pmatrix}$$

e osserviamo che:

- A_2 è nonsingolare perchè prodotto di matrici nonsingolari (in particolare $\det(A_2) = (-1)^s \det(A)$ dove $s = 0$ se $P_1 = I$ e $s = 1$ altrimenti)
- Per il teorema di Laplace si ha $\det(A_2) = a_{11}^{(2)} \det(\tilde{A}_2)$
- $\det(A_2) \neq 0$ e $a_{11}^{(2)} \neq 0$ implica che $\det(\tilde{A}_2) \neq 0$, ossia \tilde{A}_2 è nonsingolare

Segue che $a_{i2}^{(2)} \neq 0$, per almeno un indice $i \in \{2, \dots, n\}$. Si procede quindi all'eventuale scambio di righe e alla trasformazione elementare di Gauss.

Fattorizzazione di Gauss con scambio di righe: passo k

Ipotesi: A nonsingolare

$$A_k = L_{k-1}P_{k-1} \cdot \dots \cdot L_2P_2L_1P_1A = \begin{pmatrix} a_{11}^{(2)} & & & a_{1n}^{(2)} \\ & \ddots & & \\ & & a_{kk}^{(k)} & \dots & a_{kn}^{(k)} \\ & & \vdots & & \vdots \\ a_{nk}^{(k)} & \dots & & a_{nn}^{(k)} \end{pmatrix}$$

Si definisce

$$\tilde{A}_k = \begin{pmatrix} a_{kk}^{(k)} & \dots & a_{kn}^{(k)} \\ \dots & & \\ a_{nk}^{(k)} & \dots & a_{nn}^{(k)} \end{pmatrix}$$

Si ha $\det(A_k) = a_{11}^{(2)}a_{22}^{(3)} \cdot \dots \cdot a_{k-1k-1}^{(k)} \det(\tilde{A}_k)$. Siccome $\det(A_k) \neq 0$ e $a_{jj}^{(j+1)} \neq 0$ $\forall j = 1, \dots, k-1$, allora \tilde{A}_k è nonsingolare ed esiste almeno un elemento diverso da zero nella sua prima colonna.

Fattorizzazione di Gauss con scambio di righe: dopo $n - 1$ passi

Ipotesi: A nonsingolare

$$A_n = L_{n-1}P_{n-1} \cdot \dots \cdot L_2P_2L_1P_1A = \begin{pmatrix} a_{11}^{(2)} & & & a_{1n}^{(2)} \\ & \ddots & & \\ & & a_{kk}^{(k+1)} & \dots & a_{kn}^{(n+1)} \\ & & & \ddots & \vdots \\ & & & & a_{nn}^{(n)} \end{pmatrix}$$

Teorema di fattorizzazione di Gauss con scambio di righe

Se A è nonsingolare, allora esistono una matrice di permutazione P , una matrice triangolare superiore U , una matrice triangolare inferiore L con diagonale unitaria tali che

$$PA = LU.$$

Osservazione

La matrice U è definita per costruzione, ossia $U = L_{n-1}P_{n-1} \cdot \dots \cdot L_2P_2L_1P_1A$. Si può dimostrare che:

- $P = P_{n-1} \cdot \dots \cdot P_1$;
- $L = \prod_{k=1}^{n-1} P_{n-1} \cdot \dots \cdot P_{k+1}L_k^{-1}$. In altre parole, la k -esima colonna della matrice L contiene i moltiplicatori definiti al passo k permutati secondo gli scambi di righe effettuati nei passi successivi.

Soluzione di un sistema lineare mediante la fattorizzazione di Gauss con scambio di righe

$$Ax = b$$

$$PAx = Pb$$

$$LUx = Pb$$

$$\begin{cases} Ly &= Pb \\ Ux &= y \end{cases}$$

Osservazione: il determinante di A si può ottenere come

$$\det(A) = (-1)^\sigma u_{11} \cdot \dots \cdot u_{nn}$$

dove σ è il numero di permutazioni non banali effettuate.

La fattorizzazione $PA = LU$ non è ovviamente unica:

Ad ogni passo si può scegliere quale scambio di righe effettuare con l'unica condizione di portare in posizione perno un elemento diverso da zero

- Dal punto di vista teorico, qualsiasi elemento perno diverso da zero permette di calcolare i moltiplicatori e quindi di portare a termine l'algoritmo.
- Vediamo invece dal punto di vista numerico quali sono le scelte più opportune

Esempio

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1.0001 & 2 \\ 1 & 2 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}, \text{ soluzione esatta: } \begin{pmatrix} 1 \\ -1.0001\dots \\ 1.0001\dots \end{pmatrix}$$

$$L_1 = \begin{pmatrix} 1 & & \\ -1 & 1 & \\ -1 & 0 & 1 \end{pmatrix} \Rightarrow A_1 = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0.0001 & 1 \\ 0 & 1 & 1 \end{pmatrix} \Rightarrow L_2 = \begin{pmatrix} 1 & & \\ 0 & 1 & \\ 0 & -10000 & 1 \end{pmatrix}$$

$$U = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0.0001 & 1 \\ 0 & 0 & -9999 \end{pmatrix}, L = \begin{pmatrix} 1 & & \\ 1 & 1 & \\ 1 & 10000 & 1 \end{pmatrix}$$

$$Ly = b, \quad \begin{cases} y_1 &= 1 \\ y_1 + y_2 &= 2 \\ y_1 + 10000y_2 + y_3 &= 1 \end{cases} \Rightarrow y = \begin{pmatrix} 1 \\ 1 \\ -10000 \end{pmatrix}$$

Supponiamo ora di utilizzare le regole dell'aritmetica finita con 4 cifre decimali di precisione:

$$Rx = y, \quad \begin{cases} x_1 + x_2 + x_3 &= 1 \\ 0.0001x_2 + x_3 &= 1 \\ -9999x_3 &= -10^4 \end{cases} \Rightarrow \begin{array}{lll} fl(x_3) &=& 1 \text{ invece di } 1.00010001\dots \\ fl(x_2) &=& (1-1)/10^{-4} = 0 \\ fl(x_1) &=& (1-0-1)/1 = 0 \end{array}$$

Commento

Un comportamento instabile

Un piccolo errore nel calcolo di x_3 , dell'ordine di 10^{-4} , si è amplificato in un errore dell'ordine dell'unità nel calcolo delle altre due componenti della soluzione.



Instabilità dell'algoritmo di Gauss per la soluzione dei sistemi

- La causa è la scelta di un perno molto piccolo e quindi, di conseguenza, di un moltiplicatore molto grande.
- Si può utilizzare la libertà di scelta del perno nella fattorizzazione di Gauss con scambi di righe per evitare questa situazione.

Strategia di pivoting parziale

Al passo k , si sceglie come perno l'elemento più grande in valore assoluto della prima colonna della matrice \tilde{A}_k

$$A_k = \begin{pmatrix} a_{11}^{(2)} & & & a_{1n}^{(2)} \\ & \ddots & & \\ & & a_{kk}^{(k)} & \cdots & \cdots & a_{kn}^{(k)} \\ & & \cdots & & & \cdots \\ & & a_{rk}^{(k)} & \cdots & \cdots & a_{rn}^{(k)} \\ & & \cdots & & & \cdots \\ & & a_{nk}^{(k)} & \cdots & \cdots & a_{nn}^{(k)} \end{pmatrix}, \quad |a_{rk}^{(k)}| = \max_{i \in \{k, \dots, n\}} |a_{ik}^{(k)}|$$

Conseguenza del pivoting parziale

$$|m_{ik}| \leq 1, \quad \forall k = 1, \dots, n-1, \forall i = k+1, \dots, n$$

Costo computazionale del pivoting parziale

Al passo k occorre effettuare $n - k + 1$ confronti, che hanno il costo di una differenza

$$\sum_{k=1}^{n-1} k + 1 \simeq \mathcal{O}\left(\frac{n^2}{2}\right)$$

Esempio

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1.0001 & 2 \\ 1 & 2 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}, \text{ soluzione esatta: } \begin{pmatrix} 1 \\ -1.0001... \\ 1.0001... \end{pmatrix}$$

$$P_1 = I, L_1 = \begin{pmatrix} 1 & & \\ -1 & 1 & \\ -1 & 0 & 1 \end{pmatrix} \Rightarrow A_1 = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0.0001 & 1 \\ 0 & 1 & 1 \end{pmatrix} \Rightarrow P_2 A_1 = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0.0001 & 1 \end{pmatrix}$$

$$L_2 = \begin{pmatrix} 1 & & \\ 0 & 1 & \\ 0 & -0.0001 & 1 \end{pmatrix} \Rightarrow U = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0.9999 \end{pmatrix}, L = \begin{pmatrix} 1 & & \\ 1 & 1 & \\ 1 & 0.0001 & 1 \end{pmatrix}$$

$$Ly = Pb, \quad \begin{cases} y_1 &= 1 \\ y_1 + y_2 &= 1 \\ y_1 + 0.0001y_2 + y_3 &= 2 \end{cases} \Rightarrow y = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

Supponiamo ora di utilizzare le regole dell'aritmetica finita con 4 cifre decimali di precisione:

$$Rx = fl(y), \quad \begin{cases} x_1 + x_2 + x_3 &= 1 \\ x_2 + x_3 &= 0 \Rightarrow fl(x_2) &= -1 \\ 0.9999x_3 &= 1 \quad fl(x_1) &= 1 \end{cases} \quad fl(x_3) = 1 \text{ invece di } 1.00010001...$$

Applicando la strategia di pivoting parziale la soluzione calcolata in aritmetica finita contiene un errore dell'ordine di 10^{-4} rispetto al risultato esatto.

In generale:

L'introduzione della strategia di pivoting parziale rende l'algoritmo di soluzione del sistema mediante fattorizzazione di Gauss più stabile.

Strategia di pivoting totale

Al passo k consiste nell'individuare l'elemento in valore assoluto più grande della sottomatrice \tilde{A}_k

$$A_k = \begin{pmatrix} a_{11}^{(2)} & & & a_{1n}^{(2)} \\ \ddots & & & \\ & a_{kk}^{(k)} & \dots & \dots & a_{kn}^{(k)} \\ & \dots & & & \dots \\ & a_{rk}^{(k)} & \dots & \dots & a_{rn}^{(k)} \\ & \dots & & & \dots \\ & a_{nk}^{(k)} & \dots & \dots & a_{nn}^{(k)} \end{pmatrix}, \quad |a_{rs}^{(k)}| = \max_{i,j \in \{k, \dots, n\}} |a_{ij}^{(k)}|$$



$$\left(\begin{array}{cccccc|cc} a_{11} & \dots & & \dots & & a_{n1}^{(2)} & b_1 \\ 0 & a_{22}^{(2)} & & \dots & \dots & a_{2n}^{(2)} & b_2^{(2)} \\ 0 & 0 & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & a_{kk}^{(k)} & \dots & a_{ks}^{(k)} & \dots & a_{kn}^{(k)} & b_k^{(k)} \\ 0 & 0 & 0 & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & a_{rk}^{(k)} & \dots & a_{rs}^{(k)} & \dots & a_{rn}^{(k)} & b_r^{(k)} \\ 0 & 0 & 0 & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & a_{nk}^{(k)} & \dots & a_{ns}^{(k)} & \dots & a_{nk}^{(k)} & b_n^{(k)} \end{array} \right)$$



Scambio di righe e di colonne

Fattorizzazione di Gauss con scambio di righe e di colonne

Se A è non singolare esistono due matrici di permutazione P e Q , una matrice triangolare inferiore L con diagonale unitaria e una matrice triangolare superiore nonsingolare tali che

$$PAQ = LU.$$

Costo computazionale della strategia di pivoting totale

Al passo k occorre confrontare gli elementi della sottomatrice \tilde{A}_k che è di ordine $n - k + 1$

$$\sum_{k=1}^{n-1} (k+1)^2 \simeq \mathcal{O}\left(\frac{n^3}{3}\right)$$

- Il costo del pivoting totale è comparabile a quello della fattorizzazione.
- Nella pratica si utilizza pivoting parziale, che ha un buon rapporto costi-benefici in termini di stabilità.

- Si può dimostrare che se A è strettamente diagonale dominante per righe o per colonne, allora la condizione di pivoting pariale è automaticamente soddisfatta
- Per quel tipo di matrici non è necessario effettuare scambi di righe e si possono evitare i confronti

Criterio generale

La conoscenza delle proprietà del problema permette di ottimizzare l'algoritmo numerico per calcolarne la soluzione, risparmiando

- complessità computazionale
- occupazione di memoria

Varianti della fattorizzazione di Gauss per matrici speciali

- La fattorizzazione di Gauss con pivoting parziale permette la fattorizzazione di qualsiasi matrice nonsingolare
- Si possono ricavare delle varianti dell'algoritmo di Gauss, specifiche per certe classi di matrici, che tengono conto di eventuali proprietà dei dati, al fine di
 - Risparmiare complessità computazionale
 - Risparmiare memoria
- Queste implementazioni ad hoc si basano sui risultati teorici

Fattorizzazione di Gauss per matrici a banda

Matrice a banda r, s : $a_{ij} = 0$ se $j - i > s$ o $i - j > r$ (solo s diagonali secondarie superiori e r inferiori contengono elementi non nulli).

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1s+1} & & \\ \cdots & \cdots & \cdots & a_{2s+2} & \\ a_{r+11} & & & & \ddots \\ & a_{r+22} & & & \\ & & \ddots & & \end{pmatrix}$$

- Sono interessanti perchè molti problemi differenziali provenienti da diverse applicazioni danno luogo a sistemi con questa struttura
- Possono essere memorizzate in forma compatta in $\mathcal{O}(n(s + r))$ locazioni.
- Si può dimostrare che, nel caso in cui i minori principali siano diversi da zero, le matrici della fattorizzazione di Gauss $A = LU$ hanno una analoga struttura

$$L = \begin{pmatrix} 1 & & & \\ \cdots & 1 & & \\ l_{r+11} & \cdots & 1 & \\ & l_{r+22} & \cdots & 1 \end{pmatrix}, U = \begin{pmatrix} r_{11} & \cdots & r_{1s+1} & & \\ \cdots & \cdots & & r_{2s+2} & \\ & \cdots & & \cdots & \\ & & & & r_{nn} \end{pmatrix}$$

- La complessità della fattorizzazione di Gauss è ridotta nel caso di matrici a banda con minori principali non nulli, poiché gli elementi da calcolare sono solo $\mathcal{O}(n(s + r))$
- Questo non è più vero se occorre effettuare scambi di righe.
 - Esempio:

$$A = \begin{pmatrix} \frac{1}{10} & 5 & & & \\ 2 & 100 & 5 & & \\ & 2 & \frac{1}{10} & 5 & \\ & & 2 & \frac{1}{10} & 5 \\ & & & 2 & \frac{1}{10} \\ & & & & 2 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ \frac{1}{20} & 0 & -\frac{1}{8} & \frac{1}{160} & 1 \end{pmatrix}, U = \begin{pmatrix} 2 & 100 & 5 & & \\ & 2 & \frac{1}{10} & 5 & \\ & & 2 & \frac{1}{10} & 5 \\ & & & 2 & \frac{1}{10} \\ & & & & \frac{199}{1600} \end{pmatrix}$$

Con pivoting parziale L e U non sono più a banda.

Matrici sparse

Una matrice quadrata di ordine n si considera *sparsa* se il numero di elementi diversi da zero è una piccola percentuale rispetto ad n^2 .

Esempio:

$$A = \begin{pmatrix} 4 & 1 & 2 & \frac{1}{2} & 2 \\ 1 & \frac{1}{2} & & & \\ 2 & & 3 & & \\ \frac{1}{2} & & & \frac{5}{8} & \\ 2 & & & & 16 \end{pmatrix}$$

Elementi diversi da zero: $3n - 2$. Percentuale di sparsità: $\frac{3n-2}{n^2} \cdot 100 \simeq \frac{3}{n} \cdot 100$.

Memorizzazione di una matrice sparsa

Compressed Column Storage (CCS). Esempio:



Fenomeno di fill-in

Gli elementi L ed U della fattorizzazione di una matrice sparsa non è detto che siano altrettanto sparsi. Esempio:

$$A = \begin{pmatrix} 4 & 1 & 2 & \frac{1}{2} & 2 \\ 1 & \frac{1}{2} & & & \\ 2 & & 3 & & \\ \frac{1}{2} & & & \frac{5}{8} & \\ 2 & & & & 16 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & & & & \\ \frac{1}{2} & 1 & & & \\ \frac{1}{2} & 1 & 1 & & \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{4} & 1 & \\ \frac{1}{4} & -\frac{1}{2} & -\frac{1}{6} & -\frac{2}{5} & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 4 & 1 & 2 & \frac{1}{2} & 2 \\ & -\frac{1}{2} & 2 & -\frac{1}{4} & -1 \\ & & -3 & 0 & 16 \\ & & & \frac{5}{8} & -4 \\ & & & & \frac{1}{15} \end{pmatrix}$$

Esistono tecniche di permutazione, o di *reordering*, delle righe e delle colonne che, applicate a matrici sparse, hanno lo scopo di minimizzare il riempimento - *fill-in* - delle matrici L ed U .

Matrici simmetriche

- Una matrice quadrata A è simmetrica se $A = A^T$.
- La memorizzazione ottimizzata di una matrice simmetrica richiede $\simeq \frac{n^2}{2}$ locazioni di memoria.

Teorema di fattorizzazione di Gauss per matrici simmetriche

Se A è simmetrica e tutti i suoi minori principali sono diversi da zero, allora esistono una matrice L triangolare inferiore con diagonale unitaria e una matrice diagonale D con elementi diagonali diversi da zero tali che

$$A = LDL^T$$

Dimostrazione. Dalle ipotesi si ha $A = LU$ con U non singolare e L con diagonale unitaria. Definiamo

$$D = \begin{pmatrix} u_{11} & & \\ & \ddots & \\ & & u_{nn} \end{pmatrix} \Rightarrow A = LD \underbrace{D^{-1}U}_R$$

La matrice $R = D^{-1}U$ è triangolare superiore e ha diagonale unitaria. Dimostriamo che $R = L^T$. Dalla simmetria di A si ha che

$$LDR = (LDR)^T = R^T DL^T \Rightarrow \underbrace{DRL^{-T}}_{\text{superiore}} = \underbrace{L^{-1}R^TD}_{\text{inferiore}}$$

$\Rightarrow DRL^{-T}$ è diagonale $\Rightarrow RL^{-T}$ è diagonale

Ma RL^{-T} ha diagonale unitaria $\Rightarrow RL^{-T} = I \Rightarrow R = L^T$.

Fattorizzazione di Gauss per matrici simmetriche

- Il teorema di fattorizzazione di Gauss per matrici simmetriche implica che gli elementi da calcolare sono solo quelli della matrice L e della diagonale di D .
- La complessità computazionale della fattorizzazione può essere dimezzata utilizzando un diverso algoritmo per calcolare gli elementi di L e D

Algoritmo di fattorizzazione di Gauss per matrici simmetriche

$A = LDL^T$ Metodo di pavimentazione

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \dots & \dots & \dots & & \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix} = \begin{pmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \dots & & & & \\ l_{n1} & l_{n2} & l_{n3} & \dots & 1 \end{pmatrix} \begin{pmatrix} d_{11} & & & & \\ & d_{22} & & & \\ & & d_{33} & & \\ & & & & d_{nn} \end{pmatrix} \begin{pmatrix} 1 & l_{21} & l_{31} & \dots & l_{n1} \\ & 1 & l_{32} & \dots & l_{n2} \\ & & 1 & \dots & l_{n3} \\ & & & & 1 \end{pmatrix}$$

$$a_{ij} = (l_{i1}, l_{i2}, \dots, l_{ij}, \dots, l_{ii-1}, 1, 0, \dots, 0) \begin{pmatrix} d_{11}l_{j1} \\ d_{22}l_{j2} \\ \vdots \\ d_{j-1j-1}l_{jj-1} \\ d_{jj} \\ 0 \\ \dots \\ 0 \end{pmatrix} = d_{jj}l_{ij} + \sum_{k=1}^{j-1} l_{ik}d_{kk}l_{jk}, \quad \begin{matrix} j = 1, \dots, n \\ i = j, \dots, n \end{matrix}$$

Algoritmo di fattorizzazione di Gauss per matrici simmetriche

$A = LDL^T$ Metodo di pavimentazione

$$a_{ij} = d_{jj}l_{ij} + \sum_{k=1}^{j-1} l_{ik}d_{kk}l_{jk}, \quad j = 1, \dots, n, i = j, \dots, n$$

Primo passo: $j = 1$

$$\begin{aligned} i = 1 &\Rightarrow a_{11} = d_{11} \Rightarrow \textcolor{blue}{d}_{11} = a_{11} \\ i > 1 &\Rightarrow a_{i1} = \textcolor{blue}{d}_{11}l_{i1} \Rightarrow \textcolor{blue}{l}_{i1} = a_{i1}/\textcolor{blue}{d}_{11} \end{aligned}$$

$$\begin{pmatrix} 1 & & & & \\ \textcolor{blue}{l}_{21} & 1 & & & \\ \textcolor{blue}{l}_{31} & l_{32} & 1 & & \\ \dots & & & & \\ \textcolor{blue}{l}_{n1} & l_{n2} & l_{n3} & \dots & 1 \end{pmatrix} \begin{pmatrix} \textcolor{blue}{d}_{11} & & & & \\ & d_{22} & & & \\ & & d_{33} & & \\ & & & & d_{nn} \end{pmatrix} \begin{pmatrix} 1 & \textcolor{blue}{l}_{21} & \textcolor{blue}{l}_{31} & \dots & \textcolor{blue}{l}_{n1} \\ 1 & l_{32} & \dots & l_{n2} \\ 1 & \dots & l_{n3} \\ & & 1 \end{pmatrix}$$

Algoritmo di fattorizzazione di Gauss per matrici simmetriche

$A = LDL^T$ Metodo di pavimentazione

$$a_{ij} = d_{jj}l_{ij} + \sum_{k=1}^{j-1} l_{ik}d_{kk}l_{jk}, \quad j = 1, \dots, n, i = j, \dots, n$$

Secondo passo: $j = 2$

$$i = 2 \Rightarrow a_{22} = d_{22} + d_{11}l_{21}^2 \Rightarrow d_{22} = a_{22} - d_{11}l_{21}^2$$

$$i > 2 \Rightarrow a_{i2} = d_{22}l_{i2} + l_{i1}d_{11}l_{21} \Rightarrow l_{i2} = (a_{i2} - l_{i1}d_{11}l_{21})/d_{22}$$

$$\begin{pmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \dots & & & & \\ l_{n1} & l_{n2} & l_{n3} & \dots & 1 \end{pmatrix} \begin{pmatrix} d_{11} & & & & \\ & d_{22} & & & \\ & & d_{33} & & \\ & & & & d_{nn} \end{pmatrix} \begin{pmatrix} 1 & l_{21} & l_{31} & \dots & l_{n1} \\ & 1 & l_{32} & \dots & l_{n2} \\ & & 1 & \dots & l_{n3} \\ & & & & 1 \end{pmatrix}$$

Algoritmo di fattorizzazione di Gauss per matrici simmetriche

$A = LDL^T$ Metodo di pavimentazione

Al passo j , sono già stati calcolati gli elementi

$$d_{kk}, l_{ik}, \quad k = 1, \dots, j-1, i = k+1, \dots, n$$

Si vogliono calcolare d_{jj} e l_{kj} , $k = j+1, \dots, n$ dalle relazioni

$$a_{ij} = d_{jj}l_{ij} + \sum_{k=1}^{j-1} l_{ik}d_{kk}l_{jk}, \quad j = 1, \dots, n, i = j, \dots, n$$

$$i = j \Rightarrow a_{jj} = d_{jj} + \sum_{k=1}^{j-1} l_{jk}^2 d_{kk} \Rightarrow d_{jj} = a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2 d_{kk}$$

$$i > j \Rightarrow a_{ij} = d_{jj}l_{ij} + \sum_{k=1}^{j-1} l_{ik}d_{kk}l_{jk} \Rightarrow l_{ij} = (a_{ij} - \sum_{k=1}^{j-1} l_{ik}d_{kk}l_{jk})/d_{jj}$$

Pseudocodice:

```
for j = 1, ...n
    |   djj = ajj - sumk=1j-1 ljk2 dkk
    |   for i = j + 1, ..., n
        |       |   lij = (aij - sumk=1j-1 lik dkk ljk) / djj
```

Algoritmo di fattorizzazione di Gauss per matrici simmetriche

$A = LDL^T$ Implementazione e complessità

for $j = 1, \dots, n$

$$d_{jj} = a_{jj} - \sum_{k=1}^{j-1} l_{jk} l_{kj} d_{kk}$$

for $i = j+1, \dots, n$

$$l_{ij} = (a_{ij} - \sum_{k=1}^{j-1} l_{ik} d_{kk} l_{kj}) / d_{jj}$$

for $j = 1, \dots, n$
for $k = 1, \dots, j-1$
| $p_{jk} = l_{jk} d_{kk}$
 $d_{jj} = a_{jj} - \sum_{k=1}^{j-1} l_{jk} p_{jk}$
for $i = j+1, \dots, n$
| $l_{ij} = (a_{ij} - \sum_{k=1}^{j-1} l_{ik} p_{jk}) / d_{jj}$

Complessità computazionale: al passo j si effettuano

$$2(j-1) + (j-1)(n-j) \text{ prodotti}$$

In totale si calcolano

$$\begin{aligned} 2 \sum_{j=1}^n (j-1) + \sum_{j=1}^n (j-1)(n-j) &= \sum_{j=1}^n (j-1) + n \sum_{j=1}^n (j-1) - \sum_{j=1}^n (j-1)^2 \\ &= \frac{n(n-1)}{2} + n \frac{n(n-1)}{2} - \frac{n(n-1)(2n-1)}{6} \\ &= \frac{n(n-1)}{2} + n(n-1) \left(\frac{n}{2} - \frac{2n-1}{6} \right) \\ &= \frac{n(n-1)}{2} + \frac{n(n-1)(n+1)}{6} \Rightarrow \mathcal{O}\left(\frac{n^3}{6}\right) \end{aligned}$$

Soluzione di un sistema tramite fattorizzazione LDL^T

$$Ax = b \Leftrightarrow L \underbrace{DL^T x}_{=z} = b$$

$$\begin{cases} Lz &= b \\ Dy &= z \\ L^T x &= y \end{cases}$$

NB

Il vettore y si calcola con l'algoritmo di soluzione dei sistemi diagonali:

$$y_i = \frac{z_i}{d_{ii}}, i = 1, \dots, n$$

Matrici simmetriche definite positive

Sono matrici simmetriche che godono delle seguenti proprietà caratteristiche:

- $x^T Ax \geq 0 \forall x \in \mathbb{R}^n$ e $x^T Ax = 0 \Leftrightarrow x = 0$
- tutti i minori principali sono positivi
- tutti gli autovalori sono reali positivi

In particolare soddisfano le ipotesi del teorema di fattorizzazione LDL^T . Tuttavia si può ottenere un teorema di fattorizzazione ad hoc.

Teorema di fattorizzazione di Cholesky

Una matrice simmetrica $A \in \mathbb{R}^{n \times n}$ è definita positiva se e solo se esiste una matrice triangolare inferiore \mathcal{L} con elementi diagonali positivi tale che

$$A = \mathcal{L}\mathcal{L}^T$$

Dimostrazione 1: Ipotesi A s.d.p.

$$A = LDL^T \text{ e } x^T LD \underbrace{L^T x}_{=y} > 0 \quad \forall x \neq 0$$

Sia $x \neq 0$ e $y = L^T x$. Siccome L è nonsingolare, $y \neq 0$.

Sostituendo, $y^T Dy > 0 \quad \forall y \neq 0 \Rightarrow D$ è definita positiva, ossia $d_{ii} > 0 \quad \forall i = 1, \dots, n$. La tesi si ottiene ponendo $\mathcal{L} = L\Delta$, dove Δ è la matrice diagonale con elementi $\sqrt{d_{ii}}$.

Dimostrazione 2: Ipotesi $A = \mathcal{L}\mathcal{L}^T$

$$x^T Ax = x^T \mathcal{L} \underbrace{\mathcal{L}^T x}_{=y} = \|y\|_2^2$$

Dalle proprietà delle norme segue che $x^T Ax \geq 0$ per ogni $x \in \mathbb{R}^n$ e $x^T Ax = 0 \iff y = \mathcal{L}^T x = 0$. Siccome \mathcal{L}^T ha diagonale positiva, in particolare è nonsingolare e $\mathcal{L}^T x = 0 \iff x = 0$. Abbiamo quindi dimostrato che A è definita positiva.

Algoritmo di fattorizzazione di Cholesky

$$A = LDL^T = \mathcal{L}\mathcal{L}^T$$

Indichiamo con ℓ_{jk} gli elementi di \mathcal{L} . Dal teorema sappiamo che $\mathcal{L} = L\Delta$, che significa

$$\ell_{jj} = \sqrt{d_{jj}}, \quad \ell_{jk} = l_{jk} \sqrt{d_{kk}}, \quad \forall k = 1, \dots, j-1$$

Utilizziamo le regole di pavimentazione della fattorizzazione LDL^T per ottenere l'algoritmo Cholesky, sostituendo le relazioni precedenti

for $j = 1, \dots, n$

$$\left| \begin{array}{l} d_{jj} = a_{jj} - \sum_{k=1}^{j-1} \ell_{jk}^2 d_{kk} \\ \text{for } i = j+1, \dots, n \end{array} \right.$$

$$\left| \begin{array}{l} l_{ij} = (a_{ij} - \sum_{k=1}^{j-1} l_{ik} d_{kk} l_{jk}) / d_{jj} \end{array} \right.$$

for $j = 1, \dots, n$

$$\Rightarrow \left| \begin{array}{l} \ell_{jj} = \sqrt{a_{jj} - \sum_{k=1}^{j-1} \ell_{jk}^2} \\ \text{for } i = j+1, \dots, n \end{array} \right.$$

$$\left| \begin{array}{l} \ell_{ij} = (a_{ij} - \sum_{k=1}^{j-1} \ell_{ik} \ell_{jk}) / \ell_{jj} \end{array} \right.$$

Complessità computazionale: $\mathcal{O}\left(\frac{n^3}{6}\right)$ a cui si aggiungono n estrazioni di radice quadrata.

Stabilità della fattorizzazione di Cholesky

- Si può dimostrare che, se A è definita positiva, allora gli elementi perno soddisfano automaticamente la condizione di pivoting parziale.
- Si può anche dimostrare che, a differenza dell'algoritmo di Gauss con pivoting parziale o totale, gli elementi della fattorizzazione di Cholesky sono maggiorati da costanti che non dipendono dalla dimensione della matrice (stabilità forte).
- Se la matrice è simmetrica, l'operatore backslash di Matlab tenta per prima cosa di applicare l'algoritmo di Cholesky, finché non trova un radicando negativo, segno che la matrice di partenza non è definita positiva (il teorema di Cholesky caratterizza le matrici definite positive).

Fattorizzazione QR

Teorema di fattorizzazione QR

Sia A una matrice di dimensione n nonsingolare. Allora esistono una matrice ortogonale $Q \in \mathbb{R}^{n \times n}$ e una matrice triangolare superiore nonsingolare $R \in \mathbb{R}^{n \times n}$ tali che

$$A = QR$$

Diamo una dimostrazione costruttiva di questo risultato, che fornisce anche un corrispondente algoritmo per il calcolo di Q ed R .

Calcolo della fattorizzazione QR

Gli algoritmi per il calcolo delle matrici Q ed R sono i seguenti:

- ① mediante trasformazioni elementari di Householder;
- ② mediante trasformazioni elementari di Givens;
- ③ procedimento di Gram-Schmidt;
- ④ per pavimentazione.

I primi due algoritmi sono basati una successione di prodotti matriciali per matrici di trasformazione elementari, come per la fattorizzazione di Gauss. In questo caso, invece di matrici triangolari, le matrici di trasformazione sono ortogonali.

Definizione

Dato un vettore $v \in \mathbb{R}^n$, $v \neq 0$, si definisce trasformazione elementare di Householder associata a v la matrice

$$U = I - \frac{1}{\alpha}vv^T, \text{ dove } \alpha = \frac{1}{2}\|v\|_2^2$$

Proprietà:

- ① **Simmetria:** $U^T = (I - \frac{1}{\alpha}vv^T)^T = I - \frac{1}{\alpha}(vv^T)^T = I - \frac{1}{\alpha}vv^T = U$.
- ② **Ortogonalità:**

$$\begin{aligned} U^T U &= UU \\ &= \left(I - \frac{1}{\alpha}vv^T\right) \left(I - \frac{1}{\alpha}vv^T\right) \\ &= I - \frac{1}{\alpha}vv^T - \frac{1}{\alpha}vv^T + \frac{1}{\alpha^2}v \underbrace{v^Tv}_{\|v\|^2=2\alpha} v^T = I \end{aligned}$$

Complessità computazionale del prodotto matrice-vettore

Sia U la trasformazione elementare di Householder associata al vettore $v \neq 0$ e sia y un vettore di \mathbb{R}^n . Si vuole calcolare

$$z = Uy$$

- Non è necessario calcolare esplicitamente la matrice U , che richiederebbe n^2 prodotti per il termine vv^T
- Applicando le proprietà distributiva e associativa il calcolo si effettua come

$$z = Uy = \left(I - \frac{1}{\alpha} vv^T \right) y = y - \frac{1}{\alpha} v(v^T y)$$

Complessità computazionale: dati v e y ,

		prodotti	somme
α	\leftarrow	$\frac{1}{2} \ v\ ^2$	n
τ	\leftarrow	$(v^T y)/\alpha$	n
w	\leftarrow	τv	n
z	\leftarrow	$y - w$	n

Schema del procedimento di fattorizzazione

L'idea è quella di utilizzare le trasformazioni elementari di Householder, come quelle di Gauss, per eliminare gli elementi del triangolo inferiore della matrice A .

$$U_{n-1} \cdot \dots \cdot U_1 A = R$$

Al passo k , la matrice U_k è definita in modo che nel prodotto vengano eliminati tutti gli elementi della colonna k , sulle righe dalla $k + 1$ -esima fino alla n -esima.

Eliminazione delle componenti di un vettore mediante trasformazioni di Householder

Proprietà

Dato $z \in \mathbb{R}^n$, $z \neq 0$, e definita U come la trasformazione di Householder associata al vettore $v = z + \sigma e_1$, dove e_1 è la prima colonna della matrice identità di ordine n e $\sigma = \|z\|$, si ha che

$$Uz = -\sigma e_1 = \begin{pmatrix} -\sigma \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Dimostrazione. $U = I - \frac{1}{\alpha}vv^T$, dove

$$\begin{aligned}\alpha &= \frac{1}{2}\|v\|^2 = \frac{1}{2}(z + \sigma e_1)^T(z + \sigma e_1) \\ &= \frac{1}{2}(z^T z + 2\sigma z^T e_1 + \sigma^2) = \frac{1}{2}(\sigma^2 + 2\sigma z_1 + \sigma^2) \\ &= \sigma^2 + \sigma z_1\end{aligned}$$

$$\begin{aligned}Uz &= (I - \frac{1}{\alpha}vv^T)z = z - \frac{1}{\alpha}(z + \sigma e_1)(z + \sigma e_1)^T z \\ &= z - \frac{1}{\alpha}(z + \sigma e_1)(z^T z + \sigma e_1^T z) \\ &= z - \frac{1}{\alpha}(z + \sigma e_1) \underbrace{(\sigma^2 + \sigma z_1)}_{=\alpha} \\ &= z - z - \sigma e_1 = -\sigma e_1\end{aligned}$$

Fattorizzazione QR con trasformazioni di Householder

Passo 1

Indichiamo con a_k la k esima colonna di A , che si scrive a blocchi come

$$A = (a_1 \ a_2 \ \dots \ a_n)$$

Definiamo U_1 la trasformazione elementare di Householder associata al vettore $v_1 = a_1 + \sigma_1 e_1$, dove $\sigma_1 = \|a_1\|$ ed e_1 è la prima colonna di I_n . Si ha

$$A_2 = U_1 A = (U_1 a_1 \ U_1 a_2 \ \dots \ U_1 a_n) = \begin{pmatrix} -\sigma_1 & a_{12}^{(2)} & \dots & a_{1n}^{(2)} \\ 0 & a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ \vdots & \dots & & \\ 0 & a_{n2}^{(2)} & \dots & a_{nn}^{(2)} \end{pmatrix} \text{ dove } \begin{pmatrix} a_{1k}^{(2)} \\ a_{2k}^{(2)} \\ \vdots \\ a_{nk}^{(2)} \end{pmatrix} = U_1 a_k.$$

Dimostriamo che

$$\mathbb{R}^{n-1} \ni a_2^{(2)} = \begin{pmatrix} a_{22}^{(2)} \\ \vdots \\ a_{n2}^{(2)} \end{pmatrix} \neq 0$$

Siccome A è nonsingolare, $U_1 a_2 \neq 0$. Se per assurdo assumiamo $a_{12}^{(2)} \neq 0$ e $a_{i2}^{(2)} = 0$ per ogni $i = 2, \dots, n$, ossia $U_1 a_2 = a_{12}^{(2)} e_1$, allora $U_1(a_1 + \frac{\sigma_1}{a_{12}^{(2)}} a_2) = 0 \Leftrightarrow a_1 + \frac{\sigma_1}{a_{12}^{(2)}} a_2 = 0$, che implicherebbe a_1 e a_2 linearmente dipendenti.

Fattorizzazione QR con trasformazioni di Householder

Passo 2

$$A_2 = \begin{pmatrix} -\sigma_1 & a_{12}^{(2)} & a_{23}^{(2)} & \dots & a_{1n}^{(2)} \\ 0 & a_{22}^{(2)} & a_{32}^{(2)} & \dots & a_{2n}^{(2)} \\ 0 & a_{32}^{(2)} & a_{33}^{(2)} & \dots & a_{3n}^{(2)} \\ \vdots & & & & \\ 0 & a_{n2}^{(2)} & a_{n3}^{(2)} & \dots & a_{nn}^{(2)} \end{pmatrix}, \quad a_2^{(2)} = \begin{pmatrix} a_{22}^{(2)} \\ a_{32}^{(2)} \\ \vdots \\ a_{n2}^{(2)} \end{pmatrix} \in \mathbb{R}^{n-1}, \quad U_2 = \begin{pmatrix} 1 & 0 & & & 0 \\ 0 & I_{n-1} - \frac{1}{\alpha_2} v_2 v_2^T & & & \\ \vdots & & & & \\ 0 & & & & \end{pmatrix},$$

dove $v_2 = a_2^{(2)} + \sigma_2 e_1^{(n-1)}$, $\alpha_2 = \frac{1}{2} \|v_2\|^2$, con $\sigma_2 = \|a_2^{(2)}\|$ e $e_1^{(n-1)}$ è la prima colonna di I_{n-1} .

$$A_3 = U_2 A_2 = \begin{pmatrix} -\sigma_1 & a_{12}^{(2)} & a_{13}^{(2)} & \dots & a_{1n}^{(2)} \\ 0 & -\sigma_2 & a_{23}^{(3)} & \dots & a_{2n}^{(3)} \\ 0 & 0 & a_{33}^{(3)} & \dots & a_{3n}^{(3)} \\ \vdots & \vdots & & & \\ 0 & 0 & a_{n3}^{(3)} & \dots & a_{nn}^{(3)} \end{pmatrix}$$

La sottomatrice \tilde{A}_2 formata dalle ultime $n - 1$ righe e colonne di A_2 è nonsingolare (dal teorema di Laplace). Pertanto si possono utilizzare gli stessi argomenti del passo 1 per dimostrare che

$$\mathbb{R}^{n-2} \ni a_3^{(3)} = \begin{pmatrix} a_{33}^{(3)} \\ \vdots \\ a_{n3}^{(3)} \end{pmatrix} \neq 0$$

Fattorizzazione QR con trasformazioni di Householder

Passo k

$$A_k = \begin{pmatrix} -\sigma_1 & a_{12}^{(2)} & & \dots & a_{1n}^{(2)} \\ & \ddots & & & \\ & & \dots & \dots & \\ & & a_{kk}^{(k)} & \dots & a_{kn}^{(k)} \\ & & \vdots & \dots & \dots \\ & & a_{nk}^{(k)} & \dots & a_{nn}^{(k)} \end{pmatrix} U_k = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & I_{n-k+1} - \frac{1}{\alpha_k} v_k v_k^T & & \\ & & & & \end{pmatrix}$$

dove v_k ha dimensione $n - k + 1$ e

$$A_{k+1} = U_k A_k = \begin{pmatrix} -\sigma_1 & a_{12}^{(2)} & a_{13}^{(2)} & & \dots & a_{1n}^{(2)} \\ -\sigma_2 & a_{23}^{(3)} & & \dots & & a_{2n}^{(3)} \\ & \ddots & & \dots & \dots & \\ & & -\sigma_k & a_{kk+1}^{(k+1)} & \dots & a_{kn}^{(k+1)} \\ & & \vdots & & \dots & \\ & 0 & a_{nk+1}^{(k+1)} & \dots & a_{nn}^{(k+1)} & \end{pmatrix}.$$

Fattorizzazione QR con trasformazioni di Householder

Dopo $n - 1$ passi la matrice è in forma triangolare superiore

$$\underbrace{U_{n-1} \cdot \dots \cdot U_1}_{Q^T} A = R$$

- Si dimostra che le matrici U_k sono ortogonali (e simmetriche) $\Rightarrow Q$ è ortogonale
- Moltiplicando entrambi i membri dell'uguaglianza a sinistra per Q si ottiene $A = QR$.
- Per costruzione si ottiene $Q = U_1 U_2 \cdots U_{n-1}$.

Il procedimento di Householder si può applicare anche a matrici non quadrate:

Fattorizzazione di matrici a rango massimo per colonne

Sia $A \in \mathbb{R}^{m \times n}$, con $m \geq n$, tale che le sue colonne sono linearmente indipendenti. Allora esiste una matrice ortogonale Q e una matrice triangolare superiore nonsingolare $R \in \mathbb{R}^{n \times n}$ tale che

$$A = Q \begin{pmatrix} R \\ O \end{pmatrix}$$

dove $O \in \mathbb{R}^{(m-n) \times n}$ ha tutti gli elementi nulli.

Complessità computazionale della fattorizzazione QR con trasformazioni di Householder

Al passo k

$$A_k = \begin{pmatrix} -\sigma_1 & a_{12}^{(2)} & & \dots & a_{1n}^{(2)} \\ & \ddots & & & \\ & & \dots & \dots & \\ & & a_{kk}^{(k)} & \dots & a_{kn}^{(k)} \\ & & \vdots & \dots & \dots \\ & & a_{nk}^{(k)} & \dots & a_{nn}^{(k)} \end{pmatrix} U_k = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & I_{n-k+1} - \frac{1}{\alpha_k} v_k v_k^T & & \\ & & & & \end{pmatrix}$$

	somme	prodotti	radici
$\alpha_k = \ a_k^{(k)}\ ^2$	$n - k + 1$	$n - k + 1$	
$\sigma_k = \sqrt{\alpha_k}$			1
$v_k = a_k^{(k)} + \sigma_k e_1^{(n-k+1)}$	1		
$(I_{n-k+1} - \frac{1}{\alpha_k} v_k v_k^T) a_j^{(k)}$ $j = k+1, \dots, n$	$2(n - k + 1)(n - k)$	$2(n - k + 1)(n - k)$	

Totale: $\mathcal{O}\left(\frac{2}{3}n^3\right)$

Algoritmo di fattorizzazione QR mediante trasformazioni di Householder

Pseudocodice

- Si usa il triangolo superiore di A per gli elementi di R , utilizzando un vettore supplementare $\sigma \in \mathbb{R}^n$ per gli elementi diagonali di R .
- La matrice Q non viene esplicitamente calcolata, ma i vettori v_k e i coefficienti che definiscono le matrici U_k vengono memorizzati nel triangolo inferiore di A e in un vettore supplementare $\alpha \in \mathbb{R}^n$.

$$\begin{array}{lll} a_{ik} & \leftarrow & r_{ik}, \quad i < k \\ a_{ik} & \leftarrow & a_{ik}^{(k)}, \quad i \geq k \\ \sigma_k & \leftarrow & r_{kk} = -\|a_k^{(k)}\| \quad k = 1, \dots, n \\ \alpha_k & \leftarrow & \sigma_k^2 + \sigma_k(a_{kk}^{(k)}) \end{array}$$

- L'occupazione di memoria è quindi di $n^2 + 2n$ locazioni.

for $k = 1, n$

```
|   σk ← √(Σi=kn aik2)
|   akk ← akk + σk
|   αk ← σkakk
|   for j = k + 1, n
|       τ ← (Σi=kn aikaij) / αk
|       for i = k, n
|           aij ← aij - τaik
|   Output: A, α, σ
```

Soluzione di un sistema lineare mediante fattorizzazione QR

$$Ax = b \Leftrightarrow Q \underbrace{Rx}_y = b$$

$$\begin{cases} y &= Q^T b \\ Rx &= y \end{cases}$$

Il vettore $y = Q^T b$, supponendo di aver memorizzato i vettori v_k nel triangolo inferiore di A (vedi slide precedente), si ottiene con il seguente algoritmo:

```
for k = 1, n
    τ ← (Σi=kn biaik) / αk
    for j = k, n
        bj ← bj - τajk
    ⇒ Output: b ← QTb
```

Osservazione: la fattorizzazione QR ha un costo più elevato della LU. Il suo utilizzo è giustificato in casi particolari

- per problemi di approssimazione di dati sperimentali (minimi quadrati)
- quando è necessaria una maggiore stabilità

Confronto della stabilità degli algoritmi di fattorizzazione

- L'algoritmo di soluzione di un sistema triangolare può diventare instabile quando gli elementi della matrice triangolare diventano 'troppo' grandi
- La stabilità delle fattorizzazioni si definisce individuando dei limiti superiori per gli elementi dei fattori
- Si parla di **stabilità forte** se questi limiti non dipendono dalla dimensione della matrice, se invece dipendono dalla dimensione della matrice, si ha **stabilità debole**.

Si può dimostrare che:

Gauss pivoting parziale, $PA = LU$	$ l_{ij} \leq 1, u_{ij} \leq 2^{n-1} \max_{r,s \in \{1, \dots, n\}} a_{rs} $
Cholesky, $A = \mathcal{L}\mathcal{L}^T$	$ \ell_{ij} \leq a_{ii}$
QR	$ q_{ij} \leq 1, r_{ij} \leq \sqrt{n} \max_{k \in \{1, \dots, n\}} a_{ki} $

Gauss con pivoting parziale e QR sono stabili debolmente, ma QR in generale è più stabile di Gauss. Cholesky è fortemente stabile.

Confronto tra algoritmi di fattorizzazione

	ipotesi	complessità	stabilità
Gauss con pivoting parziale	A nonsingolare	$\mathcal{O}(\frac{n^3}{3})$	debole
LDL^T	A simm., minori princ. $\neq 0$	$\mathcal{O}(\frac{n^3}{6})$	debole
Cholesky	A simmetrica, def. pos.	$\mathcal{O}(\frac{n^3}{6})$	forte
QR	colonne di A lin.indip.	$\mathcal{O}(\frac{2n^3}{3})$	debole

Suggerimento per uno studio efficace...

Sistemi lineari

- Quali sono i dati? Cosa si vuole calcolare?

Dati: $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$. Soluzione: $x \in \mathbb{R}^n$ tale che $Ax = b$

- Discussione delle proprietà della/delle soluzioni:

- ① Esistenza e unicità, ipotesi...

Se A è non singolare, allora esiste un'unica soluzione del sistema.

- ② Condizionamento

Il condizionamento del problema dipende dalla matrice dei coefficienti: se il numero di condizionamento di A è grande, allora il sistema è mal condizionato.

- Presentazione degli algoritmi, delle loro proprietà (metodi diretti-fattorizzazioni)

- ① Complessità computazionale

- ② Stabilità

- ③ Confronto tra algoritmi

- Implementazione in Matlab

ipotesi	complessità ①	stabilità ②
3 Gauss con pivoting parziale	$\mathcal{O}(\frac{n^3}{3})$	debole
LDL^T	$\mathcal{O}(\frac{n^3}{6})$	debole
Cholesky	$\mathcal{O}(\frac{n^3}{6})$	forte
QR	$\mathcal{O}(\frac{2n^3}{3})$	debole

Due pacchetti di sottoprogrammi per sistemi lineari

LAPACK

<http://www.netlib.org/lapack/explore-html/index.html>

Harwell Subroutine Library (HSL)

<http://www.hsl.rl.ac.uk/catalogue/>

Metodi iterativi per sistemi lineari

I metodi diretti per la soluzione dei sistemi lineari sono algoritmi in due fasi:

- ① Fattorizzazione della matrice dei coefficienti: gli elementi della fattorizzazione sono matrici semplici (triangolari, diagonali, ortogonali).
- ② Soluzione del sistema lineare: tramite un cambiamento di variabili la soluzione si ottiene risolvendo sistemi semplici associati alle matrici della fattorizzazione.

Metodi iterativi

I metodi iterativi consistono in una formula ricorsiva che permette di generare una successione di vettori $\{x^{(k)}\}_{k \in \mathbb{N}} \subset \mathbb{R}^n$,

$$x^{(k)} = \begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \\ \vdots \\ x_n^{(k)} \end{pmatrix}$$

che, per $k \rightarrow \infty$, converge alla soluzione x^* del problema considerato.

Definizione

La successione $\{x^{(k)}\}_{k \in \mathbb{N}} \subset \mathbb{R}^n$ converge ad un punto $x^* \in \mathbb{R}^n$ se

$$\lim_{k \rightarrow \infty} \|x^{(k)} - x^*\| = 0,$$

per qualche norma vettoriale $\|\cdot\|$. In questo caso si scrive anche

$$\lim_{k \rightarrow \infty} x^{(k)} = x^*$$

Si può verificare che:

$$\lim_{k \rightarrow \infty} \|x^{(k)} - x^*\| = 0 \iff \lim_{k \rightarrow \infty} x_i^{(k)} = x_i^*, \forall i = 1, \dots, n$$

Riformulazione di un sistema lineare

$$Ax = b$$

Data una matrice $M \in \mathbb{R}^{n \times n}$ nonsingolare, il sistema si può riscrivere come

$$Mx = Mx + b - Ax$$

$$x = (I - M^{-1}A)x + M^{-1}b$$

Se definiamo

$$\begin{aligned} G &= I - M^{-1}A \\ c &= M^{-1}b \end{aligned}$$

possiamo riscrivere il sistema lineare in forma equivalente come

$$x = Gx + c$$

Se x^* è la soluzione del sistema, allora

$$Ax^* = b \iff x^* = Gx^* + c$$

Costruzione di una famiglia di metodi iterativi per la soluzione di un sistema lineare

Fissata una matrice M , a partire dalla formulazione equivalente del sistema come

$$x = Gx + c,$$

dove $G = I - M^{-1}A$, si definisce la relazione di ricorrenza

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

che permette di calcolare ogni punto della successione (detto anche *iterata*) in funzione del precedente.

- G è la **matrice di iterazione**, M è la **matrice del metodo**
- Per innescare il procedimento occorre fornire il punto iniziale $x^{(0)} \in \mathbb{R}^n$
- Diverse scelte di M corrispondono a diversi metodi iterativi
- Un metodo iterativo si dice **convergente** se per ogni scelta del punto iniziale $x^{(0)} \in \mathbb{R}^n$ la successione generata converge alla soluzione del sistema.
- La convergenza di un metodo dipende dalla scelta di M

Condizioni per la convergenza di un metodo iterativo (1)

Teorema (condizione sufficiente)

Se $\|G\| < 1$, allora il metodo iterativo

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

è convergente.

Studiamo il vettore $e^{(k)} = x^{(k)} - x^*$: convergenza $\iff \lim_{k \rightarrow \infty} e^{(k)} = 0$

$$\begin{aligned} e^{(k)} &= x^{(k)} - x^* \\ &= Gx^{(k-1)} + c - Gx^* - c \\ &= G(x^{(k-1)} - x^*) = Ge^{(k-1)} = G^2 e^{(k-2)} \\ &= G^k e^{(0)} \quad \text{dove } G^k = \underbrace{G \cdot \dots \cdot G}_{k \text{ volte}} \end{aligned}$$

$$\lim_{k \rightarrow \infty} x^{(k)} - x^* = \lim_{k \rightarrow \infty} e^{(k)} = \lim_{k \rightarrow \infty} G^k e^{(0)}$$

$$\lim_{k \rightarrow \infty} \|x^{(k)} - x^*\| \leq \lim_{k \rightarrow \infty} \|G^k\| \|e^{(0)}\| \leq \lim_{k \rightarrow \infty} \|G\|^k \|e^{(0)}\| \leq \left(\lim_{k \rightarrow \infty} \|G\|^k \right) \|e^{(0)}\|$$

Se $\|G\| < 1$, allora $\lim_{k \rightarrow \infty} \|G\|^k = 0$ e si ha convergenza per ogni punto iniziale.

Condizioni per la convergenza di un metodo iterativo (2)

Dalla dimostrazione, prima di passare alle norme:

$$\lim_{k \rightarrow \infty} x^{(k)} - x^* = \lim_{k \rightarrow \infty} e^{(k)} = \lim_{k \rightarrow \infty} G^k e^{(0)} = (\lim_{k \rightarrow \infty} G^k) e^{(0)}$$

Si può dimostrare che

$$\lim_{k \rightarrow \infty} G^k = 0 \iff \rho(G) < 1$$

dove $\rho(G)$ è il *raggio spettrale* della matrice G , ossia il modulo del suo massimo autovalore:

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

Teorema (condizione necessaria e sufficiente)

Il metodo iterativo $x^{(k+1)} = Gx^{(k)} + c$, $k = 0, 1, 2, \dots$ è convergente se e solo se $\rho(G) < 1$.

Velocità di convergenza

$$\|e^{(k)}\| \simeq \rho(G)^k$$

Quanto più $\rho(G)$ è piccolo, tanto più velocemente $x^{(k)}$ converge ad x^* .

- Per ottenere un algoritmo, che per definizione è costituito da un numero finito di passi, occorre individuare una condizione, detta *criterio di arresto*, verificata la quale si arresta il calcolo delle iterate con la garanzia che l'ultima iterata calcolata approssimi la soluzione del problema entro una certa tolleranza ϵ fissata a priori
- In altre parole, dato ϵ si vorrebbe individuare per quale k si ha

$$\|x^{(k)} - x^*\| \leq \epsilon$$

in corrispondenza del quale arrestare il procedimento

- Occorre stimare l'errore $\|x^{(k)} - x^*\|$ in base a quantità calcolabili.

Stima dell'errore

Differenza tra due iterate successive

Proposizione

Sia $\tau > 0$. Se per un certo k si ha che

$$\|x^{(k+1)} - x^{(k)}\| \leq \tau$$

allora

$$\|x^{(k)} - x^*\| \leq \epsilon \text{ con } \epsilon = \tau \| (G - I)^{-1} \|$$

Ricordando che se x^* è la soluzione del sistema si ha $x^* = Gx^* + c$, osserviamo che

$$\begin{aligned} x^{(k+1)} - x^{(k)} &= Gx^{(k)} + c - x^{(k)} = Gx^{(k)} - Gx^* + x^* - x^{(k)} \\ &= G(x^{(k)} - x^*) - (x^{(k)} - x^*) = (G - I)(x^{(k)} - x^*) \end{aligned}$$

Nelle ipotesi che il metodo sia convergente, si dimostra anche che la matrice $(G - I)$ è nonsingolare, pertanto si ottiene che

$$(x^{(k)} - x^*) = (G - I)^{-1}(x^{(k+1)} - x^{(k)})$$

$$\|x^{(k)} - x^*\| \leq \| (G - I)^{-1} \| \|x^{(k+1)} - x^{(k)}\|$$

Stima dell'errore relativo

Distanza relativa tra due iterate successive

$$\|x^{(k)} - x^*\| \leq \|(G - I)^{-1}\| \|x^{(k+1)} - x^{(k)}\|$$

Assumendo che $x^{(k+1)}$ e x^* siano abbastanza vicini da avere lo stesso ordine di grandezza, allora si può assumere

$$\frac{\|x^{(k)} - x^*\|}{\|x^*\|} \simeq \frac{\|x^{(k)} - x^*\|}{\|x^{(k+1)}\|} \leq \|(G - I)^{-1}\| \frac{\|x^{(k+1)} - x^{(k)}\|}{\|x^{(k+1)}\|}$$

Il coefficiente $\|(G - I)^{-1}\|$ di solito non è noto.

Se non è troppo grande, e si ha

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

allora

$$\frac{\|x^{(k)} - x^*\|}{\|x^*\|} \simeq \tau$$

Stima dell'errore

Calcolo del residuo

Residuo del sistema all'iterata k -esima:

$$r^{(k)} = b - Ax^{(k)}$$

Il residuo è nullo in corrispondenza della soluzione.

Proposizione

Sia $\tau > 0$. Se per un certo k si ha

$$\frac{\|r^{(k)}\|}{\|b\|} \leq \tau, \text{ allora } \frac{\|x^{(k)} - x^*\|}{\|x^*\|} \leq \kappa(A)\tau,$$

dove $\kappa(A)$ è il numero di condizionamento della matrice A .

Si ha che (vedi analisi del condizionamento dei sistemi lineari)

$$\frac{\|x^{(k)} - x^*\|}{\|x^*\|} \leq \kappa(A) \frac{\|r^{(k)}\|}{\|b\|}$$

Se la matrice è ben condizionata ($\kappa(A) \simeq 1$) e

$$\frac{\|r^{(k)}\|}{\|b\|} \leq \tau \text{ allora } \frac{\|x^{(k)} - x^*\|}{\|x^*\|} \simeq \tau$$

Schema di un metodo iterativo per la soluzione di un sistema lineare

- Non viene memorizzata l'intera successione, ma si utilizzano due vettori per l'iterato corrente e il successivo
- Se il problema è malcondizionato, soddisfare una tolleranza bassa potrebbe avere un costo proibitivo in termini di tempo di calcolo. Per questo motivo si introduce un parametro di salvaguardia N_{max} che rappresenta il numero massimo di iterazioni eseguito il quale l'algoritmo si arresta anche se i criteri di arresto non sono soddisfatti.

```
INPUT:  $x^{corr}, A, b, G, c, \tau, N_{max}$ 
for  $k = 0, 1, \dots, N_{max}$ 
     $r \leftarrow b - Ax^{corr}$ 
     $x^{next} \leftarrow Gx^{corr} + c$ 
    if  $\frac{\|x^{next} - x^{corr}\|}{\|x^{next}\|} < \tau$  AND  $\frac{\|r\|}{\|b\|} < \tau$ 
        return  $x^{corr}$ 
    else
         $x^{corr} \leftarrow x^{next}$ 
    endif
    if  $k == N_{max}$ 
        print warning message
    endif
```

- Il costo complessivo di un metodo iterativo dipende dalla tolleranza τ
- A priori è possibile calcolare solo il costo computazionale di una iterazione
- I metodi per la soluzione di sistemi lineari hanno, in generale, un **costo per iterazione** di un prodotto matrice-vettore.

$$\mathcal{O}(n^2)$$

Pertanto i metodi iterativi diventano competitivi con l'approccio di fattorizzazione quando

- L'accuracy con cui si vuole approssimare la soluzione del sistema è abbastanza bassa da richiedere poche iterazioni
- Quando la matrice del sistema (e del metodo) è sparsa o ha una struttura particolare per cui il prodotto matrice-vettore ha complessità molto inferiore ad n^2 .

Scelta della matrice del metodo

$$x^{(k+1)} = Gx^{(k)} + c$$

con

$$\begin{aligned} G &= I - M^{-1}A \\ c &= M^{-1}b \end{aligned}$$

- La scelta ideale, ma non pratica, per avere la soluzione esatta in una sola iterazione sarebbe $M = A$.
- Una scelta ragionevole consiste nel definire M abbastanza simile ad A per avere buone proprietà di convergenza, ma con una struttura ‘semplice’, diagonale o triangolare.

Metodi di decomposizione

Questa tipologia di metodi si basa su una decomposizione di A nella differenza di due matrici, M ed N

$$A = M - N$$

scegliendo M come matrice del metodo.

$$Mx = Mx + b - Ax = Mx + b - (M - N)x$$

$$Mx = Nx + b$$

$$Mx^{(k+1)} = Nx^{(k)} + b$$

- La nuova iterata $x^{(k+1)}$ è la soluzione del sistema $Mv = p$, dove $p = Nx^{(k)} + b$ è il termine noto che dipende dai dati (N, b) e dall'iterato corrente $x^{(k)}$.
- La matrice M deve essere scelta in modo che la soluzione del sistema si possa ottenere a basso costo (non superiore ad n^2).

Metodi di decomposizione

Caso particolare

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & a_{24} & \cdots & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & a_{34} & & & a_{3n} \\ \cdots & \cdots & & \ddots & \cdots & \cdots & \\ a_{i1} & \cdots & & a_{ii-1} & a_{ii} & a_{ii+1} & a_{in} \\ \cdots & \cdots & \cdots & & & \ddots & \\ a_{n-11} & \cdots & \cdots & \cdots & a_{n-1n-2} & a_{n-1n-1} & a_{n-1n} \\ a_{n1} & \cdots & \cdots & \cdots & \cdots & a_{nn-1} & a_{nn} \end{pmatrix}$$

$$E = \begin{cases} e_{ij} = -a_{ij} & i > j \\ 0 & i \geq j \end{cases} \quad F = \begin{cases} f_{ij} = -a_{ij} & i < j \\ 0 & i \leq j \end{cases} \quad D = \begin{cases} d_{ii} = a_{ii} \\ d_{ij} = 0 & i \neq j \end{cases}$$
$$A = D - E - F$$

Da questa decomposizione ricaviamo due metodi iterativi

- Metodo di Jacobi $M = D, N = E + F$
- Metodo di Gauss-Seidel $M = D - E, N = F$

Metodo di Jacobi, $M = D$, $N = E + F$

In forma matriciale:

$$Dx^{(k+1)} = (E + F)x^{(k)} + b$$

Implementazione (per componenti):

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

- Ogni componente della nuova iterata $x^{(k+1)}$ si calcola in funzione solo dell'iterata precedente $x^{(k)}$
- L'ordine con cui si calcolano le componenti della nuova iterata successiva è indifferente: *metodo degli spostamenti simultanei*
- Il metodo si può *formalmente* scrivere come

$$x^{(k+1)} = D^{-1}(E + F)x^{(k)} + D^{-1}b \Rightarrow x^{(k+1)} = \mathcal{J}x^{(k)} + c$$

dove $\mathcal{J} = D^{-1}(E+F) = \begin{pmatrix} 0 & -\frac{a_{12}}{a_{11}} & \dots & -\frac{a_{1n}}{a_{11}} \\ -\frac{a_{21}}{a_{22}} & 0 & \dots & -\frac{a_{2n}}{a_{22}} \\ \dots & \dots & \dots & \dots \\ -\frac{a_{n1}}{a_{nn}} & -\frac{a_{n2}}{a_{nn}} & \dots & 0 \end{pmatrix}$, $c = D^{-1}b = \begin{pmatrix} \frac{b_1}{a_{11}} \\ \vdots \\ \frac{b_n}{a_{nn}} \end{pmatrix}$

Metodo di Gauss-Seidel, $M = D - E$, $N = F$

In forma matriciale:

$$(D - E)x^{(k+1)} = Fx^{(k)} + b$$

Richiede la soluzione di un sistema triangolare inferiore ad ogni passo (sostituzione in avanti). Implementazione (per componenti):

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

- La componente i -esima della nuova iterata $x_i^{(k+1)}$ si calcola in funzione sia dell'iterata precedente $x^{(k)}$ che delle prime $i - 1$ componenti della nuova iterata stessa, già calcolate.
- L'ordine con cui si calcolano le componenti della nuova iterata successiva è sequenziale: *metodo degli spostamenti successivi*
- Il metodo si può *formalmente* scrivere come

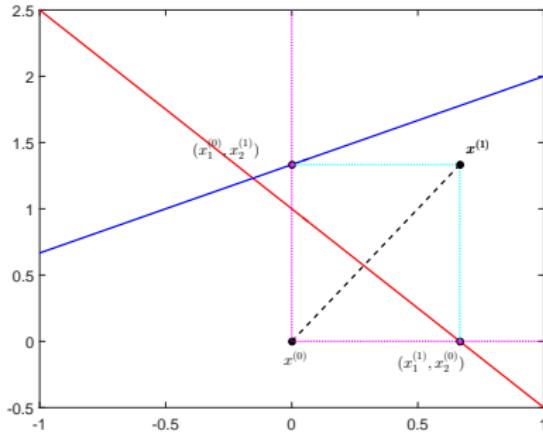
$$x^{(k+1)} = \mathcal{G}x^{(k)} + c \text{ dove } \mathcal{G} = (D - E)^{-1}F, \quad c = (D - E)^{-1}b$$

Interpretazione geometrica del metodo di Jacobi

$$r_1 : a_{11}x_1 + a_{12}x_2 = b_1 \quad \rightarrow \quad r_1 : x_1 = (b_1 - a_{12}x_2)/a_{11}$$

$$r_2 : a_{21}x_1 + a_{22}x_2 = b_2 \quad \rightarrow \quad r_2 : x_2 = (b_2 - a_{21}x_1)/a_{22}$$

$$\text{Jacobi} : \begin{cases} x_1^{(k+1)} = (b_1 - a_{12}x_2^{(k)})/a_{11} & \rightarrow (x_1^{(k+1)}, x_2^{(k)}) \in r_1 \\ x_2^{(k+1)} = (b_2 - a_{21}x_1^{(k)})/a_{22} & \rightarrow (x_1^{(k)}, x_2^{(k+1)}) \in r_2 \end{cases}$$

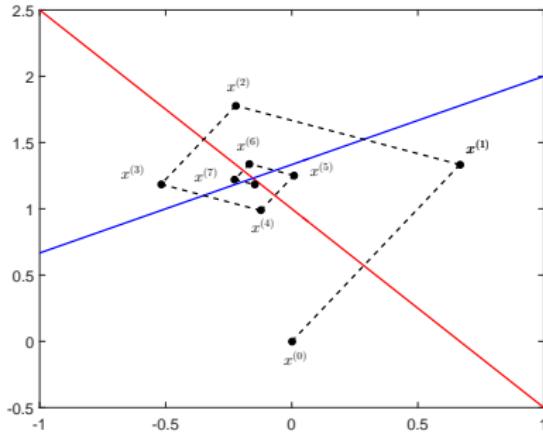


Interpretazione geometrica del metodo di Jacobi

$$r_1 : a_{11}x_1 + a_{12}x_2 = b_1 \quad \rightarrow \quad r_1 : x_1 = (b_1 - a_{12}x_2)/a_{11}$$

$$r_2 : a_{21}x_1 + a_{22}x_2 = b_2 \quad \rightarrow \quad r_2 : x_2 = (b_2 - a_{21}x_1)/a_{22}$$

$$\text{Jacobi} : \begin{cases} x_1^{(k+1)} = (b_1 - a_{12}x_2^{(k)})/a_{11} & \rightarrow (x_1^{(k+1)}, x_2^{(k)}) \in r_1 \\ x_2^{(k+1)} = (b_2 - a_{21}x_1^{(k)})/a_{22} & \rightarrow (x_1^{(k)}, x_2^{(k+1)}) \in r_2 \end{cases}$$

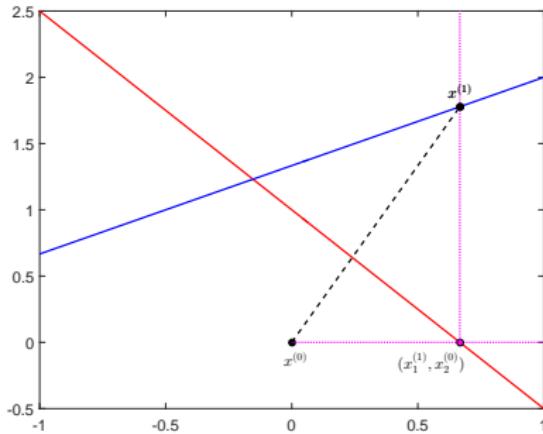


Interpretazione geometrica del metodo di Gauss-Seidel

$$r_1 : a_{11}x_1 + a_{12}x_2 = b_1 \rightarrow r_1 : x_1 = (b_1 - a_{12}x_2)/a_{11}$$

$$r_2 : a_{21}x_1 + a_{22}x_2 = b_2 \rightarrow r_2 : x_2 = (b_2 - a_{21}x_1)/a_{22}$$

Gauss-Seidel : $\begin{cases} x_1^{(k+1)} = (b_1 - a_{12}x_2^{(k)})/a_{11} & \rightarrow (x_1^{(k+1)}, x_2^{(k)}) \in r_1 \\ x_2^{(k+1)} = (b_2 - a_{21}x_1^{(k+1)})/a_{22} & \rightarrow (x_1^{(k+1)}, x_2^{(k+1)}) \in r_2 \end{cases}$

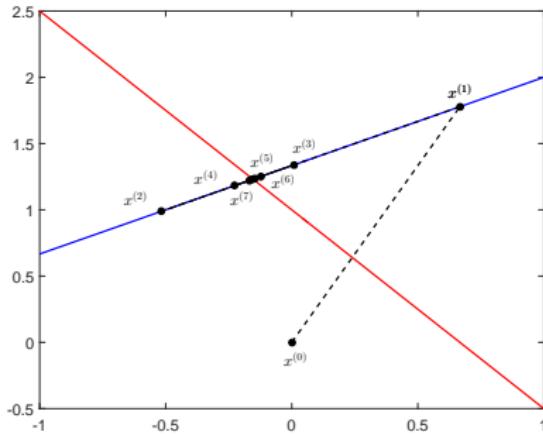


Interpretazione geometrica del metodo di Gauss-Seidel

$$r_1 : a_{11}x_1 + a_{12}x_2 = b_1 \rightarrow r_1 : x_1 = (b_1 - a_{12}x_2)/a_{11}$$

$$r_2 : a_{21}x_1 + a_{22}x_2 = b_2 \rightarrow r_2 : x_2 = (b_2 - a_{21}x_1)/a_{22}$$

Gauss-Seidel : $\begin{cases} x_1^{(k+1)} = (b_1 - a_{12}x_2^{(k)})/a_{11} & \rightarrow (x_1^{(k+1)}, x_2^{(k)}) \in r_1 \\ x_2^{(k+1)} = (b_2 - a_{21}x_1^{(k+1)})/a_{22} & \rightarrow (x_1^{(k+1)}, x_2^{(k+1)}) \in r_2 \end{cases}$



Teorema

Se A è strettamente diagonale dominante per righe o per colonne, allora entrambi i metodi di Jacobi e di Gauss-Seidel convergono.

Teorema di convergenza del metodo di Jacobi

Jacobi : $x^{(k+1)} = \mathcal{J}x^{(k)} + c$ dove

$$\mathcal{J} = D^{-1}(E + F) = \begin{pmatrix} 0 & -\frac{a_{12}}{a_{11}} & \dots & -\frac{a_{1n}}{a_{11}} \\ -\frac{a_{21}}{a_{22}} & 0 & \dots & -\frac{a_{2n}}{a_{22}} \\ \dots & \dots & \dots & \dots \\ -\frac{a_{n1}}{a_{nn}} & -\frac{a_{n2}}{a_{nn}} & \dots & 0 \end{pmatrix}, c = D^{-1}b = \begin{pmatrix} \frac{b_1}{a_{11}} \\ \vdots \\ \frac{b_n}{a_{nn}} \end{pmatrix}$$

Mostriamo che $\|\mathcal{J}\| < 1$

Per righe:

Ipotesi $ a_{ii} > \sum_{j=1, j \neq i}^n a_{ij} $ $\forall i = 1, \dots, n$	Tesi $\ \mathcal{J}\ _1 < 1$	Ipotesi $ a_{ii} > \sum_{j=1, j \neq i}^n a_{ji} $ $\forall i = 1, \dots, n$	Tesi $\ \mathcal{J}\ _0 < 1$
---	--	---	--

$$\|\mathcal{J}\|_1 = \max_{i \in \{1, \dots, n\}} \sum_{j=1, j \neq i}^n \frac{|a_{ij}|}{|a_{ii}|} < 1$$

$$\|\mathcal{J}\|_0 = \max_{i \in \{1, \dots, n\}} \sum_{j=1, j \neq i}^n \frac{|a_{ji}|}{|a_{ii}|} < 1$$

Teorema di convergenza del metodo di Gauss-Seidel

Gauss-Seidel: $x^{(k+1)} = \mathcal{G}x^{(k)} + c$ dove

$$\mathcal{G} = (D - E)^{-1}F$$

Ipotesi	Tesi
$ a_{ii} > \sum_{j=1, j \neq i}^n a_{ij} $ $\forall i = 1, \dots, n$	$\rho(\mathcal{G}) < 1$

Gauss-Seidel: sia λ un autovalore di \mathcal{G} e v il corrispondente autovettore

$$\mathcal{G}v = \lambda v \iff Fv = \lambda(D - E)v \iff -\sum_{j=i+1}^n a_{ij}v_j = \lambda \sum_{j=1}^{i-1} a_{ij}v_j + \lambda a_{ii}v_i, \forall i = 1, \dots, n$$

In particolare vale per $i = \ell$, dove ℓ è l'indice per cui $|v_\ell| = \max_{j \in \{1, \dots, n\}} |v_j|$.

$$\begin{aligned} \lambda = -\lambda \sum_{j=1}^{\ell-1} \frac{a_{\ell j}}{a_{\ell\ell}} \frac{v_j}{v_\ell} - \sum_{j=\ell+1}^n \frac{a_{\ell j}}{a_{\ell\ell}} \frac{v_j}{v_\ell} \Rightarrow |\lambda| &\leq |\lambda| \sum_{j=1}^{\ell-1} \frac{|a_{\ell j}|}{|a_{\ell\ell}|} \frac{|v_j|}{|v_\ell|} + \sum_{j=\ell+1}^n \frac{|a_{\ell j}|}{|a_{\ell\ell}|} \frac{|v_j|}{|v_\ell|} \\ &\leq |\lambda| \sum_{j=1}^{\ell-1} \frac{|a_{\ell j}|}{|a_{\ell\ell}|} + \sum_{j=\ell+1}^n \frac{|a_{\ell j}|}{|a_{\ell\ell}|} \end{aligned}$$

Se per assurdo supponiamo $|\lambda| \geq 1$, allora si ottiene

$$|\lambda| \leq |\lambda| \sum_{j=1}^{\ell-1} \frac{|a_{\ell j}|}{|a_{\ell\ell}|} + |\lambda| \sum_{j=\ell+1}^n \frac{|a_{\ell j}|}{|a_{\ell\ell}|} = |\lambda| \sum_{j=1, j \neq \ell}^n \frac{|a_{\ell j}|}{|a_{\ell\ell}|} \Rightarrow \sum_{j=1, j \neq \ell}^n |a_{\ell j}| \geq |a_{\ell\ell}|$$

Il caso per colonne si dimostra applicando gli stessi argomenti ad A^T .

- Una singola iterazione per entrambi i metodi costa, in generale, n^2 operazioni
- Se la matrice A è sparsa, la complessità è proporzionale al numero di elementi non nulli
- La complessità dell'intero metodo dipende dalla tolleranza e dalla velocità con cui le iterate si avvicinano alla soluzione.

Velocità di convergenza

- Quanto più è piccolo il raggio spettrale della matrice del metodo, tanto più veloce la successione converge.
- Si può dimostrare che $\rho(\mathcal{G}) \leq \rho(\mathcal{J})$ (il metodo di Gauss-Seidel è non meno veloce di Jacobi)
- Per migliorare la velocità di convergenza (rendere il raggio spettrale più piccolo) si può introdurre un parametro

Esempio: metodo Successive Over Relaxation (SOR)

$$\omega A = \underbrace{(D - \omega E)}_M - \underbrace{((1 - \omega)D + \omega F)}_N$$

Converge se A è simmetrica definita positiva e $0 < \omega < 2$.

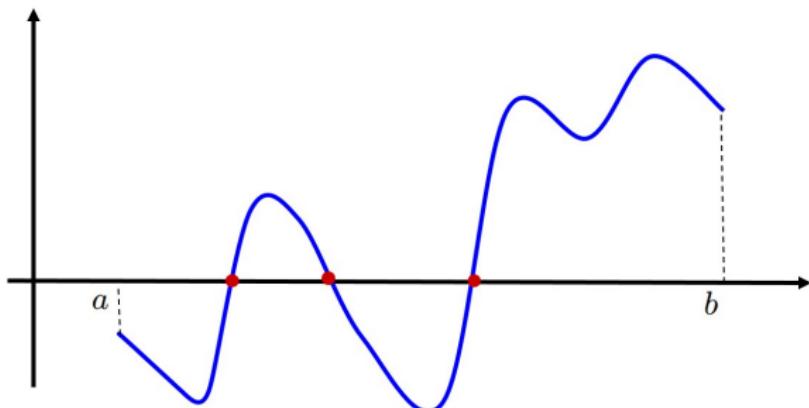
Metodi per la soluzione di equazioni nonlineari

Definizione del problema

Definizione

Sia f una funzione definita nell'intervallo $[a, b]$ a valori reali. Il punto $x_* \in [a, b]$ si dice *radice* (o *zero*) della funzione f se

$$f(x_*) = 0$$



Teorema del valor medio

Sia $f : [a, b] \rightarrow \mathbb{R}$ una funzione continua tale che

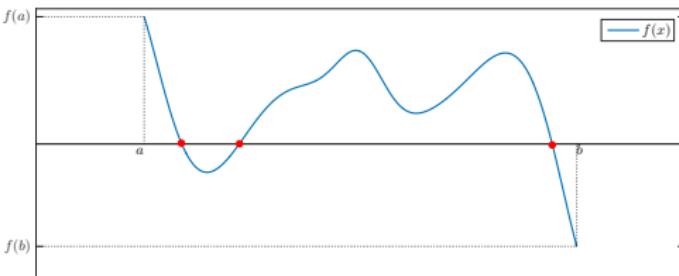
$$f(a) \cdot f(b) < 0.$$

Allora esiste almeno una radice di f nell'intervallo $[a, b]$, ossia esiste un punto $x_* \in [a, b]$ tale che

$$f(x_*) = 0$$

NB:

Le ipotesi del teorema precedente non garantiscono l'unicità della soluzione. Una condizione sufficiente per avere unicità è, per esempio, la stretta monotonia della funzione.



Condizionamento del problema della ricerca degli zeri di funzione

Sia x_* una radice di f .

Consideriamo un punto \tilde{x} tale che $|f(\tilde{x})| = \delta$, con δ “piccolo” (\tilde{x} è la soluzione del problema perturbato $f(x) = \pm\delta$).

Sotto quali condizioni possiamo concludere che $|x_ - \tilde{x}|$ è altrettanto “piccolo”?*

- Assumiamo per semplicità che la funzione f di cui vogliamo calcolare le radici sia differenziabile, per definizione si ha

$$f'(x_*) = \lim_{x \rightarrow x_*} \frac{f(x) - f(x_*)}{x - x_*}$$

da cui, in un intorno di x_* si ricava l'approssimazione

$$\frac{f(x) - f(x_*)}{(x - x_*)} \simeq f'(x_*)$$

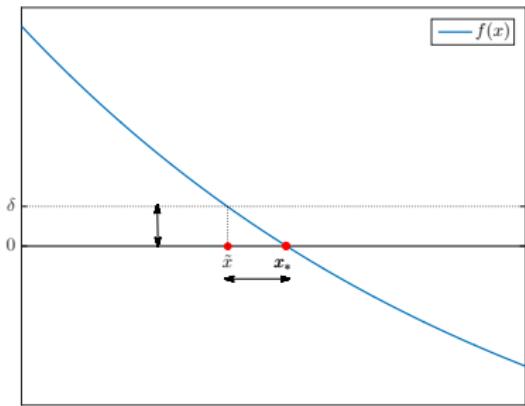
- Per $x = \tilde{x}$ possiamo scrivere

$$|x_* - \tilde{x}| \simeq \frac{|f(x_*) - f(\tilde{x})|}{|f'(x_*)|} = \frac{\delta}{|f'(x_*)|}$$

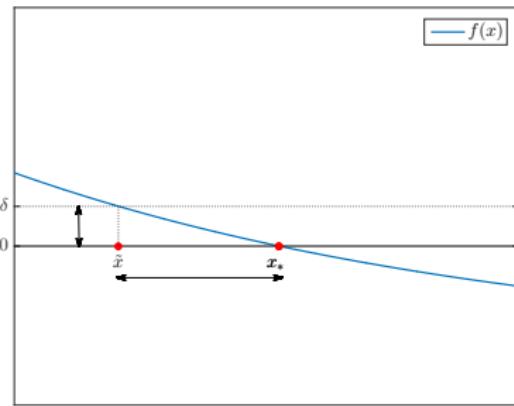
Conclusioni: se $|f'(x_*)| \simeq 0$, allora anche se δ è piccolo, la soluzione del problema perturbato \tilde{x} potrebbe essere distante dalla radice x_* , cioè $|\tilde{x} - x_*|$ è grande.

Interpretazione grafica del condizionamento del problema

Problema ben condizionato



Problema mal condizionato



Il mal condizionamento si ha quando il grafico di f è molto “schiacciato” sull’asse orizzontale.

$$f(x) = 0$$

- Condizioni di esistenza delle soluzioni: [Teorema del valor medio](#)
- Unicità della soluzione: [stretta monotonia \(condizione sufficiente\)](#)
- Condizionamento del problema: [se \$f'\(x_*\) \simeq 0\$ si ha mal condizionamento.](#)
- Metodi numerici (iterativi):
 - Metodo di bisezione
 - Metodo di regula falsi
 - Metodo di Newton (o delle tangenti)
 - Metodo delle secanti
- Analisi di convergenza dei metodi
- Definizione di complessità computazionale
- Definizione di ordine di convergenza

Metodo di bisezione

Dati

Intervallo di ricerca $[a, b]$,

Ipotesi

Funzione continua che assume segno discorde agli estremi dell'intervallo:
 $f(a) \cdot f(b) < 0$

Descrizione

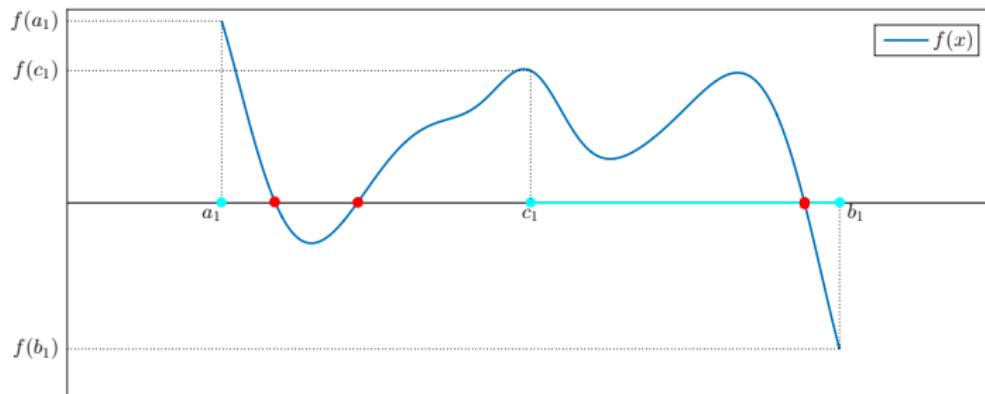
Si applica ripetutamente il teorema del valor medio, generando una successione di intervalli di ampiezza decrescente i cui punti medi convergono ad una radice di f

Metodo di bisezione

Passo 1

- Si calcola il punto medio c_1 dell'intervallo di ricerca $[a, b] \equiv [a_1, b_1]$;
- Si definisce il nuovo intervallo di ricerca $[a_2, b_2]$ come quello tra i due sottointervalli $[a_1, c_1], [c_1, b_1]$ in cui sono soddisfatte le ipotesi del teorema del valor medio:

$$[a_2, b_2] = \begin{cases} [a_1, c_1] & \text{se } f(c_1) \cdot f(a_1) < 0 \\ [c_1, b_1] & \text{se } f(c_1) \cdot f(b_1) < 0 \end{cases}$$

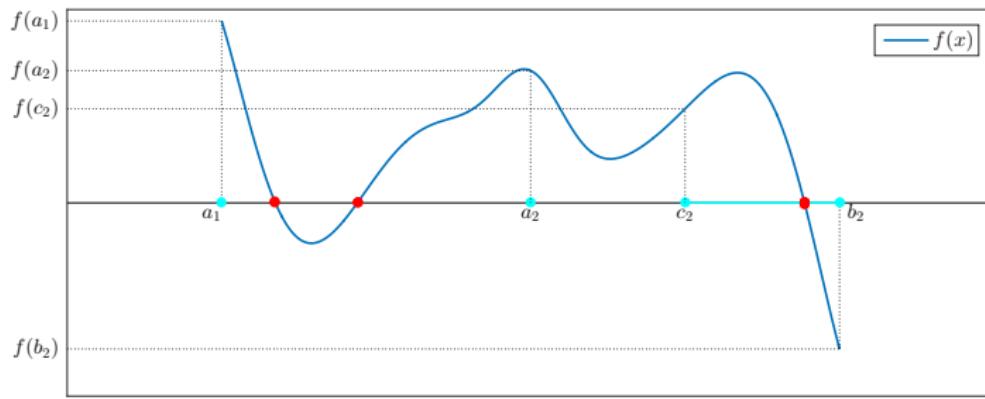


Metodo di bisezione

Passo 2

- Si calcola il punto medio c_2 dell'intervallo di ricerca $[a_2, b_2]$
- Si definisce il nuovo intervallo di ricerca $[a_3, b_3]$ come quello tra i due sottointervalli $[a_2, c_2]$, $[c_2, b_2]$ in cui sono soddisfatte le ipotesi del teorema del valor medio:

$$[a_3, b_3] = \begin{cases} [a_2, c_2] & \text{se } f(c_2) \cdot f(a_2) < 0 \\ [c_2, b_2] & \text{se } f(c_2) \cdot f(b_2) < 0 \end{cases}$$



Metodo di bisezione

Passo k

- Si calcola il punto medio c_k dell'intervallo di ricerca $[a_k, b_k]$
- Si definisce il nuovo intervallo di ricerca $[a_{k+1}, b_{k+1}]$ come quello tra i due sottointervalli $[a_k, c_k]$, $[c_k, b_k]$ in cui sono soddisfatte le ipotesi del teorema del valor medio:

$$[a_{k+1}, b_{k+1}] = \begin{cases} [a_k, c_k] & \text{se } f(c_k) \cdot f(a_k) < 0 \\ [c_k, b_k] & \text{se } f(c_k) \cdot f(b_k) < 0 \end{cases}$$

Proprietà degli intervalli di ricerca

Per ogni k , le ipotesi del teorema del valor medio sono verificate in $[a_k, b_k]$, quindi esiste almeno una radice di f in $[a_k, b_k]$. L'ampiezza del k -esimo intervallo di ricerca è data da

$$b_k - a_k = \frac{b - a}{2^{k-1}}.$$

Convergenza del metodo di bisezione

Teorema

La successione dei punti medi $\{c_k\}_{k \in \mathbb{N}}$ generata dal metodo di bisezione converge ad una radice di f nell'intervallo $[a, b]$ e soddisfa la diseguaglianza

$$|c_k - x_*| \leq \frac{b-a}{2^k}.$$

Segue che

$$|c_k - x_*| = \mathcal{O}\left(\frac{1}{2^k}\right).$$

Inoltre, fissata una tolleranza τ si ricava il numero di passi sufficiente per ottenere un'approssimazione di una radice di f entro la tolleranza τ :

$$\frac{b-a}{2^k} \leq \tau \Leftrightarrow k \geq \log_2 \frac{b-a}{\tau}$$

Pertanto, per ogni $k \geq \log_2 \frac{b-a}{\tau}$ si ha

$$|c_k - x_*| \leq \tau$$

ossia c_k approssima una radice di f entro la tolleranza τ .

Formula stabile per il calcolo del punto medio

Per calcolare il punto medio di un segmento si hanno due possibilità:

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

Operando in aritmetica finita, quando a e b sono vicini il primo algoritmo è meno stabile. Esempio: $a = 0.983$, $b = 0.986$, aritmetica decimale on 3 cifre di precisione e troncamento.

$$\begin{aligned} fl(a+b) &= fl(1.969) & fl(b-a) &= 0.3 \cdot 10^{-2} \\ &= 0.196 \cdot 10^1 & fl\left(\frac{fl(b-a)}{2}\right) &= 0.15 \cdot 10^{-2} \\ fl\left(\frac{fl(a+b)}{2}\right) &= 0.98 & fl\left(a + fl\left(\frac{fl(a+b)}{2}\right)\right) &= fl(0.983 + 0.0015) \\ &&&= fl(0.9845) \\ &&&= 0.984 \end{aligned}$$

Il punto medio calcolato è esterno all'intervallo.

Implementazione del metodo di bisezione

Si assume di avere a disposizione un metodo per calcolare il valore della funzione f

INPUT: a, b, f, τ

```
N    ←  parte intera ( $\log_2((b - a)/\tau)$ )
fa  ←  f(a)
fb  ←  f(b)
for k = 1, 2, ..., N
    c    ←  a + (b - a)/2
    fc  ←  f(c)
    If fc = 0
        return
    Endif
    If fc · fb < 0
        a    ←  c
        fa  ←  fc
    Else
        b    ←  c
        fb  ←  fc
    Endif
```

NB:

Si misura in *numero di valutazioni di funzione per iterazione*, ossia quante volte, nella singola iterazione di un metodo, viene calcolata una funzione nonlineare.

- Si assume che il calcolo (approssimato!) di una qualsiasi funzione nonlineare (trigonometrica, esponenziale,...) si ottiene con algoritmo numerico basato su una successione di operazioni aritmetiche fondamentali.
- Ogni valutazione di funzione equivale ad un ‘pacchetto’ di operazioni fondamentali.
- Il costo computazionale di una iterazione si ottiene contando i ‘pacchetti’ invece delle singole operazioni.

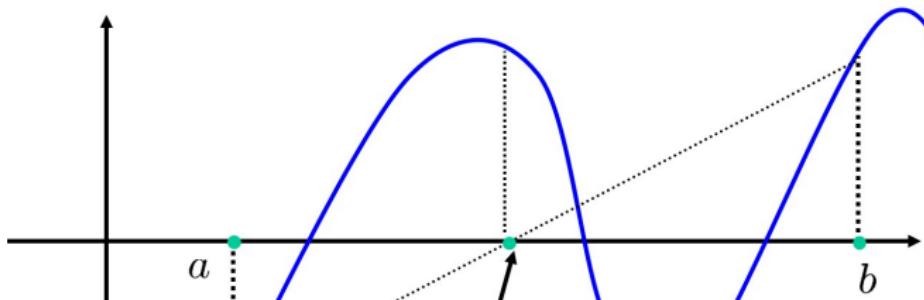
Complessità del metodo di bisezione

Una valutazione di funzione per iterazione.

Variante del metodo di bisezione: il metodo di Regula Falsi

Invece di scegliere il punto medio, si definisce c_k come il punto di intersezione tra l'asse delle ascisse e la retta che passa per i punti del piano $(a_k, f(a_k)), (b_k, f(b_k))$

$$c_k = b_k - \frac{f(b_k)}{\frac{f(b_k) - f(a_k)}{b_k - a_k}}$$



$$c = b - \frac{f(b)}{\frac{f(b) - f(a)}{b - a}}$$

Ha proprietà simili al metodo di bisezione. I metodi di bisezione e regula falsi si dicono metodi *dicotomici*.

Velocità di convergenza

Si può assumere che, per ogni k , valga l'approssimazione

$$|c_k - x_*| \simeq \frac{b - a}{2^k}$$

$$|c_{k+1} - x_*| \simeq \frac{b - a}{2^{k+1}} = \frac{1}{2} |c_k - x_*|$$

- L'errore commesso si dimezza ad ogni iterazione.
- Si guadagna meno di una cifra decimale di precisione ogni 3 iterazioni.

Per valutare il guadagno che si ottiene in termini di riduzione dell'errore ad ogni iterazione rispetto alla precedente si introduce la seguente definizione.

Ordine di convergenza di una successione

Definizione

Sia $\{x_k\}_{k \in \mathbb{N}} \subset \mathbb{R}$ una successione che converge ad un punto $x_* \in \mathbb{R}$. Si dice che la successione $\{x_k\}_{k \in \mathbb{N}}$ ha *ordine di convergenza* p se

$$\lim_{k \rightarrow \infty} \frac{|x_{k+1} - x_*|}{|x_k - x_*|^p} = C$$

per qualche $p \geq 1, C \in \mathbb{R}$.

- Se la successione è generata da un metodo iterativo, si dice che il metodo è *di ordine p*.
- Se una successione (o un metodo iterativo) è di ordine p , applicando la definizione di limite, per k grande si ha

$$|x_{k+1} - x_*| \leq |x_k - x_*|^p$$

- Più p è grande, maggiore sarà la riduzione dell'errore ad una iterazione rispetto alla precedente.

- Convergenza quadratica $p = 2$

$$\lim_{k \rightarrow \infty} \frac{|x_{k+1} - x_*|}{|x_k - x_*|^2} = C$$

- Convergenza lineare $p = 1, C \in (0, 1)$

$$\lim_{k \rightarrow \infty} \frac{|x_{k+1} - x_*|}{|x_k - x_*|} = C$$

- Convergenza superlineare $p = 1, C = 0$

$$\lim_{k \rightarrow \infty} \frac{|x_{k+1} - x_*|}{|x_k - x_*|} = 0$$

Ordine di convergenza del metodo di bisezione

Assumendo che valga l'approssimazione $|c_k - x_*| \simeq \frac{b-a}{2^k}$, si può mostrare che il metodo di bisezione ha convergenza lineare

$$\lim_{k \rightarrow \infty} \frac{|x_{k+1} - x_*|}{|x_k - x_*|} \simeq \lim_{k \rightarrow \infty} \frac{\frac{b-a}{2^{k+1}}}{\frac{b-a}{2^k}} = \frac{1}{2}$$

Riassumendo, il metodo di bisezione

- richiede solo la continuità della funzione.
- ha una bassa complessità computazionale (1 valutazione di funzione per iterazione).

D'altra parte

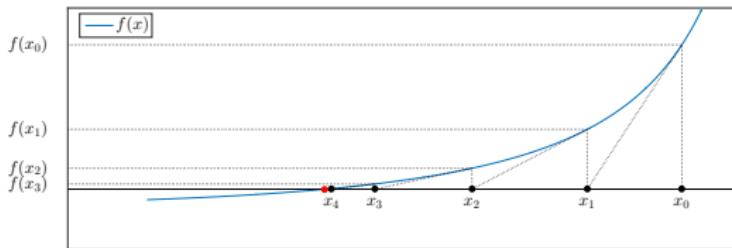
- converge lentamente (linearmente) ad una soluzione.
- non si può estendere al caso di sistemi di equazioni nonlineari.

Introduciamo un metodo di ordine superiore.

Metodo di Newton

o metodo delle tangenti

- Fissato un intervallo di riferimento $[a, b]$, il metodo di Newton si applica a funzioni differenziabili, $f \in C^1([a, b])$.
- A partire da un punto iniziale $x_0 \in [a, b]$, l'iterato successivo x_{k+1} viene calcolato come l'intersezione tra l'asse delle ascisse e la retta tangente al grafico di f nel punto $(x_k, f(x_k))$, per ogni $k = 0, 1, \dots$



$$\begin{cases} y = 0 \\ y = f(x_k) + f'(x_k)(x - x_k) \end{cases} \Rightarrow x = x_k - \frac{f(x_k)}{f'(x_k)}$$

Metodo di Newton

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

- Per essere ben posto, deve essere $f'(x_k) \neq 0$ per ogni k , altrimenti il metodo si arresta.
- Il metodo di Newton può essere interpretato anche nel modo seguente. Le iterate si ottengono sostituendo il problema nonlineare $f(x) = 0$ con una sua successione di problemi lineari $f(x_k) + f'(x_k)(x - x_k) = 0$ ottenuti approssimando la funzione f con uno sviluppo di Taylor centrato in x_k e troncato al primo ordine.
- Il costo computazionale per iterazione è di 2 valutazioni di funzione ($f(x_k)$ e $f'(x_k)$)

NB:

I metodi di ordine superiore hanno anche una maggiore complessità computazionale. Il rapporto costi/benefici va valutato in relazione allo specifico problema a cui si vuole applicare il metodo.

Ordine di convergenza quadratica del metodo di Newton

Ipotesi

$$f \in C^2([a, b]) \\ \lim_{k \rightarrow \infty} x_k = x_* \text{ con } f(x_*) = 0, f'(x_*) \neq 0$$

Tesi

$$\lim_{k \rightarrow \infty} \frac{|x_{k+1} - x_*|}{|x_k - x_*|^2} = C \text{ con } C \in \mathbb{R}$$

Applichiamo il teorema di Taylor con resto nella forma di Lagrange

$$f(x^*) = f(x_k) + f'(x_k)(x_* - x_k) + \frac{1}{2}f''(\xi_k)(x_* - x_k)^2$$

dove $\xi_k \in [x_k, x_*]$. Siccome $f(x_*) = 0$, dividendo entrambi i membri della precedente uguaglianza per $f'(x_k)$ si ottiene

$$0 = \underbrace{\frac{f(x_k)}{f'(x_k)} - x_k}_{= -x_{k+1}} + x_* + \frac{f''(\xi_k)}{2f'(x_k)}(x_* - x_k)^2$$

$$0 = x_* - x_{k+1} + \frac{f''(\xi_k)}{2f'(x_k)}(x_* - x_k)^2$$

Dividendo ancora entrambi i membri per $(x_* - x_k)^2$ si ottiene

$$\frac{x_* - x_{k+1}}{(x_* - x_k)^2} = -\frac{f''(\xi_k)}{2f'(x_k)}$$

Infine, passando al limite e applicando la continuità di f' ed f'' si ottiene

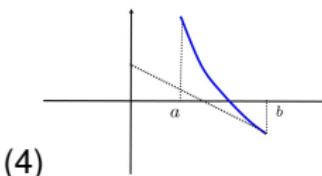
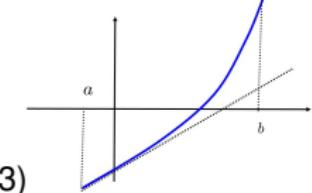
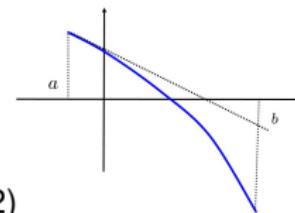
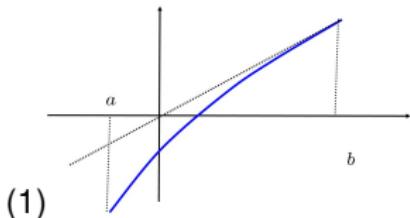
$$\lim_{k \rightarrow \infty} \frac{|x_* - x_{k+1}|}{|x_* - x_k|^2} = \frac{|f''(x_*)|}{2|f'(x_*)|}$$

Teorema di convergenza del metodo di Newton

Sia $f \in C^2([a, b])$. Se $f'(x) \neq 0$, $\forall x \in [a, b]$ e vale una delle seguenti condizioni

- ① $f(a) < 0$, $f(b) > 0$, $f''(x) \leq 0$, $\forall x \in [a, b]$ e $f(b) + f'(b)(a - b) \leq 0$
- ② $f(a) > 0$, $f(b) < 0$, $f''(x) \leq 0$, $\forall x \in [a, b]$ e $f(a) + f'(a)(b - a) \leq 0$
- ③ $f(a) < 0$, $f(b) > 0$, $f''(x) \geq 0$, $\forall x \in [a, b]$ e $f(a) + f'(a)(b - a) \geq 0$
- ④ $f(a) > 0$, $f(b) < 0$, $f''(x) \geq 0$, $\forall x \in [a, b]$ e $f(b) + f'(b)(a - b) \geq 0$

allora si ha che, per ogni scelta del punto iniziale $x_0 \in [a, b]$, le iterate del metodo di Newton appartengono tutte all'intervallo $[a, b]$ e convergono all'unica radice x_* di f in $[a, b]$ con velocità quadratica.



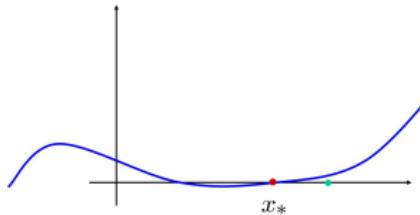
Criteri di arresto

- Ad eccezione del metodo di bisezione, nei metodi iterativi non si conosce a priori il numero di iterazioni sufficiente ad ottenere un'approssimazione della soluzione entro una tolleranza fissata
- Solitamente si adotta una combinazione di criteri di arresto, analoghi a quelli usati per i metodi iterativi per sistemi lineari, basati su due quantità:
 - la differenza tra due iterate successive
 - il residuo del problema, definito come una quantità che si annulla in corrispondenza della soluzione
- Fissata una tolleranza τ , i metodi per la ricerca degli zeri di funzione si arrestano alla prima iterazione k in cui sono soddisfatte le disuguaglianze

$$\frac{|x_{k+1} - x_k|}{|x_{k+1}|} \leq \tau \quad \text{e} \quad |f(x_k)| \leq \tau$$

NB:

Il residuo non è affidabile quando il grafico di f è molto schiacciato sull'asse x



Pseudocodice del metodo di Newton

INPUT: f, f', x, τ, N_{max}
for $k = 1, \dots, N_{max}$
 $fx \leftarrow f(x)$
 $dfx \leftarrow f'(x)$
 $x_{new} \leftarrow x - fx / dfx$
 If $|x - x_{new}| / |x_{new}| \leq \tau$ **AND** $|fx| \leq \tau$
 return
 Endif
 $x \leftarrow x_{new}$
OUTPUT: x

Estensione ai sistemi di equazioni nonlineari

Accenniamo al fatto che, a differenza dei metodi dicotomici, le idee alla base del metodo di Newton possono essere estese per problemi del tipo

$$F(x) = 0, F : \mathbb{R}^n \rightarrow \mathbb{R}^n, x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} F(x) = \begin{pmatrix} f_1(x_1, \dots, x_n) \\ f_2(x_1, \dots, x_n) \\ \vdots \\ f_n(x_1, \dots, x_n) \end{pmatrix}$$

In questo caso si introduce la matrice Jacobiana

$$JF(x) = \begin{pmatrix} \frac{\partial f_1(x)}{\partial x_1} & \frac{\partial f_1(x)}{\partial x_2} & \cdots & \frac{\partial f_1(x)}{\partial x_n} \\ \frac{\partial f_2(x)}{\partial x_1} & \frac{\partial f_2(x)}{\partial x_2} & \cdots & \frac{\partial f_2(x)}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_n(x)}{\partial x_1} & \frac{\partial f_n(x)}{\partial x_2} & \cdots & \frac{\partial f_n(x)}{\partial x_n} \end{pmatrix}$$

$$\begin{array}{rclcrcl} f'(x_k)d_k & = & -f(x_k) & \Rightarrow & JF(x^{(k)})d^{(k)} & = & -F(x^{(k)}) \\ x_{k+1} & = & x_k + d_k & & x^{(k+1)} & = & x^{(k)} + d^{(k)} \end{array}$$

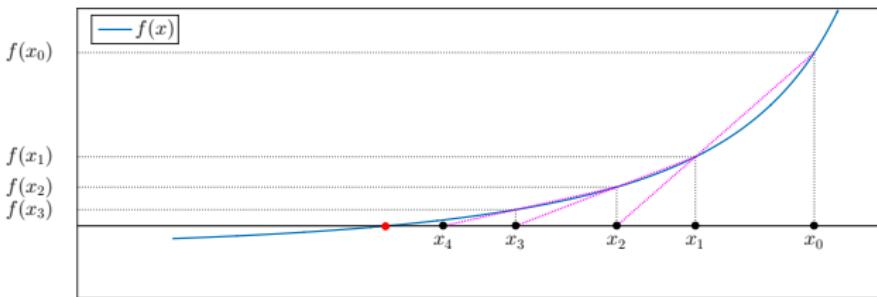
Nel caso n -dimensionale, il metodo di Newton richiede la soluzione di un sistema lineare di ordine n ad ogni passo.

Metodo delle secanti

E' una variante del metodo di Newton in cui la derivata prima viene approssimata con un rapporto incrementale:

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

Il nuovo punto è l'intersezione tra l'asse delle ascisse e la retta per i punti di coordinate $(x_{k-1}, f(x_{k-1}))$, $(x_k, f(x_k))$.



- Non richiede il calcolo della derivata prima
- Si può dimostrare, sotto opportune ipotesi, la convergenza superlineare
- Esiste un corrispondente metodo per sistemi lineari, chiamato metodo *quasi-Newton* in cui si sostituisce $JF(x^{(k)})$ con una sua approssimazione B_k
- La funzione di Matlab `fzero` è una combinazione tra metodo di bisezione e metodo delle secanti

Metodo delle approssimazioni successive

Riformulazione di un'equazione lineare

Analogia con i sistemi lineari

Sistema lineare

$$Ax = b \Leftrightarrow b - Ax = 0$$

$$0 = -M^{-1}(Ax - b)$$

dove M nonsingolare

$$x = x - M^{-1}(Ax - b),$$

$$x = \underbrace{(I - M^{-1}A)}_G x + \underbrace{M^{-1}b}_c$$

$$x = Gx + c$$

$$x^{(k+1)} = Gx^{(k)} + c$$

Equazione nonlineare

$$f(x) = 0$$

$$-\phi(x)f(x) = 0$$

dove $\phi : \mathbb{R} \rightarrow \mathbb{R}$, $\phi(x) \neq 0 \forall x$

$$x = x - \phi(x)f(x),$$

$$x = \underbrace{x - \phi(x)f(x)}_{g(x)}$$

$$x = g(x)$$

$$x_{k+1} = g(x_k)$$

Il problema del punto fisso

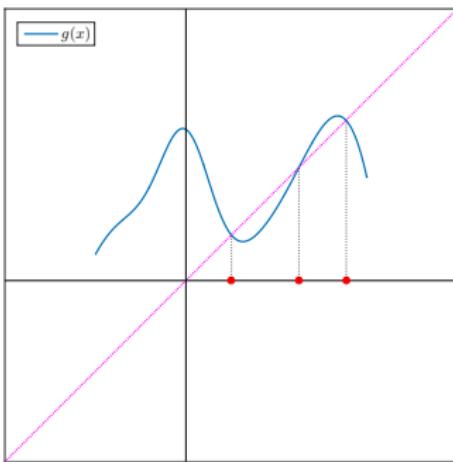
Definizione

Data una funzione $g : [a, b] \rightarrow \mathbb{R}$, un punto $x_* \in [a, b]$ tale che

$$g(x_*) = x_*$$

si definisce *punto fisso* della funzione g .

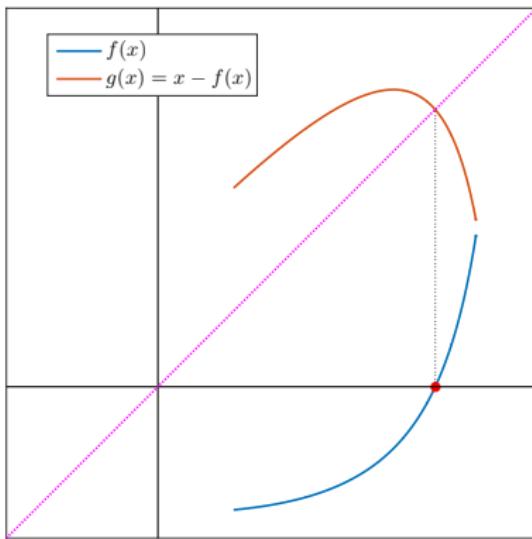
- Dal punto di vista geometrico, in corrispondenza di un punto fisso il grafico della funzione g interseca la bisettrice del primo e del terzo quadrante, di equazione $y = x$.



Equazioni nonlineari come problemi di punto fisso

Sia $f : [a, b] \rightarrow \mathbb{R}$. Data una funzione $\phi : [a, b] \rightarrow \mathbb{R}$ a valori non nulli, $\phi(x) \neq 0 \forall x \in [a, b]$, le radici di f sono tutti e soli i punti fissi della seguente funzione:

$$g(x) = x - \phi(x)f(x)$$

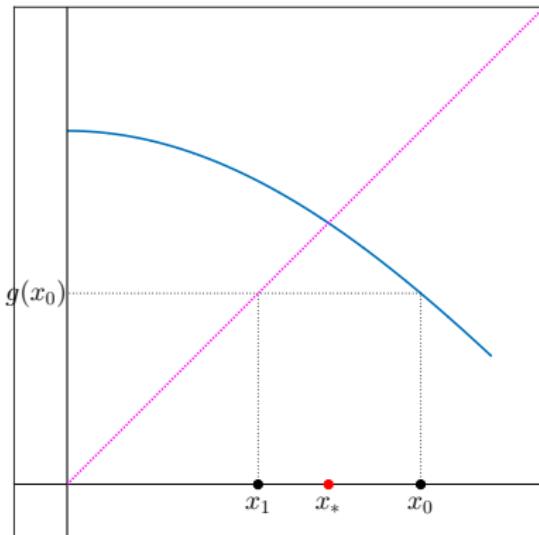


Il metodo delle approssimazioni successive

Definizione

Data una funzione $g : [a, b] \rightarrow \mathbb{R}$ e dato un punto iniziale $x_0 \in [a, b]$, *metodo delle approssimazioni successive*, detto anche *metodo del punto fisso* è definito dalla seguente iterazione:

$$x_{k+1} = g(x_k)$$



Metodo di Newton come metodo del punto fisso

- Come nel caso dei sistemi lineari, in cui la scelta di M corrisponde ad un diverso metodo, la scelta della funzione $\phi(x)$ nella definizione di $g(x)$ come $g(x) = x - \phi(x)f(x)$ determina diversi metodi delle approssimazioni successive che sono anche metodi per la ricerca degli zeri di f .
- Nelle opportune ipotesi, scegliendo

$$\phi(x) = \frac{1}{f'(x)}$$

il metodo delle approssimazioni successive per la ricerca dei punti fissi di $g(x) = x - \phi(x)f(x)$ corrisponde al metodo di Newton applicato ad f .

Esistenza e unicità del punto fisso e convergenza del metodo delle approssimazioni successive

Teorema della mappa contrattiva

Sia $g : [a, b] \rightarrow \mathbb{R}$. Se valgono le seguenti ipotesi

- ① $g(x) \in [a, b], \forall x \in [a, b];$
- ② g è una *contrazione* in $[a, b]$, ossia esiste $L \in (0, 1)$ tale che
 $|g(x) - g(y)| \leq L|x - y|, \forall x, y \in [a, b],$

allora si ha che:

- esiste un unico punto fisso x_* di g in $[a, b]$
- la successione generata dal metodo delle approssimazioni successive $x_{k+1} = g(x_k)$ converge a x_* , per ogni punto iniziale $x_0 \in [a, b]$
- per ogni iterata del metodo si ha

$$|x_k - x_*| \leq \frac{L^k}{1-L} |x_1 - x_0|$$

- Se viene a mancare una delle ipotesi del metodo, l'esistenza e/o l'unicità del punto fisso non sono più garantite
- Una condizione sufficiente affinchè una funzione differenziabile g sia contrattiva in $[a, b]$ è che $|g'(x)| \leq L < 1, \forall x \in [a, b]$
- L'ipotesi di contrattività è analoga alla condizione necessaria e sufficiente per la convergenza di un metodo iterativo per sistemi lineari

$$x^{(k+1)} = Gx^{(k)} + c \quad \rho(G) < 1$$

Dimostrazione del teorema della mappa contrattiva

(1)- convergenza della successione $\{x_k\}_{k \in \mathbb{N}}$

Dall'ipotesi 1 si ha che per ogni k si ha $x_k \in [a, b]$. Si verifica che $\{x_k\}_{k \in \mathbb{N}}$ è una successione di Cauchy, ossia che vale

$$\text{tesi: } \lim_{k \rightarrow \infty} |x_{k+p} - x_k| = 0, \forall p > 0$$

Per la diseguaglianza triangolare si ha

$$|x_k - x_{k+p}| = |x_k \pm x_{k+1} \pm \dots \pm x_{k+p-1} - x_{k+p}| \leq \sum_{j=0}^{p-1} |x_{k+j} - x_{k+j+1}|$$

Inoltre si ha

$$\begin{aligned} |x_{k+j} - x_{k+j+1}| &= |g(x_{k+j-1}) - g(x_{k+j})| \\ &\leq L|x_{k+j-1} - x_{k+j}| \leq \dots \leq L^j |x_k - x_{k+1}| \end{aligned}$$

Segue che

$$\begin{aligned} |x_k - x_{k+p}| &\leq \sum_{j=0}^{p-1} L^j |x_k - x_{k+1}| \\ &= \frac{1 - L^p}{1 - L} |x_k - x_{k+1}| \leq \frac{1}{1 - L} |x_k - x_{k+1}| \\ &\leq \frac{L^k}{1 - L} |x_0 - x_1| \xrightarrow{k \rightarrow \infty} 0 \end{aligned}$$

Dimostrazione del teorema della mappa contrattiva

(2) - esistenza ed unicità del punto fisso e stima dell'errore

Quindi la successione $\{x_k\}_{k \in \mathbb{N}}$ converge ad un punto $x_* \in [a, b]$. Per continuità, si ha che

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

da cui segue che è un punto fisso. Se per assurdo supponiamo che esista un altro punto fisso in $y_* \in [a, b]$, $g(y^*) = y^* \neq x_*$, dall'ipotesi di contrattività si avrebbe

$$L|x_* - y_*| \geq |g(x_*) - g(y_*)| = |x_* - y_*|, \text{ con } L < 1$$

che è assurdo.

Infine, dalla dimostrazione di convergenza abbiamo ottenuto

$$|x_k - x_{k+p}| \leq \frac{L^k}{1-L} |x_0 - x_1|,$$

da cui, facendo il limite di entrambi i membri per $p \rightarrow \infty$ e osservando che il lato destro non dipende da p , si ottiene la maggiorazione dell'errore

$$|x_k - x_*| \leq \frac{L^k}{1-L} |x_0 - x_1|.$$

Estensioni del teorema della mappa contrattiva

- L'enunciato e i passi della dimostrazione del teorema precedente rimangono identici anche nel caso in cui g è una mappa n dimensionale

$$g : \mathbb{R}^n \rightarrow \mathbb{R}^n, x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, g(x) = \begin{pmatrix} g_1(x_1, \dots, x_n) \\ g_2(x_1, \dots, x_n) \\ \vdots \\ g_n(x_1, \dots, x_n) \end{pmatrix}$$

- In tal caso, il valore assoluto $|\cdot|$ si generalizza con una norma vettoriale $\|\cdot\|$.
- Il teorema della mappa contrattiva è alla base delle dimostrazioni di convergenza di una grandissima parte dei metodi numerici iterativi esistenti in letteratura.

Velocità di convergenza del metodo del punto fisso

Dalla maggiorazione dell'errore

$$|x_k - x_*| \leq \frac{L^k}{1-L} |x_0 - x_1|.$$

si ha che

$$|x_k - x_*| = \mathcal{O}(L^k)$$

per cui tanto più L è piccolo, tanto più veloce sarà la convergenza (si noti l'analogia con $\rho(G)$).

Ordine di convergenza del metodo del punto fisso

Teorema

Se le ipotesi del teorema della mappa contrattiva sono soddisfatte ed inoltre si ha $g \in C^p([a, b])$, con $g^{(k)}(x_*) = 0$, $k = 1, \dots, p - 1$ e $g^{(p)}(x_*) \neq 0$, allora il metodo delle approssimazioni successive associato a g ha ordine p .

Dimostrazione. Dal teorema di Taylor e dalla definizione di punto fisso, per qualche ξ_k compreso tra x_* e x_k si ha

$$\begin{aligned} x_{k+1} &= g(x_k) \\ &= g(x_*) + g'(x_*)(x_k - x_*) + \frac{1}{2}g''(x_*)(x_k - x_*)^2 + \dots \\ &\quad \dots + \frac{1}{(p-1)!}g^{(p-1)}(x_*)(x_k - x_*)^{p-1} + \frac{1}{p!}g^{(p)}(\xi_k)(x_k - x_*)^p \\ &= x_* + \frac{1}{p!}g^{(p)}(\xi_k)(x_k - x_*)^p \end{aligned}$$

da cui si ricava

$$\frac{|x_{k+1} - x_*|}{|x_k - x_*|^p} = \frac{1}{p!}|g^{(p)}(\xi_k)| \xrightarrow{k \rightarrow \infty} \frac{1}{p!}|g^{(p)}(x_*)|$$

per continuità di $g^{(p)}$ e perché ξ_k è compreso tra x_* e x_k .

Caso particolare

La convergenza quadratica del metodo di Newton si può ottenere dal precedente risultato come caso particolare, osservando che

$$g(x) = x - \frac{f(x)}{f'(x)}$$

$$g'(x) = 1 - \frac{f'(x)^2 - f(x)f''(x)}{f'(x)^2} = \frac{f(x)f''(x)}{f'(x)}$$

da cui si ha che, essendo x_* una radice di f

$$g'(x_*) = 0$$

Stabilità del metodo del punto fisso

- Si tiene conto dell'introduzione degli errori dovuti all'aritmetica finita nel calcolo di $g(x)$
- Invece di $g(x)$ si calcola

$$g(x) + \delta(x), \quad \text{con } |\delta(x)| \leq u$$

- Invece di $x_{k+1} = g(x_k)$ si calcola

$$w_{k+1} = g(w_k) + \delta_k, \quad \text{con } |\delta_k| \leq u$$

Nelle ipotesi del teorema della mappa contrattiva si dimostra che

$$|w_k - x_*| \leq \frac{u}{1-L} + \rho L^k$$

con ρ costante positiva.

Assumendo che la diseguaglianza valga come un'uguaglianza approssimata, in generale non si ha più convergenza, poiché, mentre il termine ρL^k tende a zero, l'altro addendo $\frac{u}{1-L}$ non dipende da k e, se $L \simeq 1$, può amplificare di molto la precisione u .

Interpolazione di dati e funzioni

Esempio

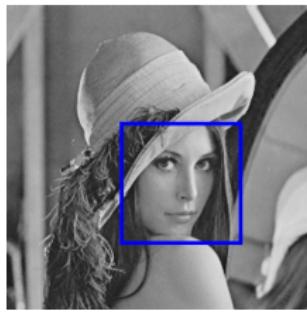


- Supponiamo di voler programmare un gioco di guida, in cui il giocatore controlla direttamente l'accelerazione della macchina e che tale accelerazione sia descritta da una funzione $a(t)$.
- L'accelerazione $a(t)$ è decisa dal giocatore durante il gioco mediante i comandi, tastiera o joystic, e non è nota a priori.
- E' ragionevole assumere che l'interpretazione dei comandi fornisca piuttosto dei valori $a(t_i)$, dove $t_i, i = 0, 1, \dots, n$ rappresentano diversi momenti del gioco.
- Per conoscere la posizione della macchina, occorre integrare due volte l'accelerazione che però non è nota in forma analitica.
- Occorre definire una approssimazione della funzione $a(t)$ a partire dai valori noti $a(t_i)$, dove $t_i, i = 0, 1, \dots, n$.

Esempio

Ingrandimento di un'immagine

- Un'immagine in formato digitale, che supponiamo per semplicità in scala di grigi, consiste in una matrice di numeri, detti *pixel*, che rappresentano la sfumatura di grigio presente nel corrispondente punto dell'immagine.
- Ingrandire un'immagine significa definire una matrice più grande, di cui dobbiamo determinare pixel in più rispetto a quelli dell'immagine di partenza, in modo che siano 'coerenti' con essi.



Esempio

Ingrandimento di un'immagine: due diverse strategie



Definizione

Siano (x_i, y_i) , $i = 0, \dots, n$ punti fissati nel piano cartesiano, con $x_i \in [a, b]$. Il problema dell'interpolazione consiste nel costruire una funzione $g : [a, b] \rightarrow \mathbb{R}$ il cui grafico passa per i punti dati, ossia che soddisfa le condizioni di interpolazione

$$g(x_i) = y_i, \quad \forall i = 0, \dots, n$$

La funzione g si dice *interpolante* rispetto ai punti di interpolazione (x_i, y_i) .

Le ascisse dei punti di interpolazione si chiamano *nodi*.

Quando le ordinate dei punti di interpolazione sono le immagini dei nodi tramite una funzione $f : [a, b] \rightarrow \mathbb{R}$, allora si dice che g *interpola la funzione f*

- Fissati i punti da interpolare, esistono infinite funzioni interpolanti.
- Per avere l'unicità della soluzione del problema dell'interpolazione occorre restringere la classe di funzioni interpolanti.

Il polinomio di interpolazione

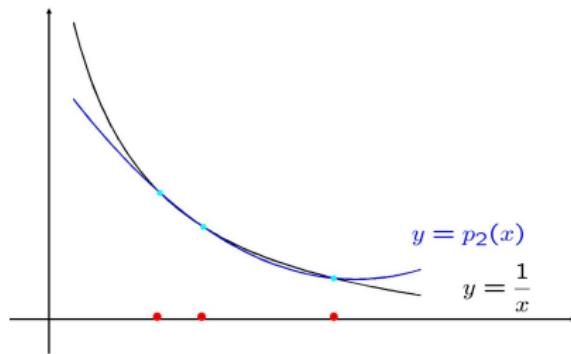
Teorema fondamentale dell'algebra

Dati $n + 1$ punti del piano (x_i, y_i) , $i = 0, \dots, n$, esiste un unico polinomio di grado al più n , denotato con $p_n(x)$, che li interpola, ossia tale che

$$p_n(x_i) = y_i, \quad i = 0, \dots, n + 1.$$

Il polinomio $p_n(x)$ viene detto *polinomio di interpolazione* dei punti (x_i, y_i) , $i = 0, \dots, n$.

- Fissati i punti da interpolare, scegliendo il grado del polinomio come il numero dei punti meno uno, il problema dell'interpolazione ha un'unica soluzione.



“Calcolo” del polinomio di interpolazione

- Dal punto di vista numerico “calcolare” il polinomio di interpolazione significa progettare un algoritmo che, dati in input i punti di interpolazione (x_i, y_i) , $i = 0, \dots, n$, e un punto $\bar{x} \in \mathbb{R}$, restituisca in output il valore $p_n(\bar{x})$
- A seconda di come si rappresenta il polinomio, ci sono diverse possibilità per individuarlo in modo univoco tramite $n + 1$ valori reali

Rappresentazione in forma canonica

Un generico polinomio di grado n si rappresenta in forma canonica come combinazione lineare delle funzioni

$$1, x, x^2, \dots, x^n$$

e si scrive come

$$p_n(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n.$$

Le condizioni di interpolazione $p_n(x_i) = y_i$, $i = 0, \dots, n$ si possono quindi scrivere come

$$\left\{ \begin{array}{l} a_0 + a_1 x_0 + a_2 x_0^2 + \dots + a_n x_0^n = y_0 \\ a_0 + a_1 x_1 + a_2 x_1^2 + \dots + a_n x_1^n = y_1 \\ \dots \\ a_0 + a_1 x_n + a_2 x_n^2 + \dots + a_n x_n^n = y_n \end{array} \right.$$

che si può interpretare come un sistema lineare di $n + 1$ equazioni

$$V\alpha = y$$

Metodo dei coefficienti indeterminati

Dati, (x_i, y_i) , $i = 0, \dots, n$, si calcolano

$$V = \begin{pmatrix} 1 & x_0 & \dots & {x_0}^n \\ 1 & x_1 & \dots & {x_1}^n \\ \dots & \dots & \dots & \dots \\ 1 & x_n & \dots & {x_n}^n \end{pmatrix}, y = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

e si risolve il sistema lineare

$$V\alpha = y$$

Le componenti della soluzione sono i coefficienti del polinomio di interpolazione $p_n(x)$ in forma canonica.

- La matrice V si chiama *matrice di Vandermonde*.
- Si dimostra che $\det(V) = \prod_{i>j}(x_i - x_j)$, pertanto se i punti di interpolazione sono distinti, V è non singolare e il sistema $V\alpha = y$ ha un'unica soluzione (abbiamo ritrovato il teorema fondamentale dell'algebra)

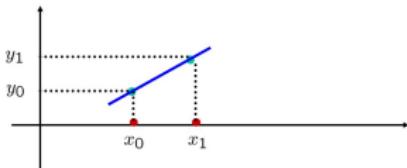
Costo computazionale:

- La soluzione diretta del sistema $V\alpha = y$ ha un costo dell'ordine di n^3 (fattorizzazione di V)
- Una volta calcolati i coefficienti a_0, a_1, \dots, a_n , dato un punto $\bar{x} \neq x_i$, $i = 0, \dots, n$, si vuole calcolare $p_n(\bar{x})$. L'algoritmo più efficiente è il metodo di Horner, che ha un costo di n prodotti e somme floating point.
- Vantaggi: i coefficienti a_0, \dots, a_n della rappresentazione canonica non dipendono dal punto \bar{x} : una volta calcolati, si possono riutilizzare per calcolare lo stesso polinomio in un diverso punto.
- Svantaggi: la matrice di Vandermonde è, in generale, malcondizionata.

Verso una nuova rappresentazione

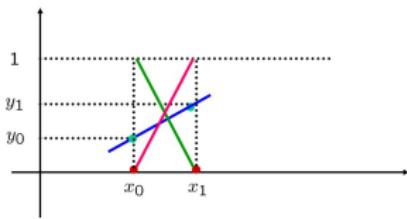
Consideriamo prima il caso $n = 1$, in cui, il polinomio $p_1(x)$ consiste nella retta che passa per i punti dati $(x_0, y_0), (x_1, y_1)$

$$p_1(x) = y_0 + (x - x_0) \frac{y_1 - y_0}{x_1 - x_0}$$



Raggruppando in modo diverso i termini, $p_1(x)$ si può scrivere come

$$p_1(x) = y_0 \underbrace{\frac{x - x_1}{x_0 - x_1}}_{L_0(x)} + y_1 \underbrace{\frac{x - x_0}{x_1 - x_0}}_{L_1(x)}$$



$$p_1(x) = y_0 \underbrace{\frac{x - x_1}{x_0 - x_1}}_{L_0(x)} + y_1 \underbrace{\frac{x - x_0}{x_1 - x_0}}_{L_1(x)} = y_0 L_0(x) + y_1 L_1(x)$$

- Abbiamo scritto il polinomio di interpolazione di grado 1 come combinazione lineare di polinomi di grado 1, denotati con $L_0(x), L_1(x)$
- I coefficienti della combinazione lineare sono esattamente le ordinate dei punti di interpolazione y_0, y_1 .
- I polinomi $L_0(x), L_1(x)$ della combinazione hanno la seguente proprietà che li caratterizza completamente

$$\begin{cases} L_0(x_0) &= 1 \\ L_0(x_1) &= 0 \end{cases}, \quad \begin{cases} L_1(x_0) &= 0 \\ L_1(x_1) &= 1 \end{cases}$$

Rappresentazione nella forma di Lagrange

Generalizzazione al grado n

Si vuole scrivere il polinomio di interpolazione relativo ai punti (x_i, y_i) , $i = 0, \dots, n$ come

$$p_n(x) = y_0 L_0(x) + y_1 L_1(x) + \dots + y_n L_n(x)$$

dove $L_k(x)$, $k = 0, \dots, n$ sono $n+1$ polinomi di grado n che si annullano in tutti i punti di interpolazione tranne uno, in cui valgono 1:

$$\begin{cases} L_k(x_k) &= 1 \\ L_k(x_j) &= 0 \text{ se } k \neq j \end{cases}$$

La forma esplicita dei polinomi $L_k(x)$ è:

$$\begin{aligned} L_k(x) &= \frac{(x - x_0) \cdot \dots \cdot (x - x_{k-1})(x - x_{k+1}) \cdot \dots \cdot (x - x_n)}{(x_k - x_0) \cdot \dots \cdot (x_k - x_{k-1})(x_k - x_{k+1}) \cdot \dots \cdot (x_k - x_n)} \\ &= \prod_{i=0, i \neq k}^n \frac{x - x_i}{x_k - x_i}, \quad k = 0, \dots, n \end{aligned}$$

La base di Lagrange

L'insieme dei polinomi

$$\{L_k\}_{k=0,\dots,n}$$

viene detto *base di Lagrange*. A differenza della base dei monomi $1, x, x^2, \dots, x^n$ la base di Lagrange dipende dai nodi.

Teorema

La base di Lagrange è un insieme linearmente indipendente

Dimostrazione. Siano $\lambda_0, \dots, \lambda_n$ $n+1$ coefficienti reali tali che $\sum_{k=0}^n \lambda_k L_k = 0$

$$0 = \lambda_0 L_0(x) + \lambda_1 L_1(x) + \dots + \lambda_n L_n(x) \quad \forall x \in \mathbb{R}.$$

In particolare l'uguaglianza deve valere per i punti di interpolazione, ossia $x = x_k$

$$0 = \lambda_0 L_0(x_k) + \lambda_1 L_1(x_k) + \dots + \lambda_n L_n(x_k) = \lambda_k L_k(x_k) = \lambda_k$$

Quindi si ha $\lambda_k = 0$ per ogni $k = 0, \dots, n$ e quindi, per definizione, segue la lineare indipendenza della base di Lagrange.

Metodo di Lagrange per il calcolo del polinomio di interpolazione

Input: (x_i, y_i) , $i = 0, \dots, n$ e $\bar{x} \neq x_i$

Output: $p_n(\bar{x})$

- ① Calcolare i valori $L_k(\bar{x})$, $k = 0, \dots, n$
- ② Calcolare la combinazione lineare $p_n(\bar{x}) = y_0 L_0(\bar{x}) + y_1 L_1(\bar{x}) + \dots + y_n L_n(\bar{x})$

La parte computazionalmente più costosa del metodo è la valutazione della base di Lagrange in \bar{x}

$$L_k(\bar{x}) = \prod_{i=0, i \neq k} \frac{\bar{x} - x_i}{x_k - x_i}, k = 0, \dots, n$$

Per evitare di ripetere più volte le stesse operazioni nel calcolo dei numeratori, si può procedere calcolando inizialmente la quantità

$$\omega_n(\bar{x}) = (\bar{x} - x_0) \cdot \dots \cdot (\bar{x} - x_n)$$

e osservando che

$$L_k(\bar{x}) = \frac{\omega_n(\bar{x})}{(\bar{x} - x_k) \prod_{i=0, i \neq k} (x_k - x_i)}$$

Pseudocodice del metodo di Lagrange

$$\left\{ \begin{array}{l} L_k(\bar{x}) = \prod_{i=0, i \neq k} \frac{\bar{x} - x_i}{x_k - x_i} \\ \omega_n(\bar{x}) = (\bar{x} - x_0) \cdot \dots \cdot (\bar{x} - x_n) \end{array} \right. \Rightarrow L_k(\bar{x}) = \frac{\omega_n(\bar{x})}{(\bar{x} - x_k) \prod_{i=0, i \neq k} (x_k - x_i)}$$

$$p_n(\bar{x}) = y_0 L_0(\bar{x}) + y_1 L_1(\bar{x}) + \dots + y_n L_n(\bar{x})$$

```
q ← 1
For i = 0, ..., n
    q ← q * (bar{x} - x_i)                                n somme e prodotti
For k = 0, 1, ..., n
    d ← 1
    For i = 0, ..., k - 1, k + 1, ..., n
        d ← d * (x_k - x_i)                                n2 + n prodotti, n divisioni.
    l_k ← q / ((bar{x} - x_k) d)
s ← 0
For k = 0, ..., n
    s ← s + y_k * l_k                                    n somme e prodotti
```

$$f(x) = \frac{\sin(6x)(3x)^2}{3x + 4}$$

$$f(x) = \frac{1}{1 + 25x^2}$$

$$x \in [-1, 1]$$

Norme di funzioni

Il concetto di ‘somiglianza’ o di distanza tra funzioni è espresso mediante la definizione di opportune norme di funzioni.

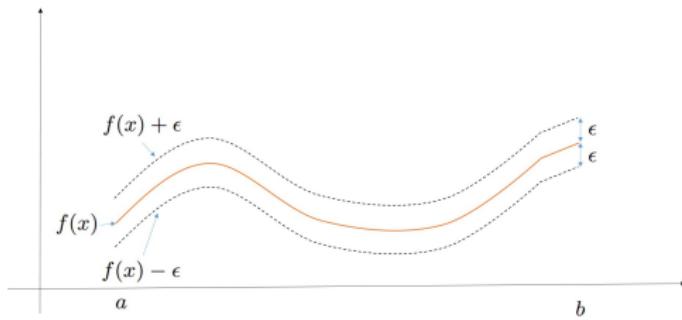
Definizione

Sia $f : [a, b] \rightarrow \mathbb{R}$ una funzione continua. Allora la *norma infinito* di f è definita come

$$\|f\|_{\infty} = \max_{x \in [a, b]} |f(x)|$$

La distanza in norma infinito tra f e $g : [a, b] \rightarrow \mathbb{R}$ è definita come $\|f - g\|_{\infty}$.

Se $\|f - g\|_{\infty} < \epsilon$, con $\epsilon > 0$, allora il grafico di g si trova in un ‘canale’ di raggio ϵ centratato sul grafico di f .

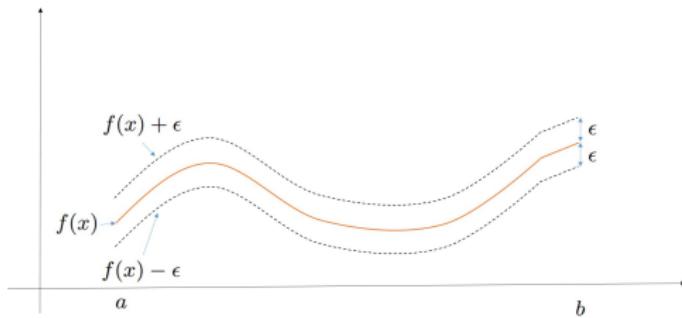


Errore (o resto) di interpolazione

- Ipotesi: $p_n(x)$ è il polinomio di interpolazione relativo a $(x_i, f(x_i))$,
 $i = 0, \dots, n$.
- Obiettivo: studiare la funzione

$$R_n(x) = f(x) - p_n(x), \text{ ed in particolare } \|R_n\|_\infty$$

per individuare le condizioni per cui $p_n(x)$ si può considerare una buona approssimazione di $f(x)$ nell'intervallo $[a, b]$.



Errore (o resto) di interpolazione

Teorema

Sia $f \in C^{n+1}([a, b])$, con $x_i \in [a, b]$, $i = 0, \dots, n$ e sia $p_n(x)$ il polinomio di interpolazione relativo a $(x_i, f(x_i))$, $i = 0, \dots, n$. Allora si ha

$$R_n(x) = \frac{\omega_{x_0, \dots, x_n}(x)}{(n+1)!} f^{(n+1)}(\xi)$$

dove $\xi \in [a, b]$ e $\omega_{x_0, \dots, x_n}(x) = (x - x_0) \cdot \dots \cdot (x - x_n)$.

Dimostrazione. Definiamo

$$\Omega(x, t) = (f(t) - p_n(t))\omega_{x_0, \dots, x_n}(x) - (f(x) - p_n(x))\omega_{x_0, \dots, x_n}(t)$$

Si ha che

$$\Omega(x, x) = 0 \quad \forall x \in [a, b] \text{ e } \Omega(x, x_i) = 0, \quad i = 0, \dots, n$$

Sia $x \in [a, b]$, $x \neq x_i$, $i = 0, \dots, n \Rightarrow \Omega(x, \cdot)$ si annulla negli $n+2$ punti x, x_0, \dots, x_n .

Per il teorema degli zeri (di Rolle), $\Omega'(x, \cdot)$ si annulla in $n+1$ punti in $[a, b]$.

Ripetendo gli stessi argomenti, si ottiene che la derivata $n+1$ -esima di $\Omega(x, \cdot)$ si annulla in un punto $\xi \in [a, b]$, ossia

$$0 = \Omega^{(n+1)}(x, \xi) = f^{(n+1)}(\xi)\omega_{x_0, \dots, x_n}(x) - (f(x) - p_n(x))\omega_{x_0, \dots, x_n}^{(n+1)}(\xi)$$

Osservando che $\omega_{x_0, \dots, x_n}^{(n+1)}(\xi) = (n+1)!$ e che $f(x) - p_n(x) = R_n(x)$ si ottiene il risultato.

Analisi del resto di interpolazione

Se $f \in C^{n+1}([a, b])$, allora

$$R_n(x) = \frac{\omega_{x_0, \dots, x_n}(x)}{(n+1)!} f^{(n+1)}(\xi)$$

dove $\xi \in [a, b]$ e

$$\omega_{x_0, \dots, x_n}(x) = (x - x_0) \cdot \dots \cdot (x - x_n).$$

Siccome $f^{(n+1)}$ e ω_{x_0, \dots, x_n} sono funzioni continue, hanno massimo e minimo in $[a, b]$. Definendo

$$M_{n+1}^f = \max_{x \in [a, b]} |f^{(n+1)}(x)|, \quad \omega_{x_0, \dots, x_n}^* = \max_{x \in [a, b]} |\omega_{x_0, \dots, x_n}(x)|$$

si ottiene la seguente stima:

$$|R_n(x)| \leq \frac{M_{n+1}^f \omega_{x_0, \dots, x_n}^*}{(n+1)!} \forall x \in [a, b]$$

che, con la notazione delle norme di funzioni, equivale a scrivere

$$\|R_n\|_\infty \leq \frac{M_{n+1}^f \omega_{x_0, \dots, x_n}^*}{(n+1)!}$$

Analisi del resto di interpolazione

$$|R_n(x)| \leq \frac{M_{n+1}^f \omega_{x_0, \dots, x_n}^*}{(n+1)!} \forall x \in [a, b]$$

Supponiamo che la funzione f sia sufficientemente regolare da poter maggiorare tutte le sue derivate con una stessa costante che non dipende dall'ordine:

$$|f^{(n+1)}(x)| \leq M^f, \forall x \in [a, b]$$

La stima si può scrivere come

$$|R_n(x)| \leq \frac{M^f \omega_{x_0, \dots, x_n}^*}{(n+1)!} \forall x \in [a, b]$$

Fattori che influiscono sull'errore

- numero dei punti da interpolare \Leftrightarrow grado del polinomio (al denominatore si trova $(n+1)!$)
- la tipologia dei punti da interpolare ($\omega_{x_0, \dots, x_n}^*$ dipende da x_0, \dots, x_n)

Strategie per minimizzare l'errore

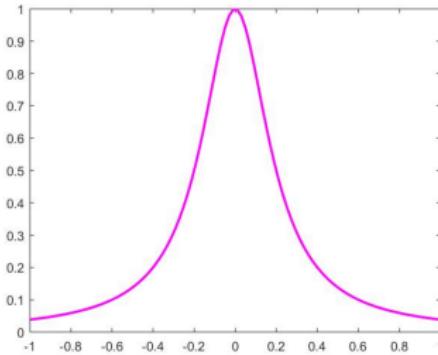
$$|R_n(x)| \leq \frac{M^f \omega_{x_0, \dots, x_n}^*}{(n+1)!} \quad \forall x \in [a, b]$$

- Aumentare il numero di punti apparentemente dovrebbe portare ad una riduzione dell'errore
- Tuttavia, se la collocazione dei punti da interpolare non è opportuna, questo non porta i risultati sperati, come mostra il seguente controesempio

Esempio: la funzione di Runge

Supponiamo di interpolare la funzione di Runge $f : [-1, 1] \rightarrow \mathbb{R}$

$$f(x) = \frac{1}{1 + 25x^2}$$

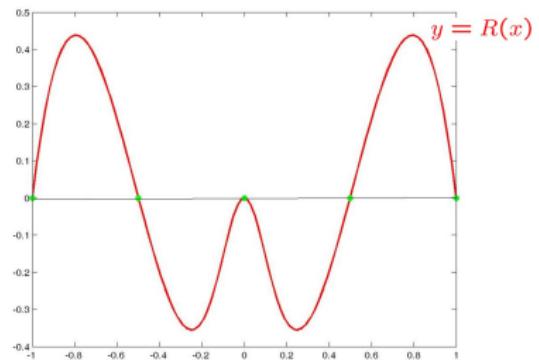
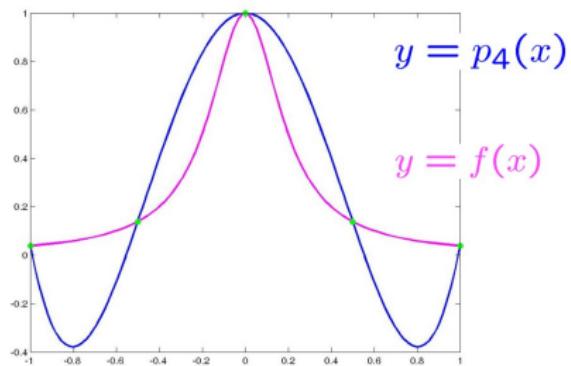


su una partizione di $n + 1$ punti *equispaziati* nell'intervallo $[-1, 1]$

$$x_i = -1 + \frac{i}{2n}, \quad i = 0, \dots, n$$

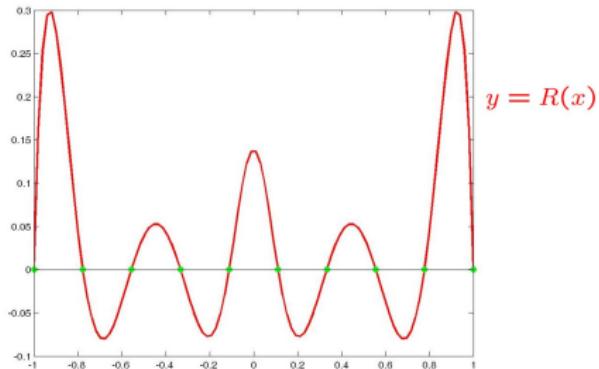
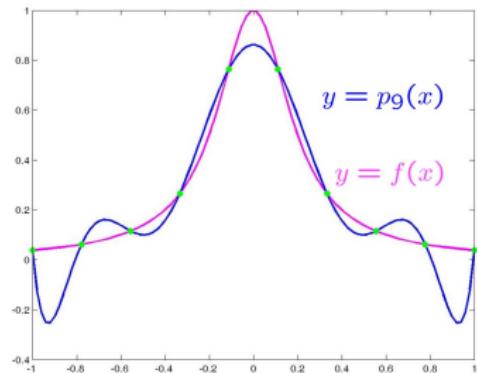
Esempio: la funzione di Runge

$n = 4$



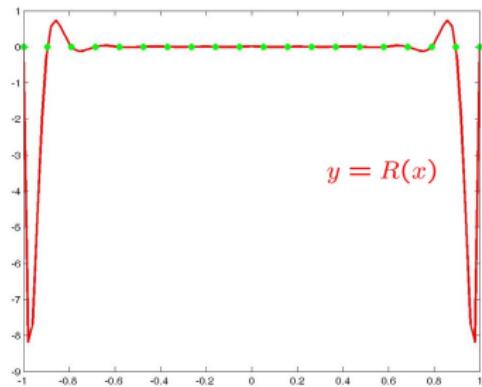
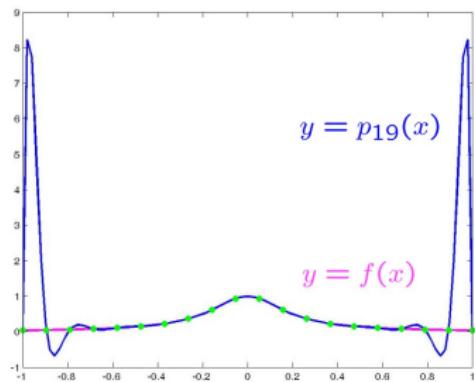
Esempio: la funzione di Runge

$n = 9$



Esempio: la funzione di Runge

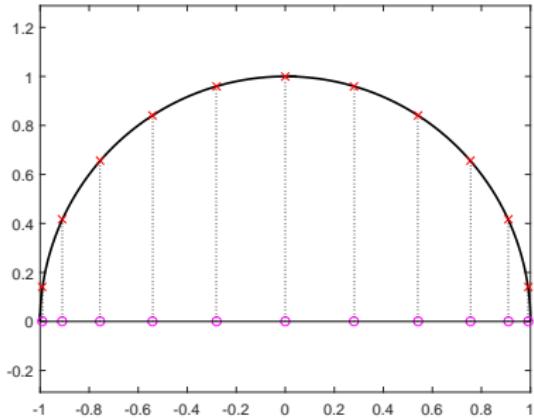
$n = 19$



- Aumentare semplicemente il numero dei punti senza considerarne la distribuzione non significa aumentare l'accuratezza dell'approssimazione.
- Nell'esempio di Runge, utilizzando un numero crescente di punti equispaziati si ottiene l'effetto opposto, per cui il polinomio di interpolazione presenta oscillazioni sempre più ampie in prossimità degli estremi dell'intervallo $[-1, 1]$, detto *fenomeno di Runge*.
- Una possibile strategia per evitare il fenomeno di Runge consiste nell'utilizzare una diversa distribuzione di nodi, più densa in prossimità degli estremi dell'intervallo dove occorre maggiore controllo.

Nodi di Chebychev

I nodi di Chebychev si ottengono partizionando in modo uniforme la semicirconferenza goniometrica e proiettando i punti di questa partizione sul diametro.



In formule:

$$x_i = \cos\left(\frac{(2i+1)\pi}{2(n+1)}\right), \quad i = 0, \dots, n$$

Proprietà dei nodi di Chebychev

$$x_i = \cos\left(\frac{(2i+1)\pi}{2(n+1)}\right), \quad i = 0, \dots, n$$

- ① Si possono estendere su un qualsiasi intervallo $[a, b]$ mediante la mappa $\mu(x) = \frac{b-a}{2}x + \frac{a+b}{2}$ che ne mantiene la distribuzione
- ② *Proprietà di min-max dei nodi di Chebychev.* Si dimostra che costituiscono la scelta dei punti di interpolazione ottimale, nel senso che la quantità

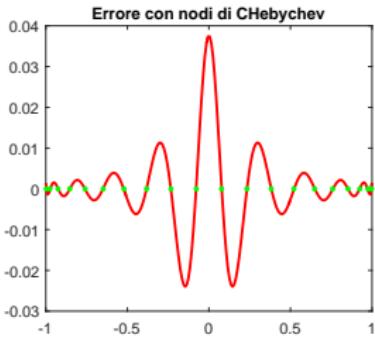
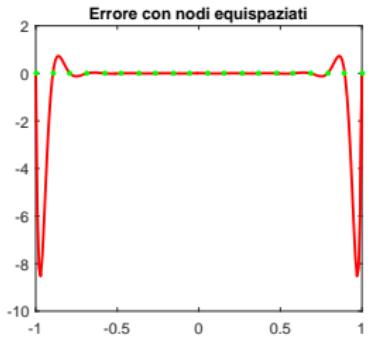
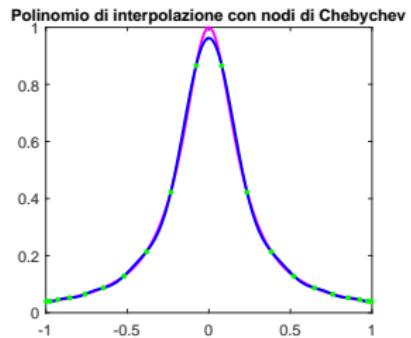
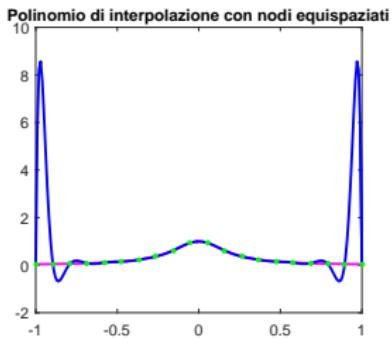
$$\omega_{x_0, \dots, x_n}^* = \max_{x \in [a, b]} |(x - x_0) \cdot \dots \cdot (x - x_n)|$$

calcolata quando x_0, \dots, x_n sono i nodi di Chebychev è minima rispetto a qualsiasi altra scelta della distribuzione dei nodi.

- ③ In questo caso si dimostra anche che

$$\omega_{x_0, \dots, x_n}^* = 2 \left(\frac{b-a}{4}\right)^{n+1}$$

Esempio: funzione di Runge con $n = 19$



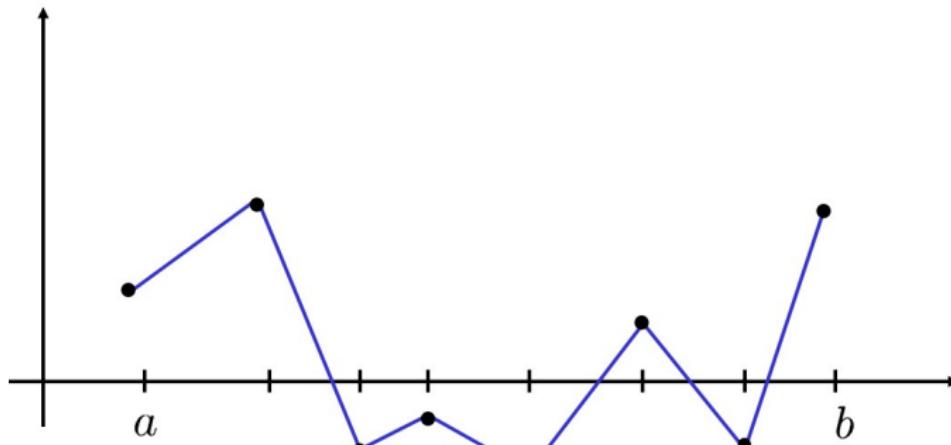
Interpolazione polinomiale a tratti

Svantaggi del polinomio di interpolazione

- Il polinomio di interpolazione dipende e si costruisce solo se tutti i punti di interpolazione sono noti a priori.
- Se si hanno a disposizione molti dati, si ottiene un polinomio di grado molto elevato (uno in meno del numero dei dati)
- Spesso nelle applicazioni non è possibile scegliere la distribuzione dei nodi che, nella maggior parte dei casi, è uniforme. Quindi il polinomio di interpolazione può essere soggetto al fenomeno di Runge.

Polinomi a tratti

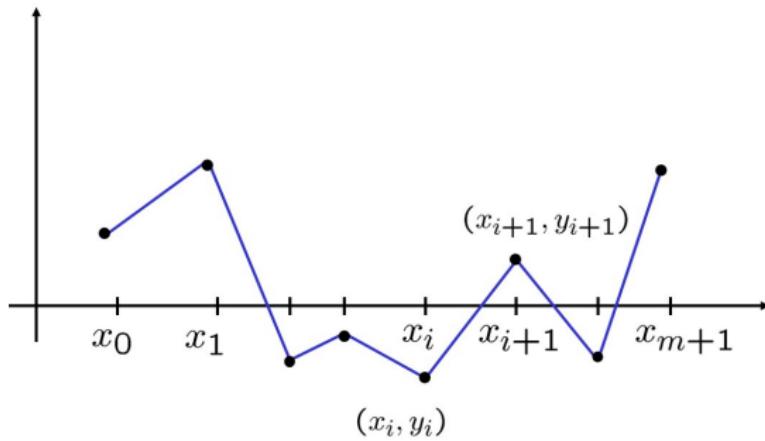
- Invece di costruire una funzione interpolante che sia un globalmente un polinomio, si considerano funzioni polinomiali a tratti, che sono polinomi di un grado fissato se ristrette a ciascun sottointervallo $[x_i, x_{i+1}]$
- Il caso più semplice è quello delle funzioni lineari a tratti



Interpolante lineare a tratti

Fissati $m+2$ punti di interpolazione (x_i, y_i) , $i = 0, \dots, m+1$, si definisce la funzione lineare a tratti $s(x)$ che interpola i punti dati come

$$s(x) = y_i + \underbrace{\frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x - x_i)}_{s_i(x)}, \quad \text{se } x \in [x_i, x_{i+1}], i = 0, \dots, m$$



Errore nell'interpolazione lineare a tratti

- Supponiamo che $y_i = f(x_i)$, $i = 0, \dots, m + 1$.
- Obiettivo: vogliamo studiare la funzione resto $R^s(x) = f(x) - s(x)$.
- Ipotesi: $f \in C^2([x_0, x_{m+1}])$.
- Conviene considerare la funzione resto $R^s(x)$ restringendosi a $[x_i, x_{i+1}]$

In ogni intervallo $[x_i, x_{i+1}]$, $i = 0, \dots, m$, si può applicare il teorema del resto del polinomio di interpolazione con $n = 1$:

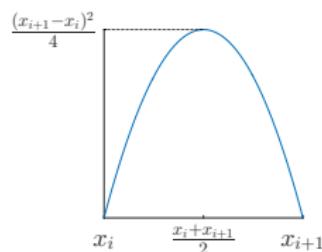
$$f(x) - s_i(x) = \frac{f''(\xi_i)}{2}(x - x_i)(x - x_{i+1}), \text{ con } \xi \in [x_i, x_{i+1}], \forall x \in [x_i, x_{i+1}]$$

Se $|f''(x)| \leq M_2^f \forall x \in [x_0, x_{m+1}]$, allora

$$|f(x) - s_i(x)| \leq \frac{M_2^f}{2} \max_{x \in [x_i, x_{i+1}]} |(x - x_i)(x - x_{i+1})| \quad \forall x \in [x_i, x_{i+1}]$$

Studiamo il termine a destra per $x \in [x_i, x_{i+1}]$:

$$\begin{aligned} |(x - x_i)| \cdot |(x - x_{i+1})| &= (x - x_i)(x_{i+1} - x) \\ &\downarrow \quad x = \frac{x_i + x_{i+1}}{2} \\ \max_{x \in [x_i, x_{i+1}]} |(x - x_i)(x - x_{i+1})| &= \frac{(x_{i+1} - x_i)^2}{4} \end{aligned}$$



$$|f(x) - s_i(x)| \leq \frac{M_2^f}{8} (x_{i+1} - x_i)^2, \forall x \in [x_i, x_{i+1}]$$

Errore nell'interpolazione lineare a tratti(continua)

$$|f(x) - s_i(x)| \leq \frac{M_2^f}{8} (x_{i+1} - x_i)^2, \forall x \in [x_i, x_{i+1}]$$

Indichiamo con h l'ampiezza massima dei sottointervalli:

$$h = \max_{i \in \{0, \dots, m\}} (x_{i+1} - x_i)$$

maggiorando ulteriormente si ottiene

$$|f(x) - s(x)| \leq \frac{M_2^f}{8} h^2, \forall x \in [x_0, x_{m+1}]$$

Teorema

Sia $f \in C^2([a, b])$ e siano $x_0, \dots, x_{m+1} \in [a, b]$. Indichiamo con M_2^f il massimo di $|f''(y)|$ in $[a, b]$ e con h la seguente quantità:

$$h = \max_{i \in \{0, \dots, m\}} (x_{i+1} - x_i)$$

Se $s(x)$ è il polinomio lineare a tratti tale che $s(x_i) = f(x_i)$, $i = 0, \dots, m + 1$, allora si ha

$$\|f - s\|_\infty \leq \frac{M_2^f}{8} h^2$$

$$\|f - s\|_{\infty} \leq \frac{M_2^f}{8} h^2$$

- Caso particolare: partizione uniforme di un intervallo $[a, b]$,

$$x_i = a + i \frac{(b-a)}{m+1}, i = 0, \dots, m+1$$

- Si ha

$$h = \frac{(b-a)}{m+1} \Rightarrow \|f - s\|_{\infty} \leq \frac{M_2^f (b-a)^2}{8} \frac{1}{(m+1)^2}$$

- all'aumentare di m l'errore commesso approssimando f con l'interpolante lineare a tratti $s(x)$ diminuisce come $\frac{1}{(m+1)^2}$
- Con l'interpolazione a tratti NON si verifica il fenomeno di Runge

Svantaggi dell'interpolazione lineare a tratti

La funzione $s(x)$ è continua ma non derivabile. Si perde una caratteristica importante della funzione f .

Si definisce una classe di funzioni polinomiali a tratti molto regolari: lo spazio delle funzioni spline.

Definizione

Sia $[a, b]$ un intervallo e x_0, \dots, x_{m+1} una partizione di esso tale che $x_0 = a$ e $x_{m+1} = b$. Si definisce *spline* di grado n relativa alla partizione x_0, \dots, x_{m+1} ogni funzione $s(x)$ tale che

- 1 la funzione $s(x)$ ristretta all'intervallo $[x_i, x_{i+1}]$ è un polinomio $s_i(x)$ di grado al più n , per $i = 0, \dots, m$
- 2 la funzione s e tutte le sue derivate fino all'ordine $n - 1$ sono continue nell'intervallo $[a, b]$:

$$s \in C^{(n-1)}([a, b])$$

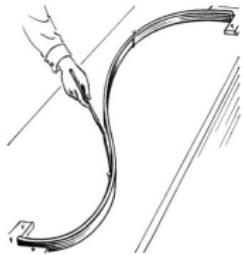
- La funzione s ristretta ad ogni intervallo $[x_i, x_{i+1}]$ è $C^{(n-1)}$ perché è un polinomio;
- La seconda condizione equivale a richiedere che i polinomi $s_i(x)$ soddisfino le seguenti condizioni

$$s_i^{(k)}(x_{i+1}) = s_{i+1}^{(k)}(x_{i+1}), k = 0, \dots, n-1, i = 0, \dots, m-1$$

Cenni storici sulle funzioni splines

Le origini

- Il termine inglese spline(=flessibile) deriva dal nome di un particolare strumento di disegno formato da una striscia di metallo o di legno a cui veniva data la forma desiderata per mezzo di grossi pesi ai quali veniva ancorata. Serviva per realizzare disegni che non era possibile ottenere con riga e compasso, in scala anche molto grande, per esempio per la costruzione di navi.



- Il termine spline per indicare una funzione polinomiale a tratti molto regolare fu utilizzato per la prima volta nel 1946 da Schoenberg.
- Un tipo particolare di funzioni spline, dette B-spline, è alla base dei software per il disegno tecnico (CAD) e della grafica vettoriale.

- Le funzioni lineari a tratti sono un caso particolare della definizione di spline con $n = 1$ e si dicono anche spline lineari.
- L'altro caso di particolare interesse è quello delle spline cubiche ($n = 3$). In questo caso le condizioni di regolarità sono

$$s_i(x_{i+1}) = s_{i+1}(x_{i+1})$$

$$s'_i(x_{i+1}) = s'_{i+1}(x_{i+1})$$

$$s''_i(x_{i+1}) = s''_{i+1}(x_{i+1})$$

per $i = 0, \dots, m - 1$.

Dimensione dello spazio delle funzioni spline

Contiamo da quanti parametri dipende una generica spline di grado n relativa ad una partizione di $m + 2$ punti.

- Ogni ‘pezzo’ di spline $s_i(x)$ è un polinomio di grado $n \Rightarrow$ dipende da $n + 1$ parametri (i suoi coefficienti).

$$s_i(x) = a_{0i} + a_{1i}x + \dots + a_{ni}x^n, \quad i = 0, \dots, m$$

- In tutto ci sono $m + 1$ ‘pezzi’
- In totale una spline dipende da $(n + 1)(m + 1)$ parametri

Le condizioni di regolarità

$$s_i^{(k)}(x_{i+1}) = s_{i+1}^{(k)}(x_{i+1}), \quad k = 0, \dots, n - 1, \quad i = 0, \dots, m - 1$$

sono nm uguaglianze, che fissano altrettanti parametri

Gradi di libertà delle spline di grado n

Una spline di grado n dipende da $(n + 1)(m + 1)$ parametri, di cui

$$(n + 1)(m + 1) - nm = n + m + 1$$

sono liberi. Questo numero corrisponde ai gradi di libertà (o dimensione dello spazio) delle spline di grado n relative ad $m + 2$ punti.

Spline di interpolazione

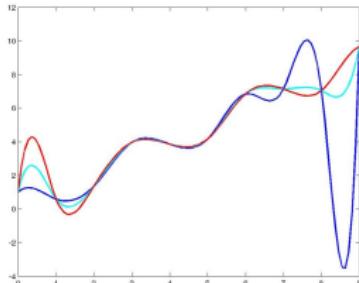
- Obiettivo: utilizzare le funzioni spline per interpolare i dati
- Più precisamente: dati (x_i, y_i) , $i = 0, \dots, m + 1$, calcolare una spline di grado n relativa ai punti x_0, \dots, x_{m+1} tale che

$$s(x_i) = y_i, \quad i = 0, \dots, m + 1$$

Le condizioni di interpolazione sono $m + 2$, pertanto fissano altrettanti parametri. I gradi di libertà rimanenti sono quindi

$$n + m + 1 - (m + 2) = n - 1$$

Eccetto il caso lineare ($n = 1$) non c'è un'unica spline di interpolazione. Per $n > 1$, fissati i punti, esistono infinite spline di grado n che li interpolano, al variare di $n - 1$ parametri liberi.



Costruzione delle spline cubiche di interpolazione

- Dati (x_i, y_i) , $i = 0, \dots, m + 1$, si vuole costruire una spline di grado $n = 3$ relativa ai punti x_0, \dots, x_{m+1} , che interpoli i dati.
- Si vogliono calcolare i coefficienti degli $m + 1$ polinomi di grado 3

$$s_i(x) = \alpha_i + \beta_i(x - x_i) + \gamma_i(x - x_i)^2 + \delta_i(x - x_i)^3, \quad i = 0, \dots, m$$

che soddisfano le condizioni di regolarità e le condizioni di interpolazione

$$\begin{aligned} s_i(x_{i+1}) &= s_{i+1}(x_{i+1}), \quad i = 0, \dots, m - 1 \\ s'_i(x_{i+1}) &= s'_{i+1}(x_{i+1}), \quad i = 0, \dots, m - 1 \\ s''_i(x_{i+1}) &= s''_{i+1}(x_{i+1}), \quad i = 0, \dots, m - 1 \\ s_i(x_i) &= y_i, \quad i = 0, \dots, m + 1 \end{aligned}$$

- Calcolare una spline cubica di interpolazione significa quindi calcolare $\alpha_i, \beta_i, \gamma_i, \delta_i$, per ogni $i = 0, \dots, m$, in funzione dei dati x_i, y_i , utilizzando le condizioni di regolarità e di interpolazione.

Costruzione di una spline cubica di interpolazione (1)

$$s_i(x) = \alpha_i + \beta_i(x - x_i) + \gamma_i(x - x_i)^2 + \delta_i(x - x_i)^3 \Rightarrow \begin{cases} s'_i(x) = \beta_i + 2\gamma_i(x - x_i) + 3\delta_i(x - x_i)^2 \\ s''_i(x) = 2\gamma_i + 6\delta_i(x - x_i) \\ i = 0, \dots, m \end{cases}$$

Si ha: $s(x_i) = s_i(x_i) = \alpha_i$, $s'(x_i) = s'_i(x_i) = \beta_i$, $i = 0, \dots, m$

Poniamo $h_i = x_{i+1} - x_i$, $i = 0, \dots, m$.

$$\begin{cases} s_i(x_{i+1}) &= s_{i+1}(x_{i+1}) \\ s'_i(x_{i+1}) &= s'_{i+1}(x_{i+1}) \\ s''_i(x_{i+1}) &= s''_{i+1}(x_{i+1}) \\ i &= 0, \dots, m-1 \\ s(x_i) &= y_i \\ i &= 0, \dots, m+1 \end{cases} \Rightarrow \begin{cases} \alpha_i + \beta_i h_i + \gamma_i h_i^2 + \delta_i h_i^3 &= y_{i+1} \\ \beta_i + 2\gamma_i h_i + 3\delta_i h_i^2 &= \beta_{i+1} \\ 2\gamma_i + 6\delta_i h_i &= 2\gamma_{i+1} \\ i &= 0, \dots, m-1 \\ \alpha_i &= y_i, \\ i &= 0, \dots, m \end{cases}$$

Introduciamo un ulteriore parametro $\beta_{m+1} = s'(x_{m+1})$:
l'uguaglianza $s_m(x_{m+1}) = y_{m+1}$

$$\begin{cases} \alpha_i + \beta_i h_i + \gamma_i h_i^2 + \delta_i h_i^3 &= y_{i+1} \\ \beta_i + 2\gamma_i h_i + 3\delta_i h_i^2 &= \beta_{i+1} \\ i &= 0, \dots, m \\ 2\gamma_i + 6\delta_i h_i &= 2\gamma_{i+1} \\ i &= 0, \dots, m-1 \end{cases} \Rightarrow \begin{cases} \beta_i h_i + \gamma_i h_i^2 + \delta_i h_i^3 &= y_{i+1} - y_i \\ \beta_i + 2\gamma_i h_i + 3\delta_i h_i^2 &= \beta_{i+1} \\ i &= 0, \dots, m \\ \gamma_i + 3\delta_i h_i &= \gamma_{i+1} \\ i &= 0, \dots, m-1 \end{cases}$$

Costruzione di una spline cubica di interpolazione (2)

$$\left\{ \begin{array}{lcl} \beta_i h_i + \gamma_i h_i^2 + \delta_i h_i^3 & = & y_{i+1} - y_i \\ \beta_i + 2\gamma_i h_i + 3\delta_i h_i^2 & = & \beta_{i+1} \\ i = 0, \dots, m \\ \gamma_i + 3\delta_i h_i & = & \gamma_{i+1} \\ i = 0, \dots, m-1 \end{array} \right.$$

Dalle prime 2 equazioni si ricavano γ_i, δ_i in funzione di β_i, β_{i+1} e dei dati.

$$\Rightarrow \left\{ \begin{array}{lcl} \gamma_i h_i^2 + \delta_i h_i^3 & = & y_{i+1} - y_i - \beta_i h_i \\ 2\gamma_i h_i + 3\delta_i h_i^2 & = & \beta_{i+1} - \beta_i \end{array} \right. \Rightarrow \left\{ \begin{array}{lcl} \gamma_i h_i + \delta_i h_i^2 & = & \frac{y_{i+1} - y_i}{h_i} - \beta_i \\ 2\gamma_i h_i + 3\delta_i h_i^2 & = & \beta_{i+1} - \beta_i \end{array} \right.$$
$$\Rightarrow \left\{ \begin{array}{lcl} \delta_i & = & \frac{1}{h_i^2} \left(\beta_{i+1} + \beta_i - 2\frac{y_{i+1} - y_i}{h_i} \right) \\ \gamma_i & = & \frac{1}{h_i} \left(3\frac{y_{i+1} - y_i}{h_i} - (\beta_{i+1} + 2\beta_i) \right) \end{array} \right. \quad i = 0, \dots, m$$

Infine si sostituiscono le espressioni di δ_i, γ_i nella terza:

$$\frac{1}{h_i} \left(3\frac{y_{i+1} - y_i}{h_i} - (\beta_{i+1} + 2\beta_i) \right) + 3\frac{1}{h_i^2} \left(\beta_{i+1} + \beta_i - 2\frac{y_{i+1} - y_i}{h_i} \right) h_i =$$
$$= \frac{1}{h_{i+1}} \left(3\frac{y_{i+2} - y_{i+1}}{h_{i+1}} - 2(\beta_{i+2} + \beta_{i+1}) \right)$$
$$i = 0, \dots, m-1$$

Calcolo di una spline cubica di interpolazione (3)

$$\begin{aligned} \frac{1}{h_i} \left(3 \frac{y_{i+1} - y_i}{h_i} - (\beta_{i+1} + 2\beta_i) \right) + 3 \frac{1}{h_i} \left(\beta_{i+1} + \beta_i - 2 \frac{y_{i+1} - y_i}{h_i} \right) = \\ = \frac{1}{h_{i+1}} \left(3 \frac{y_{i+2} - y_{i+1}}{h_{i+1}} - 2(\beta_{i+2} + \beta_{i+1}) \right) \\ i = 0, \dots, m-1 \end{aligned}$$

Portando a sinistra i termini che contengono le incognite ' β ', a destra i termini noti e moltiplicando tutto per $h_i h_{i+1}$ si ottiene

$$h_{i+1}\beta_i + 2(h_i + h_{i+1})\beta_{i+1} + h_i\beta_{i+2} = 3h_{i+1} \left(\frac{y_{i+1} - y_i}{h_i} \right) + 3h_i \left(\frac{y_{i+2} - y_{i+1}}{h_{i+1}} \right)$$
$$i = 0, \dots, m-1$$

Si tratta di un sistema di m equazioni lineari nelle $m+2$ incognite β_i , $i = 0, \dots, m+1$

$$\left\{ \begin{array}{lllll} h_1\beta_0 & +2(h_0 + h_1)\beta_1 & +h_0\beta_2 & & = b_0 \\ h_2\beta_1 & & +2(h_1 + h_2)\beta_2 & +h_1\beta_3 & = b_1 \\ & \ddots & & & \\ h_m\beta_{m-1} & & +2(h_{m-1} + h_m)\beta_m & +h_{m-1}\beta_{m+1} & = b_{m-1} \end{array} \right.$$

$$b_i = 3h_{i+1} \left(\frac{y_{i+1} - y_i}{h_i} \right) + 3h_i \left(\frac{y_{i+2} - y_{i+1}}{h_{i+1}} \right), i = 0, \dots, m-1$$

- Non c'è un'unica spline cubica di interpolazione, ma ne esiste una famiglia infinita che dipende da due parametri
- Per individuare un'unica spline nella famiglia, ci si restringe ad un sottoinsieme, aggiungendo due condizioni, per esaurire i gradi di libertà.

Spline cubica vincolata di interpolazione

$$h_{i+1}\beta_i + 2(h_i + h_{i+1})\beta_{i+1} + h_i\beta_{i+2} = b_i, \quad i = 0, \dots, m-1$$

- Si assegnano i valori β_0 e β_{m+1} che corrispondono alla derivata prima della spline negli estremi dell'intervallo di interpolazione.
- Si riscrive il sistema portando i valori assegnati di β_0 e β_{m+1} al termine noto

Il sistema diventa

$$Tz = c, \quad \text{dove}$$

$$T = \begin{pmatrix} 2(h_0 + h_1) & h_0 & & & \\ h_2 & 2(h_1 + h_2) & h_1 & & \\ & & \ddots & \ddots & \\ & & & \ddots & h_{m-2} \\ & & h_m & 2(h_{m-1} + h_m) & \end{pmatrix}$$

$$c = \begin{pmatrix} b_0 - h_1\beta_0 \\ b_1 \\ \vdots \\ b_{m-1} - h_{m-1}\beta_{m+1} \end{pmatrix}, \quad z = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_m \end{pmatrix}$$

Osservazioni

$$T = \begin{pmatrix} 2(h_0 + h_1) & h_0 & & & \\ h_2 & 2(h_1 + h_2) & h_1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & h_{m-2} & \\ h_m & & h_m & 2(h_{m-1} + h_m) & \end{pmatrix}$$

- La matrice $T \in \mathbb{R}^{m \times m}$ è strettamente diagonale dominante per righe
- Il sistema $Tz = c$ ha un'unica soluzione.
- Dopo aver calcolato la soluzione del sistema, ottenendo i ‘ β ’, per sostituzione si ottengono anche i ‘ γ ’ e i ‘ δ ’.
- In questo modo di calcola l'unica spline cubica vincolata che intercala i punti dati.

Altri tipi di spline cubiche

- Spline cubica naturale:
 - si impongono le condizioni $s''(x_0) = s''(x_{m+1}) = 0$
- Spline cubica periodica (si utilizzano quando $y_0 = y_{m+1}$):
 - si impongono le condizioni $s'(x_0) = s'(x_{m+1})$ e $s''(x_0) = s''(x_{m+1})$
- Spline cubica not-a-knot (Matlab)
 - è un polinomio di grado 3 se ristretta agli intervalli $[x_0, x_2]$, $[x_{m-1}, x_{m+1}]$.

Fissati i punti di interpolazione, esiste ed è unica la spline cubica di interpolazione in ciascuno dei casi precedenti.

Teoremi sulle spline cubiche: proprietà di minima curvatura

Teorema

Tra tutte le funzioni $f \in C^2([a, b])$ che interpolano i punti (x_i, y_i) , $i = 0, \dots, m + 1$ e che soddisfano una delle seguenti condizioni:

- ① $f'(x_0) = \beta_0$, $f'(x_{m+1}) = \beta_{m+1}$, con β_0, β_{m+1} valori fissati;
- ② $f''(x_0) = f''(x_{m+1}) = 0$
- ③ (se $y_0 = y_{m+1}$) $f'(x_0) = f'(x_{m+1})$ e $f''(x_0) = f''(x_{m+1})$

la spline cubica di interpolazione vincolata, naturale o periodica è quella per cui vale la proprietà di minimo

$$\int_a^b (s''(x))^2 dx \leq \int_a^b (f''(x))^2 dx$$

e l'uguaglianza vale solo se $f = s$.

Significato: tra tutte le funzioni che interpolano gli stessi punti, le spline cubiche sono quelle il cui grafico presenta le curve più ‘dolci’, senza bruschi cambiamenti di pendenza.

Teoremi sulle spline cubiche: errore di interpolazione

Teorema

Sia $s(x)$ la spline cubica vincolata, naturale o periodica che intercala i punti (x_i, y_i) , $i = 0, \dots, m + 1$, dove $y_i = f(x_i)$, con $f \in C^2([a, b])$. Allora, per ogni $x \in [a, b]$ si ha

$$\begin{aligned}|f(x) - s(x)| &\leq h^{\frac{3}{2}} \left(\int_a^b (f''(x))^2 dx \right)^{\frac{1}{2}} \\|f'(x) - s'(x)| &\leq h^{\frac{1}{2}} \left(\int_a^b (f''(x))^2 dx \right)^{\frac{1}{2}}\end{aligned}$$

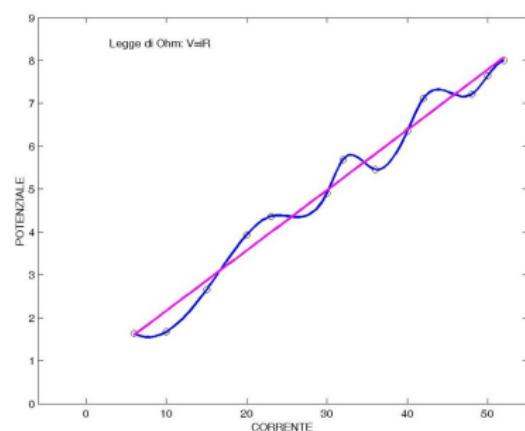
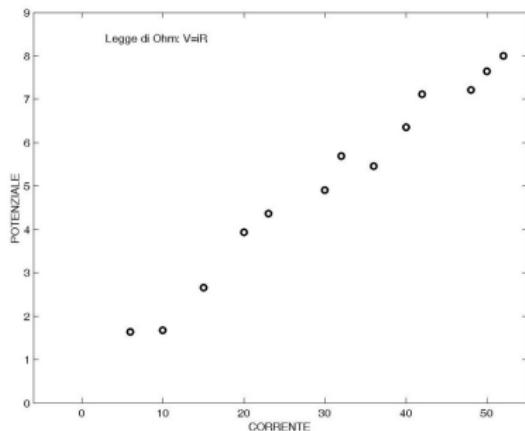
dove $h = \max_{i \in \{0, \dots, m\}} |x_{i+1} - x_i|$.

- Le spline cubiche di interpolazione approssimano la funzione interpolata e, se l'ampiezza degli intervalli della partizione diminuisce, l'accuratezza aumenta (non si verifica il fenomeno di Runge)
- La derivata prima della spline cubica di interpolazione approssima la derivata prima della funzione interpolata.

Approssimazione di dati e funzioni: il criterio dei minimi quadrati

Svantaggi dell'approssimazione di dati mediante interpolazione

- Nelle applicazioni, le tecniche di approssimazione di dati servono per definire il **modello matematico** che descrive un fenomeno osservabile.
- Il caso più semplice si ha quando si cerca di individuare uno o più **parametri** della formula che lega due quantità, a partire da un numero finito di misurazioni effettuate. Ad esempio:
 - si fa variare la differenza di V potenziale tra due capi di un filo elettrico, misurandola con un voltmetro, misurando la corrente generata i (con un amperometro), per determinare la resistenza R del filo;
 - la legge di Ohm, $V = iR$, che descrive il fenomeno è lineare, ma le misurazioni effettuate contengono una certa quantità di **errore**



Interpolare i dati con errore non fornisce un modello significativo del fenomeno osservato.

La regressione lineare

- Nell'esempio relativo alla legge di Ohm, è noto che il modello che descrive il fenomeno è di tipo lineare
- Tra tutte le rette di equazione

$$y = a_0 + a_1 x$$

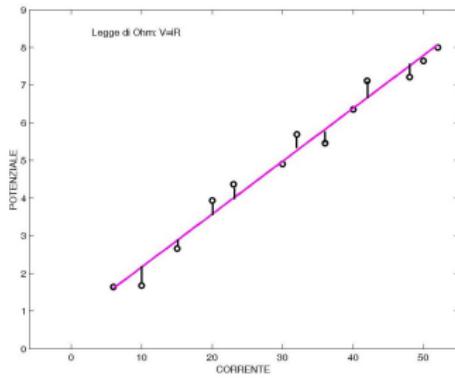
si vuole determinare quella che 'spiega' meglio i dati

$$(x_i, y_i), i = 1, \dots, m$$

Criterio dei minimi quadrati: retta di regressione

Dati $(x_i, y_i), i = 1, \dots, m$, si determinano i parametri del modello scelto $y = a_0 + a_1 x$ che minimizzano la funzione

$$Q(a_0, a_1) = \sum_{i=1}^m (a_0 + a_1 x_i - y_i)^2$$



- Il criterio dei minimi quadrati porta ad individuare i due parametri di una retta

$$f(a_0, a_1; x) = a_0 + a_1 x$$

la cui ‘distanza cumulativa’ dai dati è minima;

- Il tipo di modello scelto, in questo caso la retta, è fissato a priori, solitamente in base alla conoscenza delle caratteristiche del fenomeno che si vuole descrivere.

Lo stesso approccio si estende a modelli più generali.

- Dati: (x_i, y_i) , $i = 1, \dots, m$
- Ipotesi: si assume che i dati corrispondano ad un modello che dipende da n parametri

$$f(a_0, \dots, a_{n-1}; x)$$

- Obiettivo: si vogliono determinare i parametri del modello che minimizzano la funzione

$$Q(a_0, \dots, a_{n-1}) = \sum_{i=1}^m (f(a_0, \dots, a_{n-1}; x_i) - y_i)^2$$

Il problema dei minimi quadrati: la retta di regressione in forma matriciale

$$Q(a_0, a_1) = \sum_{i=1}^m (a_0 + a_1 x_i - y_i)^2$$

Introduciamo i vettori

$$q(a_0, a_1) = \begin{pmatrix} a_0 + a_1 x_1 \\ a_0 + a_1 x_2 \\ \vdots \\ a_0 + a_1 x_m \end{pmatrix} \in \mathbb{R}^m \text{ e } y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}.$$

Allora, per definizione

$$Q(a_0, a_1) = \|q(a_0, a_1) - y\|^2$$

Inoltre, se definiamo

$$A = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_m \end{pmatrix} \in \mathbb{R}^{m \times 2} \text{ e } \alpha = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} \in \mathbb{R}^2$$

si ha

$$q(a_0, a_1) = A\alpha \Rightarrow Q(a_0, a_1) = \|A\alpha - y\|^2$$

Il problema dei minimi quadrati: caso polinomiale

Supponiamo di aver scelto un modello polinomiale con n parametri:

$$\begin{aligned} f(a_0, a_1, \dots, a_{n-1}; x) &= a_0 + a_1 x + \dots + a_{n-1} x^{n-1} \\ &\Downarrow \\ Q(a_0, a_1, \dots, a_{n-1}) &= \sum_{i=1}^m (a_0 + a_1 x_i + \dots + a_{n-1} x_i^{n-1} - y_i)^2 \end{aligned}$$

Definiamo

$$A = \begin{pmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{n-1} \\ \vdots & \vdots & & & \\ 1 & x_m & x_m^2 & \dots & x_m^{n-1} \end{pmatrix} \in \mathbb{R}^{m \times n} \text{ e } \alpha = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} \in \mathbb{R}^n$$

Allora si ha

$$Q(a_0, \dots, a_{n-1}) = \|A\alpha - y\|^2$$

Osservazione: il caso $n = m$ corrisponde al polinomio di interpolazione.

Il problema dei minimi quadrati: caso generale lineare

Supponiamo di aver scelto un modello con n parametri come combinazione lineare di un insieme di n funzioni di base $\{\phi_i\}_{i=0}^{n-1}$

$$\begin{aligned} f(a_0, a_1, \dots, a_{n-1}; x) &= a_0\phi_0(x) + a_1\phi_1(x) + \dots + a_{n-1}\phi_{n-1}(x) \\ &\Downarrow \\ Q(a_0, a_1, \dots, a_{n-1}) &= \sum_{i=1}^m (a_0\phi_0(x_i) + a_1\phi_1(x_i) + \dots + a_{n-1}\phi_{n-1}(x_i) - y_i)^2 \end{aligned}$$

Definiamo

$$A = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \phi_2(x_1) & \dots & \phi_{n-1}(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \phi_2(x_2) & \dots & \phi_{n-1}(x_2) \\ \vdots & \vdots & & & \\ \phi_0(x_m) & \phi_1(x_m) & \phi_2(x_m) & \dots & \phi_{n-1}(x_m) \end{pmatrix} \in \mathbb{R}^{m \times n} \text{ e } \alpha = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} \in \mathbb{R}^n$$

Allora si ha

$$Q(a_0, \dots, a_{n-1}) = \|A\alpha - y\|^2$$

Calcolo della soluzione del problema dei minimi quadrati

Dati (x_i, y_i) , $i = 1, \dots, m$, e fissato il modello

$$f(a_0, a_1, \dots, a_{n-1}; x) = a_0\phi_0(x) + a_1\phi_1(x) + \dots + a_{n-1}\phi_{n-1}(x)$$

occorre risolvere il problema

$$\min_{\alpha \in \mathbb{R}^n} \|A\alpha - y\|^2$$

- si può affrontare mediante opportune fattorizzazioni della matrice $A \in \mathbb{R}^{m \times n}$
- descriviamo nel seguito due procedure, che si applicano rispettivamente ai due casi:
 - *caso non degenere*: $A \in \mathbb{R}^{m \times n}$, con $n \leq m$, e le colonne sono linearmente indipendenti (la regressione polinomiale rientra in questo caso)
 - *caso generale*: $A \in \mathbb{R}^{m \times n}$, con $n \leq m$, e k colonne sono linearmente indipendenti, con $k \leq n$.

Caso non degenere

Teorema

Se $A \in \mathbb{R}^{m \times n}$ con $n \leq m$ ha rango n , allora la soluzione del problema

$$\min_{\alpha \in \mathbb{R}^n} \|A\alpha - y\|^2$$

è unica.

La dimostrazione è costruttiva e fornisce un algoritmo per calcolare la soluzione. Per prima cosa si osserva che:

- Se Q è ortogonale, allora $\|Qx\|^2 = x^T Q^T Qx = x^T x = \|x\|^2$;
- Se $z = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \in \mathbb{R}^m$ con $z_1 \in \mathbb{R}^n$, $z_2 \in \mathbb{R}^{m-n}$, allora $\|z\|^2 = \|z_1\|^2 + \|z_2\|^2$
- Se $A \in \mathbb{R}^{m \times n}$ con $n \leq m$ ha rango n , allora esistono $Q \in \mathbb{R}^{m \times m}$ ortogonale e $R \in \mathbb{R}^{n \times n}$ triangolare superiore nonsingolare tali che

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix} \iff Q^T A = \begin{pmatrix} R \\ 0 \end{pmatrix}$$

Dalle precedenti osservazioni si ha:

$$\|A\alpha - y\|^2 = \|Q^T A\alpha - \underbrace{Q^T y}_{\tilde{y}}\|^2 = \left\| \begin{pmatrix} R \\ 0 \end{pmatrix} \alpha - \begin{pmatrix} \tilde{y}_1 \\ \tilde{y}_2 \end{pmatrix} \right\|^2 = \|R\alpha - \tilde{y}_1\|^2 + \|\tilde{y}_2\|^2$$

Il minimo di $\|A\alpha - y\|^2$ si ha per $R\alpha = \tilde{y}_1$. Essendo R nonsingolare, la soluzione è unica.

Caso non degenere: calcolo della soluzione

Riassumendo i passaggi della dimostrazione, la soluzione del problema dei minimi quadrati

$$\min_{\alpha \in \mathbb{R}^n} \|A\alpha - y\|^2$$

quando $A \in \mathbb{R}^{m \times n}$ con $n \leq m$ ha rango n si può calcolare nel modo seguente.

- ① Calcolare la fattorizzazione

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix}$$

mediante trasformazioni di Householder

- ② Calcolare $\tilde{y} = Q^T y$.
- ③ Estrarre le prime n componenti di \tilde{y} e memorizzarle nel vettore \tilde{y}_1
- ④ Risolvere il sistema triangolare superiore $R\alpha = \tilde{y}_1$.

- Se le colonne di A non sono linearmente indipendenti, ossia se il rango di A è k con $k < n$, allora il problema dei minimi quadrati ha infinite soluzioni, che dipendono da $n - k$ parametri liberi.
- Nel sottospazio delle soluzioni, si individua quella che ha norma euclidea più piccola, detta *soluzione di norma minima*
- Per la dimostrazione dell'esistenza e unicità della soluzione di norma minima, che definisce anche l'algoritmo per calcolarla, si utilizza una diversa fattorizzazione della matrice A

La decomposizione ai valori singolari

Singular Value Decomposition (SVD)

Teorema

Sia $A \in \mathbb{R}^{m \times n}$ una matrice di rango k . Allora A si può fattorizzare nella forma

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

dove

- $U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times n}$ sono matrici ortogonali
- $\Sigma \in \mathbb{R}^{m \times n}$ è una matrice rettangolare diagonale della forma

$$\Sigma = \begin{pmatrix} \sigma_1 & & & 0 & \dots & 0 \\ & \sigma_2 & & 0 & \dots & 0 \\ & & \ddots & & & \\ 0 & \dots & \dots & \sigma_k & 0 & \dots & 0 \\ \dots & & & 0 & 0 & \dots & 0 \\ 0 & \dots & \dots & 0 & 0 & \dots & 0 \end{pmatrix}$$

con $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_k > 0$.

I valori σ_i sono detti i *valori singolari* di A . Si ha che σ_i^2 sono gli autovalori di $A^T A$, mentre le colonne di U e di V sono gli autovettori di AA^T e di $A^T A$ rispettivamente.

Problema dei minimi quadrati - caso generale

Teorema

Se $A \in \mathbb{R}^{m \times n}$ con $n \leq m$ ha rango $k \leq n$, allora la soluzione di minima norma del problema

$$\min_{\alpha \in \mathbb{R}^n} \|A\alpha - y\|^2$$

è unica.

Dimostrazione. Applicando il teorema di fattorizzazione SVD si ha $A = U\Sigma V^T$. Sostituendo,

$$\|A\alpha - y\|^2 = \|U\Sigma V^T \alpha - y\|^2 = \|\underbrace{\Sigma V^T \alpha}_\gamma - \underbrace{U^T y}_z\|^2 = \left\| \begin{pmatrix} \sigma_1 \gamma_1 - z_1 \\ \vdots \\ \sigma_k \gamma_k - z_k \\ -z_{k+1} \\ \vdots \\ -z_m \end{pmatrix} \right\|^2$$

Le infinite soluzioni si trovano per $\gamma_i = z_i/\sigma_i$, $i = 1, \dots, k$, al variare di $\gamma_{k+1}, \dots, \gamma_n$ in \mathbb{R}^{n-k} .

$$\gamma = V^T \alpha \Rightarrow \alpha = V\gamma = V \begin{pmatrix} z_1/\sigma_1 \\ \vdots \\ z_k/\sigma_k \\ \gamma_{k+1} \\ \vdots \\ \gamma_n \end{pmatrix} \Rightarrow \|\alpha\|^2 = \|\gamma\|^2 = \frac{z_1^2}{\sigma_1^2} + \dots + \frac{z_k^2}{\sigma_k^2} + \gamma_{k+1}^2 + \dots + \gamma_n^2$$

La soluzione di norma minima si ottiene per $\gamma_{k+1} = \dots = \gamma_n = 0$.

Calcolo della soluzione di minima norma

$$\alpha = V\gamma = V \begin{pmatrix} z_1/\sigma_1 \\ \vdots \\ z_k/\sigma_k \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Dati A, y

- ① Calcolare la fattorizzazione SVD di A : $A = U\Sigma V^T$
- ② Calcolare $z = U^T y$

$$\text{③ Calcolare } \gamma = \begin{pmatrix} z_1/\sigma_1 \\ \vdots \\ z_k/\sigma_k \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

- ④ Calcolare $\alpha = V\gamma$.

- Il calcolo della SVD è molto costoso.
- Esistono diversi algoritmi per calcolare la SVD con complessità differenti. In generale procedono in due fasi: prima si riduce la matrice in una forma più semplice (bidiagonale), poi si procede con metodi iterativi.
- Nel caso non degenere, è più conveniente utilizzare la QR, al costo di $n^2(3m - n)/3$.

Applicazioni della SVD

Benchè costosa, la SVD è utilizzata in tantissime applicazioni del mondo reale, poichè fornisce uno strumento per approssimare le matrici

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

Indichiamo con u_i , $i = 1, \dots, m$ la i -esima colonna di U e con v_i , $i = 1, \dots, n$ la i -esima colonna di V . Dalla definizione di prodotto matriciale si ottiene

$$A = \sigma_1 u_1 v_1^T + \dots + \sigma_k u_k v_k^T$$

- I termini $u_i v_i^T$ sono matrici di dimensione $m \times n$
- A si scrive come somma pesata di matrici in cui i coefficienti, i valori singolari, sono in ordine decrescente.

Applicazioni della SVD

Regolarizzazione di problemi malcondizionati

Supponiamo di voler risolvere un sistema lineare

$$Ax = b$$

con $A \in \mathbb{R}^{n \times n}$

- Il numero di condizionamento in norma 2 è $k(A) = \frac{\sigma_1}{\sigma_n}$
- Nei casi di malcondizionamento, si ha $\sigma_1 \gg \sigma_n$.
- Una strategia per ottenere una matrice meno malcondizionata, ma non troppo diversa da A consiste nell'individuare un $\bar{k} < n$ e definire

$$\tilde{A} = \sigma_1 u_1 v_1^T + \dots + \sigma_{\bar{k}} u_{\bar{k}} v_{\bar{k}}^T$$

e risolvere il sistema $\tilde{A}x = b$

Laboratorio sui minimi quadrati

Il problema dei minimi quadrati: caso generale lineare

Supponiamo di aver scelto un modello con n parametri come combinazione lineare di un insieme di n funzioni di base $\{\phi_i\}_{i=0}^{n-1}$

$$\begin{aligned} f(a_0, a_1, \dots, a_{n-1}; x) &= a_0\phi_0(x) + a_1\phi_1(x) + \dots + a_{n-1}\phi_{n-1}(x) \\ &\Downarrow \\ Q(a_0, a_1, \dots, a_{n-1}) &= \sum_{i=1}^m (a_0\phi_0(x_i) + a_1\phi_1(x_i) + \dots + a_{n-1}\phi_{n-1}(x_i) - y_i)^2 \end{aligned}$$

Definiamo

$$A = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \phi_2(x_1) & \dots & \phi_{n-1}(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \phi_2(x_2) & \dots & \phi_{n-1}(x_2) \\ \vdots & \vdots & & & \\ \phi_0(x_m) & \phi_1(x_m) & \phi_2(x_m) & \dots & \phi_{n-1}(x_m) \end{pmatrix} \in \mathbb{R}^{m \times n} \text{ e } \alpha = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} \in \mathbb{R}^n$$

Allora si ha

$$Q(a_0, \dots, a_{n-1}) = \|A\alpha - y\|^2$$

Laboratorio: minimi quadrati caso lineare generalizzato

$$f^\nu(a_1, \dots, a_q, b_1, \dots, b_q; x) = \sum_{j=1}^q b_j \sin(2\pi\nu jx) + a_j \cos(2\pi\nu jx)$$

per ν fissato.

Numero di parametri: $n = 2q$.

Matrice di regressione:

$$A = \begin{pmatrix} \sin(2\pi\nu x_1) & \dots & \sin(2\pi\nu q x_1) & \cos(2\pi\nu x_1) & \dots & \cos(2\pi\nu q x_1) \\ \sin(2\pi\nu x_2) & \dots & \sin(2\pi\nu q x_2) & \cos(2\pi\nu x_2) & \dots & \cos(2\pi\nu q x_2) \\ \vdots & & \vdots & & \vdots & \vdots \\ \sin(2\pi\nu x_m) & \dots & \sin(2\pi\nu q x_m) & \cos(2\pi\nu x_m) & \dots & \cos(2\pi\nu q x_m) \end{pmatrix}$$

$$\alpha = \begin{pmatrix} b_1 \\ \vdots \\ b_q \\ a_1 \\ \vdots \\ a_q \end{pmatrix}$$