

CalcNum-merged

0. Introduzione

Branca della matematica che si occupa di dare strumenti per il calcolo numerico di quantità continue (soluzioni di un'equazione)

Example

$x^2 = 2$ soluzione $\alpha > 0$

$\alpha = \sqrt{2}$ (soluzione simbolica)

α limite dell'iterazione

$$x_{k+1} = \frac{x_k^2 + 2}{2x_k}$$

Calcolo Simbolico VS Calcolo Numerico

Example

$P(x) = 0$ con $p(x) = x^3 + x^2 - x + 1$ ha un'unica soluzione α

- $\alpha = \frac{1}{3} \left(-1 - \frac{4}{\sqrt[3]{19-3\sqrt{33}}} - \sqrt[3]{19-3\sqrt{33}} \right)$ soluzione simbolica
- α è il limite dell'iterazione:

$$x_{k+1} = x_k - \frac{p(x_k)}{p'(x_k)}, \quad x_0 = 2$$

Example

Eq: $x^5 + x^4 - x + 1 = 0$ unica soluzione α

- α non esprimibile in forma chiusa con simboli noti (soluzione simbolica)
- α è il limite dell'iterazione:

$$x_{k+1} = x_k - \frac{p(x_k)}{p'(x_k)}, \quad x_0 = -2$$

Quasi tutte le civiltà hanno una formula per le equazioni di 2° grado.

Nel XVI secolo sono state trovate per le equazioni di 3° e 4° grado (Tartaglia, Cardano, Ferrari).

Important

Teorema di Ruffini-Abel:

Non esiste una formula per le equazioni di 5° grado

La materia si occupa di dare strumenti teorici (teorie matematiche) per il calcolo numerico di quantità continue.

Si cerca di trovare una soluzione ai problemi continui che sia utilizzabile. Non basta una formula, ma serve un algoritmo che fornisca una soluzione in un tempo ragionevole tenendo conto che il calcolatore lavora in quantità finite. Vogliamo quindi trattare **quantità infinite** con **strumenti finiti**.

Quantità Continue

- Una o più equazioni non lineari, radici di polinomi
- Calcolo quantità algebra lineare, soluzione di sistemi lineari
- Determinanti, autovalori/vettori, rango, nucleo, immagine
- Calcolo degli integrali
- Approssimazione di funzioni

Per ogni problema esistono una o più soluzioni.

Efficienza

Gli algoritmi devono avere importanti requisiti:

- Basso costo computazionale
- [Stabilità numerica](#)

1. Stabilità numerica

Esempio

Sviluppo in serie di Taylor di e^x con resto di Lagrange

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots = \sum_{k=0}^n \frac{x^k}{k!} = e^{\epsilon} \frac{x^{n+1}}{(n+1)!}$$

Si hanno due algoritmi per il calcolo di e^x

$$e^x \approx \sum_{k=0}^n \frac{x^k}{k!}$$

$$e^x \approx \frac{1}{e^{-x}} \approx \frac{1}{\sum_{k=0}^n \frac{(-x)^k}{k!}}$$

Il primo funziona meglio con esponente positivo, il secondo con esponente negativo

Relazioni Con Altre Discipline

Matematica Discreta

Si cerca di utilizzare una sola formula per la risoluzione di problemi, mentre qua ci servono più formule

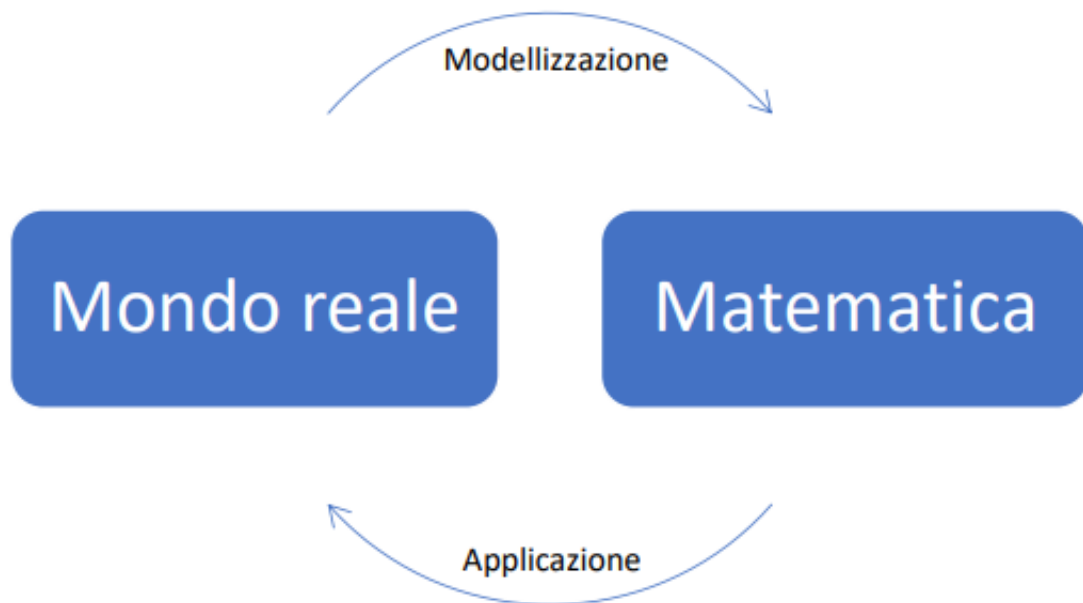
Algoritmi E Strutture Dati

Si occupa dello studio di problemi discreti, mentre qui ci si occupa di problemi continui

Relazione Con Il Mondo Reale

I problemi del mondo reale possono essere **modellizzati** in teorie matematiche.

Le soluzioni matematiche vengono **applicate** nel mondo reale.



2. Error Analysis

I numeri reali per la maggior parte contengono infinite informazioni, a parte i pochi numeri razionali che ne contengono di limitate.

Numeri Razionali

Sono dei numeri che rispettano la descrizione di un determinato teorema, e che non rispettano altri teoremi come, ad esempio, il teorema degli zeri o quello di Lagrange.

I numeri razionali sono infiniti, ma noi possiamo lavorare solo con insiemi finiti di numeri.

? Problema

Selezionare un numero finito di valori che possano rappresentare i numeri reali.

Distribuzione uniforme

Scegliere $\varepsilon > 0$, N pari:

$$\varepsilon = \left\{ \varepsilon k : -\frac{N}{2} < k \leq \frac{N}{2}, k \in \mathbb{Z} \right\}$$

Possiamo rappresentare i numeri reali con l'insieme:

$$S = \left\{ -\varepsilon \frac{N}{2} < x \leq \varepsilon \frac{N}{2} \right\} \subset N$$

Errore Assoluto Vs Relativo

- Errore assoluto: $\tilde{x} - x$
- Errore relativo: $\frac{\tilde{x} - x}{x}, x \neq 0$

Problemi Con L'errore Assoluto

La qualità dell'approssimazione dell'errore assoluto non è sempre la stessa, ad esempio per i numeri più piccoli c'è un errore più grande rispetto a quelli più grandi.

Con l'errore relativo le cifre errate nella rappresentazione dei numeri sono sempre le stesse.

A noi serve sapere solo alcune cifre, quindi è meglio utilizzare una rappresentazione dei numeri con l'errore relativo migliore.

Numeri Reali

Limiti di sequenze di numeri razionali che possono essere approssimate.

Data una base di numerazione $\beta \geq 2$ posso prendere un numero reale tra 0 e 1 e le sequenze di cifre

$$\mathbb{R} \iff \{d_i\}_{i=1,2,3,\dots}, \quad d_i \in \{0, \dots, \beta - 1\}$$

Teorema Della Rappresentazione In Basi

Dato $x \in \mathbb{R} \setminus \{0\}$ e data una base di numerazione $\beta \geq 2$ esiste un unico $p \in \mathbb{Z}$ e una sequenza $\{d_i\}_{i=1,2,3,\dots}$ tali che:

1. $d_i \in \{0, 1, \dots, \beta - 1\}$
2. $d_1 \neq 0$
3. d_i non è definitivamente uguale a $\beta - 1$ così che

$$x = \text{sign}(x) \beta^p \sum_{i=1}^{\infty} \beta^{-i} d_i$$

Il numero $\sum_{i=1}^{\infty} \beta^{-i} d_i$ si dice **mantissa**

Floating Point

Data una base di numerazione $\beta \geq 2$, il numero $t > 0$ di cifre della mantissa m , M numeri positivi, definiamo un insieme di **floating point numbers**

$$F(\beta, t, m, M) = \{0\} \cup \left\{ \pm \beta^p \sum_{i=1}^t \beta^{-i} d_i, -m \leq p \leq M \right\}$$

con $0 \leq d_i < \beta$ intero per $i = 1, \dots, t$, $d_1 \neq 0$

Esempio

$$F := F(10, 2, 2, 3)$$

Cardinalità: $1 + 2(m + M + 1)(\beta - 1)\beta^{t-1}$

$$\text{Max}(F): \Omega = \beta^M \sum_{i=1}^t \beta^{-i} (\beta - 1) = \beta^M (1 - \beta^{-t})$$

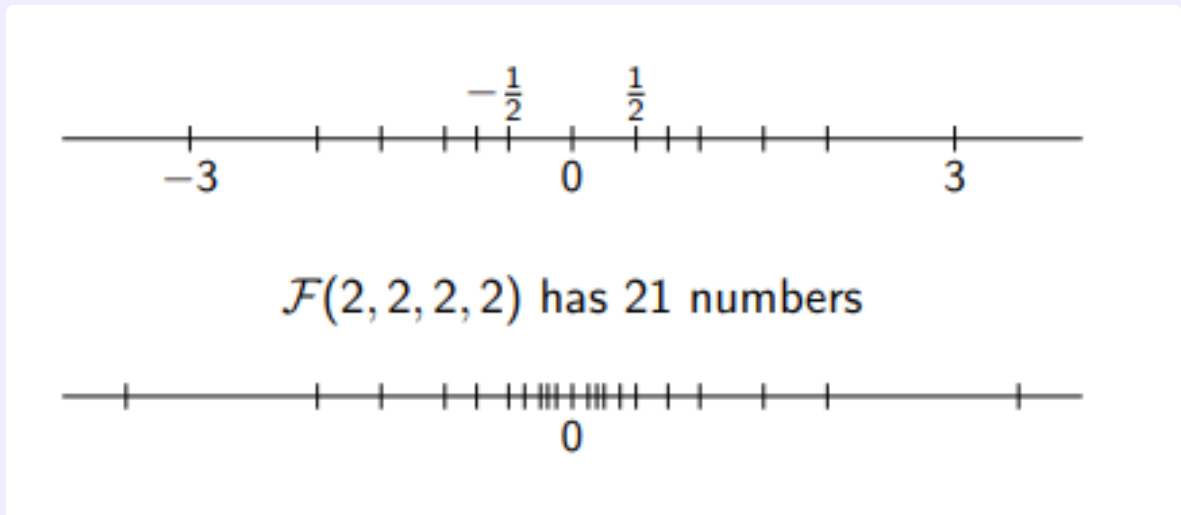
$$\text{MinPositivo}(F): \omega = \beta^{-m} \beta^{-1} = \beta^{-m-1}$$

Per $x \geq \Omega$ i numeri non possono essere rappresentati, mentre per $0 < x < \omega$ c'è un grande errore relativo

Esempio 2 (Utile per dopo)

$F(2, 2, 1, 1)$ è composto da 13 numeri

Questi numeri **non sono uniformi**. Tra $\frac{1}{2}$ e $\frac{1}{4}$ e tra $\frac{1}{2}$ e 1 c'è lo stesso numero di elementi di F .



Data $S = \{x \in \mathbb{R} : \omega \leq x \leq \Omega\}$ costruiamo una **funzione di rappresentazione**

$$fl : \mathbb{R} \rightarrow F \cup \{\pm\infty\}$$

Con una delle due regole, dato $x = \beta^p \sum_{i=1}^{\infty} \beta^{-i} d_i \in S$:

- Troncamento: $\tilde{x} = fl(x) = \beta^p \sum_{i=1}^t \beta^{-i} d_i$
- Arrotondamento: $\tilde{x} = fl(x)$ il troncamento di $x + \frac{\beta^{p-t-1}}{2}$

Se $x > \Omega$ impostiamo $fl(x) = \infty$ (**overflow**), se $0 \leq x \leq \omega$ impostiamo $fl(x) = 0$ (**underflow**), stesso ragionamento per i numeri negativi.

Definiamo la **precisione macchina** come $u = \beta^{-t+1}$ per il troncamento e $u = \frac{\beta^{-t+1}}{2}$ per l'arrotondamento.

Teorema

Dato $x \in S$ abbiamo il seguente limite per l'errore relativo:

$$\left| \frac{fl(x) - x}{x} \right| < u$$

F deve seguire determinate proprietà algebriche:

$$x, y \in F \not\Rightarrow x + y \in F$$

Dobbiamo definire delle **operazioni** con i numeri a virgola mobile.

Assumendo che esiste una somma a virgola mobile \oplus tale che se $x, y \in F$ (se non avviene overflow):

$$x \oplus y \in F, \quad x \oplus y = (x + y)(1 + \varepsilon), \quad |\varepsilon| < u$$

Similmente definiamo $\otimes, \ominus, \oslash$.

Un'idea è quella di definire $x \oplus y = fl(x + y)$ ma i dettagli sono più complicati.

Infatti le varie operazioni seguono solo alcune delle proprietà delle operazioni elementari:

- commutatività della somma
- commutatività del prodotto
- $x \oslash x = 1$

Non seguono infatti:

- associatività di somma e prodotto
- proprietà distributiva
- semplificazione
- potrebbe succedere che $x \otimes y = z \otimes y, \quad y \neq 0, \quad x \neq z$

3. Errore Per Le Funzioni Razionali

3. Errore Per Le Funzioni Razionali

Data una funzione razionale $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ che sia $f = \frac{p}{q}$ con p e q polinomi.

Dall'analisi matematica sappiamo che f è definita e differenziabile per $q \neq 0$ (assumendo che p e q siano primi).

Ci sono due tipi di errori nella valutazione di f con i valori in virgola mobile

Errore Inerente

Non valutiamo $f(x)$ ma valutiamo $f(\tilde{x})$ dove $\tilde{x} = fl(x)$ [funzione approssimata](#)

$$\varepsilon_{IN} = \frac{f(\tilde{x}) - f(x)}{f(x)}, \quad f(x) \neq 0$$

Se l'errore inerente è relativamente piccolo possiamo dire che il problema è **ben condizionato**, altrimenti che è **mal condizionato**.

L'errore inerente può essere definito anche per funzioni **non razionali**

Errore Algoritmico

Non si valuta $f(\tilde{x})$ ma si valuta $\tilde{f}(\tilde{x})$

$$\varepsilon_{ALG} = \frac{\tilde{f}(\tilde{x}) - f(\tilde{x})}{f(\tilde{x})}, \quad f(\tilde{x}) \neq 0$$

Se l'errore algoritmico è relativamente piccolo possiamo dire che l'algoritmo f è **numericamente stabile**, altrimenti che è **numericamente instabile**.

L'errore algoritmico può essere definito anche per le funzioni elementari che vengono trattate come operazioni.

Errore Totale

$$\varepsilon_{TOT} = \frac{\tilde{f}(\tilde{x}) - f(x)}{f(x)}, \quad f(x) \neq 0$$

Dà una misura genuina dell'errore nella valutazione.

La situazione ideale è $|\varepsilon_{tot}| < u$, ma in pratica è sufficiente $|\varepsilon_{tot}| < Mu$ con M costante.

Teorema

Dato $x \in \mathbb{R}^n \setminus \{0\}$ e $f : \mathbb{R}^n \rightarrow \mathbb{R}$ razionale con $f(x) \neq 0$, dove $\tilde{x} = fl(x)$, allora

$$\varepsilon_{TOT} = \varepsilon_{IN} + \varepsilon_{ALG} + \varepsilon_{IN}\varepsilon_{ALG}$$

Se ε_{IN} e ε_{ALG} tendono a 0 con $u \rightarrow 0$, abbiamo

$$\varepsilon_{TOT} = \varepsilon_{IN} + \varepsilon_{ALG} + o(u) = \varepsilon_{IN} + \varepsilon_{ALG}$$

≡ Example ▾

Sia $\tilde{x} = fl(x) = x(1 + \varepsilon_1)$, $|\varepsilon_1| < u$ ottenuto come

$$\frac{\tilde{x} - x}{x} = \varepsilon_1 \iff \tilde{x} - x = x\varepsilon_1 \iff \tilde{x} = x(1 + \varepsilon_1)$$

Abbiamo per $x \neq 0$:

$$\begin{aligned}\varepsilon_{IN} &= \frac{\tilde{x}^2 - x^2}{x^2} = \frac{[x(1 + \varepsilon_1)]^2 - x^2}{x^2} = \frac{x^2(1 + \varepsilon_1)^2 - x^2}{x^2} = \frac{x^2[(1 + \varepsilon_1)^2 - 1]}{x^2} = \\ &= 2\varepsilon + \varepsilon^2\end{aligned}$$

Dato che a noi interessa quello che succede con $u \rightarrow 0$, possiamo considerare solo i termini più lenti.

$$|\varepsilon_{IN}| = |2\varepsilon + \varepsilon^2| \leq 2|\varepsilon| + |\varepsilon^2| < 2u + u^2 = 2u, \quad u \rightarrow 0$$

è ben condizionato.

Esiste un'altra formula per calcolare l'errore inerente:

$$\varepsilon_{IN} = \frac{x}{f(x)} f'(\xi) \varepsilon_x$$

Dove $\varepsilon_x = \varepsilon_1$ è la rappresentazione dell'errore in x se $f \in C^2(\text{conv}(x, \tilde{x}))$ allora:

$$\varepsilon_{IN} = \frac{x}{f(x)} f'(x) \varepsilon_x + o(u) = \frac{x \cdot 2x}{x^2} \varepsilon_x = 2\varepsilon_x$$

✔ Dimostrazione

Uso il teorema di Lagrange applicato alla funzione in un intorno di 0

$$f(x) = f(0) + f'(0)x + o(x) \doteq f(0) + f'(0)x$$

Important

Il termine $\varepsilon_{IN} = \frac{f(\tilde{x}) - f(x)}{f(x)} \doteq \frac{x}{f(x)} f'(x) \varepsilon_x$ si chiama **fattore di amplificazione** e misura l'amplificazione dell'errore

4. Problemi Ben Posti

Un problema è composto di tre parti:

- Dati
- Incognite
- Condizioni

Example

Sistemi lineari

Data una matrice A (coefficienti) e un vettore b (lato destro), trovare tutti i vettori x (incognite) tali che $Ax = b$

Le incognite sono funzioni dei dati e nella pratica, possono essere presentate solo delle approssimazioni.

Problemi ben posti

- Hanno una soluzione
- La soluzione è unica
- La soluzione dipende continuamente dai dati

La dipendenza continua dai dati è meno ovviamente importante:

- I dati nei problemi reali sono affetti da errori

- I calcoli vengono eseguiti da un'aritmetica finita e ci sono errori di arrotondamento

Dipendenza Continua Dai Dati

5. Derivata di più variabili

Derivata Di Fréchet

$$f : \mathbb{R}^n \rightarrow \mathbb{R}$$

$$Df(x_0) : \mathbb{R}^n \rightarrow \mathbb{R}$$

Se $x = (x_1, \dots, x_n)$, $x_i \neq 0$, $f(x) \neq 0$ l'[errore inerente](#) è:

$$\varepsilon_{IN} \doteq c_1 \varepsilon_1 + \dots + c_n \varepsilon_n, \quad c_i = \frac{x_i}{f(x)} \frac{\partial f}{\partial x_i}(x), \quad \varepsilon_i = \frac{\tilde{x}_i - x_i}{x_i}$$

dove ε_i è l'[errore di rappresentazione](#) e c_i il [coefficiente di amplificazione](#)

$$\varepsilon_{IN} \doteq \frac{x_1}{f(x)} \frac{\partial f}{\partial x_1}(x) \varepsilon_1 + \dots + \frac{x_n}{f(x)} \frac{\partial f}{\partial x_n}(x) \varepsilon_n$$

Example ▾

Data $f(x, y) = xy$, la moltiplicazione. Allora $\varepsilon_{IN} \doteq c_x \varepsilon_x + c_y \varepsilon_y$, dove

$$c_x = \frac{x}{f(x, y)} \frac{\partial f}{\partial x}(x, y) = \frac{x}{xy} y = 1, \quad c_y = \frac{y}{f(x, y)} \frac{\partial f}{\partial y}(x, y) = \frac{y}{xy} x = 1$$

Abbiamo quindi

$$c_x = 1, \quad c_y = 1$$

E il problema è [ben posto](#).

Example ▾

Data $f(x, y) = x + y$ la somma. Allora $\varepsilon_{IN} \doteq c_x \varepsilon_x + c_y \varepsilon_y$, dove

$$c_x = \frac{x}{f(x,y)} \frac{\partial f}{\partial x}(x,y) = \frac{x}{x+y}, \quad c_y = \frac{y}{f(x,y)} \frac{\partial f}{\partial y}(x,y) = \frac{y}{x+y}$$

Abbiamo quindi

$$c_x = \frac{x}{x+y}, \quad c_y = \frac{y}{x+y}$$

E il problema diventa mal posto quando $x \approx -y$

Questo fenomeno si chiama **cancellazione numerica**

$$|\varepsilon_{IN}| = \left| \frac{x\varepsilon_x + y\varepsilon_y}{x+y} \right| \leq \frac{|x\varepsilon_x| + |y\varepsilon_y|}{|x+y|}$$

6. Valutazione di un polinomio

Dati p e x , calcolare $p(x)$

- $\mathbb{R}^n(K^n)$ vettori con n componenti in $\mathbb{R}(K)$
- $\mathbb{R}_n(x)(K_n(x))$ polinomi di grado al più n con coefficienti reali
- $\mathbb{R}^{n \times m}(K^{n \times m})$ matrici $n \times m$ con elementi in \mathbb{R}
- $\mathbb{R}^{n_1 n_2 \dots n_l}$ tensore di l dimensioni
- $p(x) = a_0 + a_1 x + \dots + a_n x^n$

In computer grafica quasi tutto dipende da delle curve, le superfici possono essere curve, ecc...

Il calcolatore per disegnare il grafico di una funzione conta numerosi puntini all'interno del grafico della funzione e ne valuta il valore.

$$\sum_{i=0}^n a_i x^i$$

```
s = 0
for i = 0:n
    s = s + a(i) * x^i
end
```

ho $\frac{(n+1)(n+2)}{2}$ operazioni

```
s = 0
p = 1
for i = 0:n
    s = s + a(i) * p
    p = p * x
end
```

il numero totale è $3(n+1)$ operazioni

Algoritmo Di Ruffini-Horner

$$a_0 + a_1x + a_2x^2 + a_3x^3$$

$$((a_3x + a_2)x + a_1)x + a_0$$

$$((\dots((a_nx + a_{n-1})x + a_{n-2})x + \dots)x + a_1)x + a_0$$

$$s_n = a_n s_i = s_{i+1}x + a_i$$

$$\begin{cases} s_n = a_n \\ s_i = s_{i+1}x + a_i \end{cases} \quad \text{metodo di Ruffini-Horner}$$

```
s = a(n)
for i = n-1:0
    s = s * x + a(i)
end
```

$2n$ operazioni

✔ Dimostrazione per induzione

Passo base:

Per $n = 0$ il polinomio è $p(x) = a_0$ per ogni x , il passo base quindi è verificato.

Passo induttivo:

La proposizione è vera per polinomi di grado $n - 1 \implies$ vera per polinomi di grado n

$$p(x) = a_0 + x(a_1 + a_2x + \dots + a_{n-1}x^{n-2} + a_nx^{n-1}) = a_0 + xq(x)$$

Dove il polinomio q ha grado $n - 1$ e può essere scomposto con il teorema di Ruffini-Horner

Funzioni Razionali con Ruffini-Horner

Il teorema di Ruffini-Horner può essere usato anche per una funzione razionale del tipo:

$$r(x) = \frac{p(x)}{q(x)}$$

Dove p ha grado m e q ha grado n , è sufficiente applicare l'algoritmo a $p(x)$ e $q(x)$ separatamente e poi dividere i risultati, richiedendo $2(m + n) + 1$ operazioni aritmetiche.

7. Vettori

$$w, v \in K^n$$

$$v = \langle v_1, \dots, v_n \rangle$$

$$w = \langle w_1, \dots, w_n \rangle$$

$$v + w = \langle v_1 + w_1, \dots, v_n + w_n \rangle$$

Prodotto Riga per Colonna Tra Due Vettori

$$v^t \cdot w = [v_1, \dots, v_n] \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} = v_1w_1 + v_2w_2 + \dots + v_nw_n$$

Prodotto Matrice per Vettore

$$A \in K^{m \times n}$$

$$b \in K^n$$

$$Ab = c \in K^m$$

$$A = \langle r_i^t, \dots, r_m^t \rangle$$

Faccio il prodotto riga per colonna tra le righe della matrice e il vettore.

Ci sono anche altre interpretazioni, come quello impiegato nel deep learning

Prodotto Tra Matrici Di Strassen

Algoritmo per semplificare il prodotto tra due matrici 2×2 .

La moltiplicazione riga per colonna impiega 8 moltiplicazioni e 4 addizioni.

L'algoritmo di Strassen impiega 7 moltiplicazioni e 18 addizioni.

Può sembrare che impieghi un tempo peggiore rispetto alla riga per colonna, ma la moltiplicazione tra due vettori impiega molto di più che l'addizione, rendendo quindi l'algoritmo più veloce.

L'algoritmo viene poi trasformato in moltiplicazioni in blocchi. Rendendo qualsiasi moltiplicazione tra matrici $2^n \times 2^n$ come 7 moltiplicazioni e alcune addizioni tra 4 matrici $2^{(n-1)} \times 2^{(n-1)}$ ricorsivamente.

Questo ha fatto scaturire una corsa all'algoritmo più veloce per risoluzione di prodotti tra matrici.

Ma questi algoritmi non vengono usati.

Somme Di Elementi in Parallelo

Ho un vettore di 2^n elementi che devono essere sommati tra loro.

Potrei scorrere tutto il vettore e sommare il primo elemento con il secondo poi con il terzo e così via, però sarebbe troppo lento. Posso quindi sommare in parallelo gli elementi a due a due, creando così un secondo vettore di 2^{n-1} elementi e ripeto l'operazione finché non arrivo a un vettore di 1 elemento. Il risultato è lo stesso ma eseguendo le operazioni parallelamente con vettori molto grandi l'algoritmo è molto più veloce.

8. Matrici

Il [metodo di Ruffini-Horner](#) può essere interpretato come moltiplicazioni di matrici, che quindi può essere eseguito in parallelo impiegando $2 \log_2(n)$ cicli macchina.

$$\begin{bmatrix} x & a_0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x & a_1 \\ 0 & 1 \end{bmatrix} \cdots \begin{bmatrix} x & a_n \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Queste matrici hanno una struttura particolare per cui si possono effettuare meno operazioni, aggiungendo che queste operazioni possono essere eseguite in parallelo non si ottiene un considerevole vantaggio.

Matrice Sparsa

Una matrice che ha molti 0:

$$A \in K^{n \times n}$$

$$\text{numZeri}(A) \ll n^2$$

si riesce a implementare il [prodotto matrice-vettore](#) con $2l$ operazioni anziché con $2n^2$

9. Altre Strutture

- Sottospazi vettoriali
- Matrici diagonali (prodotto matrice-vettore, prodotto righe per colonne)

Matrici Diagonali

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$D_n = \{A \in K^{n \times n}, a_{ij} = 0, i \neq j\}$$

❓ Le matrici diagonali sono sparse?

$$\text{numZeri}(A) \leq n$$

$$\lim_{n \rightarrow \infty} \frac{\text{numZeri}(A)}{n^2} = 0$$

✓ Check

Sono strutture sparse, sono sufficienti $2p$.

? è un sottospazio vettoriale?

$$A, B \in D_n, \quad \alpha \in K \implies A + B \in D_n, \quad \alpha A \in D_n$$

✓ Somma tra matrici

Per dimostrare la somma basta dimostrare che $c_{ij} = 0, i \neq j$:

$$c_{ij} = a_{ij} + b_{ij} = 0 + 0 = 0, i \neq j$$

✓ Moltiplicazione per uno scalare

Stessa cosa della somma:

$$c_{ij} = \alpha \cdot a_{ij} = \alpha \cdot 0 = 0, i \neq j$$

? è una sottoalgebra di $K^{n \times n}$?

Un'algebra è un sottospazio vettoriale in cui è definito un prodotto compatibile con le operazioni dello spazio vettoriale.

V è un gruppo se dati $A, B, C \in V$ e $\alpha \in K$:

- $A(B + C) = AB + AC$
- $(A + B)C = AC + BC$
- $\alpha(AB) = (\alpha A)B = A(\alpha B)$

Le matrici $n \times n$ sono un'algebra con il prodotto riga per colonna

Algoritmi per Il Prodotto Tra Matrici Diagonali

Algoritmo Non Strutturato

```
for i = 1:n
    for j = 1:n
        c(i,j) = 0
        for l = 1:n
            c(i,j) = c(i,j) + a(i,l) * b(l,j)
```

L'algoritmo strutturato si può ottenere spesso partendo dal codice di quello non strutturato

Algoritmo Strutturato

```
c = 0
for i = 1:n
    c(i,i) = a(i,i) * b(i,i)
```

Strutture E Algoritmi Strutturati

Strutture: sottospazi vettoriali, sottoalgebre

Algoritmo strutturato: algoritmo che sfrutta la struttura

Example ▾

- Matrici diagonali: sottoalgebra
- Matrici triangolari: sottoalgebra
- Matrici tridiagonali: sottospazio vettoriale

a0. Sistemi lineari

$$Ax = b$$

$$A \in K^{n \times n}, x \in K^n, b \in K^n$$

$$\begin{cases} a_{1,1}x_1 + \dots + a_{1,n}x_n = b_1 \\ a_{2,1}x_1 + \dots + a_{2,n}x_n = b_2 \\ \vdots \\ a_{n,1}x_1 + \dots + a_{n,n}x_n = b_n \end{cases}$$

Teorema Di Rouché-Capelli

Il sistema lineare $Ax = b$ ammette soluzione se e solo se $\text{rango}(A) = \text{rango}(A|b)$.

Se esiste almeno una soluzione allora l'insieme delle soluzioni è un sottospazio affine di dimensione $n - \text{rango}(A)$.

K^n spazio vettoriale

v sottospazio vettoriale di dimensione l con $0 \leq l \leq n$

$v_0 \in K^n$ la traslazione di v , t_v è l'applicazione

$$t_{v_0} : K^n \rightarrow K^n$$

$$v \rightarrow v + v_0$$

L'insieme $W = \{v + v_0, v \in V, v_0 \in K^n\}$ è un sottospazio affine di K^n di dimensione $\dim(V)$.

Info

- I sottospazi affini di dimensione 1 sono detti rette
- I sottospazi affini di dimensione 2 sono detti piani
- I sottospazi affini di dimensione $n - 1$ sono detti iperpiani
- I sottospazi affini di dimensione 0 sono detti punti (non sono sottospazi)

$Ax = b \iff c_1x_1 + c_2x_2 + \dots + c_nx_n = b \iff b$ è combinazione lineare delle colonne di $A \iff \text{rango}(A) = \text{rango}(A|b)$

$$|c_1| \dots |c_n| = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

Problema Ben Posto

- una soluzione esiste
- la soluzione è unica
- la soluzione dipende in modo continuo dai dati

la soluzione è unica $\iff \text{rango}(A) = \text{rango}(A|b) = n \iff n - \text{rango}(A) = 0$

Teorema

Il sistema lineare $Ax = b$ è ben posto se e solo se A è quadrata e invertibile

Caso 1

$m < n \quad \text{rango}(A) \leq m < n \implies$ la soluzione non può essere unica \implies non è ben posto

Caso 2

$m > n \quad \text{rango}(A) = \text{rango}(A|b) = n \iff$ la soluzione non è unica, anche se il sistema ammette una soluzione unica per un dato b , esiste $c : \forall \varepsilon > 0$ il sistema $Ax = c + \varepsilon c$ non ammette soluzioni \implies la soluzione non può dipendere in modo continuo dai dati

$$\text{Im}(A) = \{v \in K^n, v = Ax, x \in \mathbb{R}^n\} \subseteq K^n$$

$$\dim(\text{Im}(A)) = \text{rango}(A) \leq n < m$$

esiste $c \in K/\text{Im}(A)$ ma $x : Ax = b, x \in \text{Im}(A)$

$$Ay = b + c\varepsilon, \varepsilon > 0$$

per ogni $\varepsilon > 0$ questo sistema non ha soluzione se esiste $y : Ay = b + c\varepsilon$

$$Ay = Ax + c\varepsilon \quad Ay - Ax = c\varepsilon \quad A\left(\frac{y - x}{\varepsilon}\right) = c$$

è una contraddizione.

Caso 3

$m = n \quad \det(A) = 0 \implies$ la soluzione non esiste o non è unica \implies non è ben posto

Caso 4

A quadrata, $m = n \quad \det(A) \neq 0 \implies$ la soluzione è unica.

Dipende in modo continuo dai dati?

$$x = A^{-1}b \quad (A^{-1})_{ij} = \frac{\det(\bar{A}^{ij})}{\det(A)} (-1)^{i+j}$$
$$x_i = \sum_{l=1}^n (A^{-1})_{il} b_l = \sum_{l=1}^n \frac{\det(\bar{A}^{il})}{\det(A)} (-1)^{i+l} b_l, \quad i = 1, \dots, n$$

è una funzione continua di a_{il}, b_l poiché è razionale

Algoritmi Di Risoluzione

Caso Facile (matrici triangolari)

$$T_n = \{A \in K^{n \times n}, a_{ij} = 0, i > j\}$$

$$\begin{bmatrix} a_{1,1} & \dots & a_{1,n} \\ \vdots & \ddots & \vdots \\ 0 & \dots & a_{n,n} \end{bmatrix}$$

Sistema lineare $Tx = b$ dove T è triangolare superiore

$$\begin{cases} t_{1,1}x_1 + t_{1,2}x_2 + t_{1,3}x_3 = b_1 \\ \quad t_{2,2}x_2 + t_{2,3}x_3 = b_2 \\ \quad \quad t_{3,3}x_3 = b_3 \end{cases}$$

Calcolo il termine i -esimo

$$t_{i,i}x_i + t_{i,i+1}x_{i+1} + \dots + t_{i,n}x_n = b_i$$
$$x_i = \frac{b_i - t_{i,i+1}x_{i+1} - \dots - t_{i,n}x_n}{t_{i,i}} = \frac{b_i - \sum_{j=i+1}^n t_{i,j}x_j}{t_{i,i}}$$

$$Tx = b \implies t_{i,1}x_1 + t_{i,2}x_2 + \dots + t_{i,i}x_i + \dots = b_i$$

$$\sum_{j=i}^n t_{i,j}x_j = b_i$$

Algoritmo Di Sostituzione All'indietro

```

for i=n:-1:1
    s=b(i)
    for j= i+1:n
        s = s - t(i,j) * x(j)
    x(i) = s / t(i,i)

```

costo di un passo del ciclo esterno: $2(n - i) + 1$ ops

$$\sum_{i=1}^n 2(n - i) + 1 = 2 \sum_{i=1}^n 2n - 2i + 1 = 2 \sum_{i=1}^n n - 2 \sum_{i=1}^n i + \sum_{i=1}^n 1 = 2n^2 - n(n + 1) + n = n^2 \text{ op}$$

Note

L'algoritmo fallisce se $t_{i,i} = 0$

Si può dimostrare che $\det(T) = t_{1,1}t_{2,2} \dots t_{n,n}$

$\det \neq 0 \iff t_{i,i} \neq 0 \iff$ l'algoritmo è applicabile

Algoritmo Di Sostituzione in Avanti

Per sistemi triangolari inferiori

$$\begin{cases} t_{1,1}x_1 & & & & = b_1 \\ t_{2,1}x_1 + t_{2,2}x_2 & & & & = b_2 \\ t_{3,1}x_1 + t_{3,2}x_2 + t_{3,3}x_3 & & & & = b_3 \\ t_{4,1}x_1 + t_{4,2}x_2 + t_{4,3}x_3 + t_{4,4}x_4 & & & & = b_4 \end{cases}$$

$$\begin{bmatrix} * & 0 & 0 & 0 \\ * & * & 0 & 0 \\ * & * & * & 0 \\ * & * & * & * \end{bmatrix}$$

```

for i = 1:n
    s = b(i)
    for j = 1:i-1
        s = s - t(i,j) * x(j)
    x(i) = s / t(i,i)

```

a1. Algoritmo di Gauss

$Ax = b$, con A quadrata e invertibile

```
for h=1:n-1:
    for i = h+1:n:
        l(i,h)=a(i,h) / a(h,h)
        a(i,h) = 0
        for j=h+1:n:
            a(i,j) -= l(i,h) * a(h,j)
        b(i) -= l(i,h) * b(h)
```

L'algoritmo per come è implementato richiede

$$\sum_{h=1}^{n-1} 2(n-h)^2 = \frac{2}{3}n^3$$

$$\begin{bmatrix} a_{11}, & a_{12}, & a_{13}, & \dots \\ a_{21}, & a_{22}, & a_{23}, & \dots \\ \vdots, & \vdots, & \vdots, & \ddots \end{bmatrix}$$

Il primo passo l'algoritmo ha come pivot a_{11}

Non ce se la fa a staglie dietro mannaggia

Dato $A \in K^{n \times n}$ il minore principale di testa di ordine $j \leq h \leq n$ di A è la matrice $M_h \in K^{n \times n} : (M_h)_{ij} = a_{ij}$

Teorema

Sia $A \in K^{n \times n}$ e siano M_1, \dots, M_n i suoi minori principali di testa. Il metodo di Gauss è applicabile se e solo se $\det(M_h) \neq 0, h = 1, \dots, n-1$

Dimostrazione

Supponiamo il metodo di Gauss sia applicabile \implies si può ottenere la matrice

$$U = A_n = \begin{bmatrix} a_{11} & * & & \\ & a_{22} & & \\ & & a_{33} & \\ & 0 & & a_{44} \end{bmatrix}$$

Sia N_h il minore principale di testa di U di ordine h , esso è una matrice triangolare e il suo determinante è dato dagli elementi sulla sua diagonale, quindi $\det(N_h) = u_{11}u_{22}\dots u_{hh}$

Siccome il metodo è applicabile allora

$$a_{ii} \neq 0, i = 1, \dots, n-1 \implies a_{11}\dots a_{hh} \neq 0 \implies \det(N_h) \neq 0, h = 1, \dots, n-1$$

✓ Done

Si ha che $\det(N_h) = \det(M_h) \implies \det(M_h) \neq 0, h = 1, \dots, n-1$ poiché le operazioni (le combinazioni di righe) del metodo di Gauss non cambiano il determinante della matrice a cui si applica, ma non cambia neanche quello dei suoi minori principali di testa

✗ Failure

Se il metodo di Gauss fallisce al passo h allora

$a_{hh} = 0$ ($a_{jj} \neq 0, j = 1, \dots, h-1$) e si può costruire la matrice

$$A_h = \begin{bmatrix} a_{11} & & & \\ & \ddots & & \\ & & a_{hh} = 0 & \\ & & & \end{bmatrix}$$

Sia N_h il minore principale di testa di ordine h di A_h , $\det(N_h) = 0$

Ma $\det(M_h) = \det(N_h)$ per quanto detto prima, quindi il metodo non è applicabile

Se A è una matrice definita positiva allora la condizione del teorema è verificata

$A \in \mathbb{R}^{n \times n}$ è definita positiva se:

- $A^T = A$ (simmetrica)
- $v^T A v > 0$ se $v \in \mathbb{R}^n \setminus \{0\}$

Se A è a dominanza diagonale allora la condizione del teorema è verificata

A è a dominanza diagonale (stretta) se

- $|a_{ii}| > \sum_{j=1, j \neq i}^n |a_{ij}|$

A è a dominanza diagonale se:

- $|a_{ii}| \geq \sum_{j=1, j \neq i}^n |a_{ij}|$
- vale il maggiore stretto per almeno un indice

Sistema lineare

Come si risolve un sistema lineare?

$$A_1 = A \rightarrow A_2 \rightarrow A_3 \rightarrow \dots \rightarrow A_n = U$$

$$[A|b] = [A_1|b_1] \rightarrow [A_2|b_2]$$

```
for h=1:n-1:
    for i = h+1:n:
        l(i,h)=a(i,h) / a(h,h)
        for j=h+1:n:
            a(i,j) = a(i,j) - l(i,h) * a(h,j)
        a(i,h) = 0
    b(i) = b(i) - l(i,h) * b(h)
```

Si può interpretare il metodo di Gauss come operazioni sulle equazioni

$$Ax = b \iff A_1x = b_1 \iff \dots \iff A_nx = b_n \iff Ux = \tilde{b}$$

Applicando il metodo di Gauss si parte da $Ax = b$ e si ottiene un sistema triangolare equivalente $Ux = \tilde{b}$ che si può risolvere con il metodo di [sostituzione all'indietro](#)

Qual è il costo computazionale?

$\frac{2}{3}n^3$ per l'eliminazione

n^2 per la soluzione di $Ux = \tilde{b}$

$\sum_{h=1}^n 2(n-h) \sim n^2$ per il calcolo di \tilde{b}

Note

$\det(A_1) = \det(A_2) = \dots = \det(U)$ perché le operazioni che eseguiamo non modificano il determinante.

è sufficiente quindi applicare il metodo di Gauss con $\frac{2}{3}n^3$ operazioni (e $n-1$ moltiplicazioni che si possono trascurare)

Sistemi Lineari Con Membro Destro Multiplo

$$Ax = b_1, Ax = b_2, \dots, Ax = b_l$$

risolvendo indipendentemente i sistemi lineari sono richieste $\frac{2}{3}n^3l$ operazioni.

Applico il metodo di Gauss ad A modificando consistentemente anche le colonne relative ai termini noti

$$[A|b_1b_2\dots b_l] = [A_1|b_1^{(1)}b_2^{(1)}\dots b_l^{(1)}] \rightarrow [A_2|b_1^{(2)}b_2^{(2)}\dots b_l^{(2)}] \rightarrow \dots \rightarrow [A_n|b_1^{(n)}b_2^{(n)}\dots b_l^{(n)}] = [U|\tilde{b}_1\tilde{b}_2\dots\tilde{b}_l]$$

$$Ux = \tilde{b}_1, Ux = \tilde{b}_2, \dots, Ux = \tilde{b}_l$$

Risolvendo in "parallelo" $\frac{2}{3}n^3l$ operazioni.

Quante operazioni in più al primo passo? $2l(n-1)$

al secondo passo? $2l(n-2)$

al passo i ? $2l(n-i)$

in totale saranno $\sum_{i=1}^{n-1} 2l(n-i)$

$$\sum_{i=1}^{n-1} 2l(n-i) = 2l \sum_{i=1}^{n-1} (n-i) \sim \frac{2ln^2}{2} = ln^2$$

$\frac{2}{3}n^3 + ln^2$ operazioni anziché $\frac{2}{3}n^3l$ per trovare i sistemi $Ux = \tilde{b}_1, \dots, Ux = \tilde{b}_l$ per risolverli sono necessarie l sostituzioni all'indietro e quindi altre ln^2 operazioni.

In totale sono $\frac{2}{3}n^3 + 2n^2l$ operazioni per risolvere i sistemi lineari con sistemi coefficienti

Calcolare l'inversa di una matrice

X è l'inversa di A ($n \times n$)

se $AX = I$

$$A[X_1|X_2|\dots|X_n] = [F_1|F_2|\dots|F_n]$$

$$[AX_1|AX_2|\dots|AX_n] = [F_1|F_2|\dots|F_n]$$

$$AX_1 = F_1, AX_2 = F_2, \dots, AX_n = F_n$$

Nota

Il problema di calcolo dell'inversa si riconduce ad un problema di risolvere un sistema lineare con membro destro multiplo

Utilizzando l'algoritmo precedente ci vogliono


$$\frac{2}{3}n^3 + 2n^2n = \frac{8}{3}n^3$$

In realtà si può calcolare l'inversa con $2n^3$ operazioni

a2. Matrici tridiagonali

$$A = \begin{bmatrix} * & * & & & & \\ * & * & * & & & \\ & * & * & * & & \\ & & * & * & * & \\ & & & * & * & * \\ & & & & * & * & * \\ & & & & & * & * \end{bmatrix}$$

$T_n = \{A \in K^{n \times n}, a_{ij} = 0, |i - j| > 1\}$ è un sottospazio vettoriale poiché è una def. dalla posizione degli elementi nulli se sommo o multiplico per uno scalare elementi nulli rimangono nulli.

 sono una sottoalgebra?

è vero che $A, B \in T_n \implies AB \in T_n$?

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 2 \end{bmatrix}$$

Il risultato non è una matrice tridiagonale, quindi non è una sottoalgebra

Trovare un algoritmo che calcola il prodotto di una matrice diagonale $A \in T_n$ per un vettore $b \in K^n$ e che non richieda più di $6n$ operazioni e darne un'implementazione in pseudocodice (non usare il prodotto matrice sparsa-vettore)

Primo approccio: scrivo la matrice e il vettore e vedo cosa succede

Secondo approccio: scrivo il componente c_i

algoritmo mio

```
prodotto(A,b):
    c[1] = A[1,1] * b[1]
    for i=:2:n:
        c[i-1] += A[i-1,i] * b[i]
        c[i] = A[i,i-1] * b[i-1] + A[i, i] * b[i]
```

a3. Fattorizzazione LU

Algoritmo di Gauss \iff fatt. LU

$$A_1 \rightarrow A_2 \rightarrow \dots \rightarrow U = A_n = \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{bmatrix}$$

costruiamo una matrice L triangolare inferiore con 1 sulle diagonali tale che L_{ij} è definito dal metodo di Gauss $i > j$

$$L = \begin{bmatrix} 1 & 0 & 0 \\ l_{ij} & 1 & 0 \\ l_{ij} & l_{ij} & 1 \end{bmatrix}$$

$$l_{ih} = \frac{a_{ih}}{a_{hh}} \quad l_{ij} = \frac{a_{ij}}{a_{jj}}$$

$$a_{ij}^{(h+1)} = \begin{cases} a_{ij}^{(h)} - l_{ih}a_{hj}^{(h)}, & i > h, j \geq h \\ a_{ij}^{(h)} & \text{alternative} \end{cases}$$

Teorema

$A = LU$ (con le valutazioni viste sopra)

Implementazione Del Metodo Di Gauss

a1. Algoritmo di Gauss

Si parte dalla matrice completa $[A_1|b_1]$ e si eliminano tutti gli elementi della prima colonna a parte il primo, per fare questo si usa la somma di una riga per il multiplo della prima. Il valore per cui si moltiplica è dato dal rapporto dell'elemento nella posizione da eliminare per l'elemento sulla diagonale.

Example ▾

Al passo 1, se voglio eliminare l'elemento nella riga 2 (a_{21}) devo eseguire l'operazione $r2 = r2 - \frac{a_{21}}{a_{11}}r1$

Bisogna ripetere questi passaggi per ogni colonna usando come pivot la diagonale della matrice

Il rapporto è poi utilizzato per formare la matrice L , infatti ogni elemento di L :

$$l_{ij} = \frac{a_{ij}}{a_{jj}}$$

Il risultato saranno poi due matrici triangolari:

- U triangolare superiore
- L triangolare inferiore

Come Si Risolve Un Sistema Lineare?

$A = LU$ calcoliamo la fattorizzazione LU di A con il metodo di Gauss

$$Ax = b \iff L U x = b \iff L(Ux) = b \implies \begin{cases} Ly = b \\ Ux = y \end{cases}$$

Risolviamo il primo sistema $Ly = b$ con l'[algoritmo di sostituzione in avanti](#) e poi $Ux = y$ con l'[algoritmo di sostituzione all'indietro](#) ottenendo x .

Quante Operazioni Richiede?

Il costo totale delle operazioni sulla matrice A è di

$$\sum_{k=1}^{n-1} [2(n-k)^2 + (n-k)]$$

Di cui:

- $\frac{2}{3}n^3$ per la fattorizzazione
- $n^2 + n^2$ per le due sostituzioni di \tilde{b} e la soluzione di $Ux = \tilde{b}$ (trascurabili)

il costo computazionale è lo stesso ($y = \tilde{b}$)

$$\begin{bmatrix} \varepsilon & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 + \varepsilon \\ 1 \end{bmatrix}$$

il pivot per $\varepsilon > 0$ non è nullo e la soluzione è unica $x = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

a3a. Esercizio

Descrivere una variante strutturata dell'algoritmo di Gauss che richieda non più di $10n$ operazioni per la soluzione del sistema lineare $Ax = b$ con A matrice tridiagonale.

$$[A|b] \rightarrow \dots \rightarrow [U|\tilde{b}]$$

$$Ux = \tilde{b}$$

$\frac{2}{3}n^3$ operazioni algoritmo di Gauss applicato ad A

n^2 operazioni per aggiornare b

n^3 operazioni per risolvere $Ux = \tilde{b}$

A è tridiagonale se $a_{ij} = 0, |i - j| > 1$

Al primo passo del metodo di Gauss è necessario eliminare gli elementi

$$a_{21}, \dots, a_{n1}$$

Alcuni di essi sono già nulli e quindi alcune operazioni si possono evitare

$$r_2^T \leftarrow r_2^T - l_{21}r_1^T$$

Nel caso tridiagonale la combinazione di righe modifica solo un elemento e quindi sono richieste solo 2 operazioni (e una divisione per calcolare l_{21}) il primo passo richiede 3 operazioni.

Al secondo passo si osserva che la sottomatrice B_2 è tridiagonale, è sufficiente quindi la combinazione lineare

$$r_3^T \leftarrow r_3^T - l_{32}r_2^T$$

che richiede 3 operazioni.

L'algoritmo richiede $3(n - 1)$ passi in totale

Implementazione:

Algoritmo originale di gauss

```
for h=1:n-1
    for i=h+1:n
        l(i,h) = a(i,h) / a(h,h)
        for j=h+1:n
            a(i,j)=a(i,j)-l(i,h)*a(h,j)
```

Algoritmo modificato


```

for h=1:n-1
    l(h+1, h) = a(h+1, h) / a(h,h)
    a(h+1,h+1) = a(h+1,h+1) - l(h+1,h)*a(h,h+1)
    a(h+1,h=0)=0
    b(h+1) = b(h+1) - l(h+1, h)*b(h)

```

Per calcolare U sono sufficienti $3(n-1)$ operazioni, per calcolare \tilde{b} sono sufficienti $2(n-1)$ operazioni

Per risolvere $Ux = \tilde{b}$ sono sufficienti $3(n-1) + 1$ operazioni

$$\begin{bmatrix} a_{11} & a_{12} & 0 & \dots & \dots & 0 \\ & a_{22} & a_{23} & 0 & \dots & 0 \\ & & \ddots & \ddots & \ddots & \vdots \\ & & & & & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \tilde{b}_1 \\ \tilde{b}_2 \\ \vdots \\ \tilde{b}_n \end{bmatrix}$$

Si può risolvere con $O(n)$ operazioni?

Modo 1

$$i -esima \rightarrow \begin{cases} U_{11}x_1 + U_{12}x_2 = \tilde{b}_1 \\ U_{22}x_2 + U_{23}x_3 = \tilde{b}_2 \\ \vdots \\ U_{ii}x_i + U_{i+1,i+1}x_{i+1} = \tilde{b}_i \\ \vdots \\ U_{nn}x_n = \tilde{b}_n \end{cases}$$

```

x(n)=bt(n)/u(n,n)
for i=n-1:-1:1
    x(i) = (bt(i)- u(i,i+1) *x(i+1)) / u(i,i)

```

$3(n-1) + 1$ operazioni $\sim 3n$ operazioni

[a3. Fattorizzazione LU](#)

a4. Variante del Massimo Pivot Parziale (GEPP)

Riassunto a1. Algoritmo di Gauss

- Implementazione
- Costo computazionale $\frac{2}{3}n^3$
- Risolve molti problemi
 - sistemi lineari
 - inversa
 - determinante
- Equivalente alla fattorizzazione LU
- non è sempre applicabile ma ci sono dei criteri di applicabilità (veri per alcune matrici)

$A = LU$ in aritmetica esatta

$\tilde{A} = \tilde{L}\tilde{U}$ in aritmetica finita, ed esiste $|\tilde{A}_{ij} - A_{ij}| \leq \gamma(n)(|\tilde{A}_{ij}| + |\tilde{L}_{ij}||\tilde{U}_{ij}|)$

Quindi il metodo di Gauss è un **algoritmo instabile**, non è adeguato.

Implementazione

Ad ogni passo prima dell'eliminazione si scambiano due righe per ottenere il pivot di massimo modulo.

Al primo passo si cerca sulla riga/colonna il primo elemento di massimo modulo e se questo elemento non è a_{11} allora si scambia la prima riga con quella dell'elemento di massimo modulo.

$$A_1 = \begin{bmatrix} a_{11} & \dots \\ a_{22} & \dots \\ a_{33} & \dots \\ \vdots & \vdots \\ a_{nn} & \dots \end{bmatrix} \rightarrow |a_{p1}| = \max\{|a_{11}|, \dots, |a_{nn}|\} \rightarrow B_1 = \begin{bmatrix} a_{p1} & \dots \\ a_{22} & \dots \\ a_{33} & \dots \\ \vdots & \vdots \\ a_{nn} & \dots \end{bmatrix} \rightarrow \text{eliminazione} \rightarrow$$
$$\rightarrow A_2 = \begin{bmatrix} a_{11} & a_{12} & \dots \\ 0 & a_{22} & \dots \\ 0 & a_{32} & \dots \\ \vdots & \vdots & \vdots \\ 0 & a_{n2} & \dots \end{bmatrix} \rightarrow \text{scambio} \rightarrow B_2 = \begin{bmatrix} a_{11} & a_{12} & \dots \\ 0 & a_{p2} & \dots \\ 0 & a_{32} & \dots \\ \vdots & \vdots & \vdots \\ 0 & a_{n2} & \dots \end{bmatrix} \rightarrow A_3 = \dots$$

$\operatorname{argmax}\{|a_{11}|, \dots, |a_{nn}|\}$ è l'insieme degli indici in cui il massimo viene raggiunto

```
for h=1: n-1
    p = 1; M = abs(a(h,h));
    for i = h+1: n
        m = abs(a(i,h));
        if(m > M)
            M = m;
            p = i;
    if p != h
        for j=h:n
            swap = a(h, j)
            a(h, j) = a(p, j)
            a(p, j) = swap
        swap = b(h)
        b(h) = b(p)
        b(p) = swap
    for i = k+1 : n
//metodo di gauss
```

Costo Computazionale

I confronti al primo h sono

$$n - (h + 1) + 1 = n - h$$

E i valori assoluti da calcolare

$$n - h + 1$$

$2(n - h) + 1$ confronti al passo h

Il numero totale è

$$\sum_{h=1}^{n-1} 2(n - h) + 1 \sim n^2$$

n^2 è trascurabile rispetto al costo necessario per l'eliminazione ($\frac{2}{3}n^3$)

Applicabilità

$$\begin{bmatrix} 0 & \dots \\ 0 & \vdots \\ \vdots & \vdots \\ 0 & \dots \end{bmatrix}$$

Al primo passo il metodo si arresta se la prima colonna è nulla $\implies A$ non è invertibile $\iff \det(A) = 0$

Teorema

Sia $A \in K^{n \times n}$ invertibile (condizione sufficiente). Il metodo di Gauss con la variante del massimo pivot parziale è applicabile ad A .

Dimostrazione

Se il metodo fallisce al passo ℓ allora A non è invertibile perché si avrà che $a_{\ell\ell} = 0$, di conseguenza se l'elemento con il pivot massimo è 0, tutti gli altri elementi al disotto sono 0 (il GEPP funziona con i valori assoluti).

Calcoliamo il $\det(A)$ sviluppandolo rispetto alla prima colonna, procedendo in questo modo anche per la colonna 2 e così via.

Si arriva ad A_ℓ tramite operazioni che non modificano il determinante (combinazioni di righe) e scambi che non modificano il segno

$$|\det(A)| = |\det(A_\ell)| = 0 \implies \det(A) = 0$$

Il rango della matrice è quindi al più $k - 1 < n$, e quindi non è invertibile, di conseguenza non si può utilizzare il GEPP

Matrici Di Permutazione

Interpretiamo gli scambi di righe tramite operazioni tra matrici:

$T \in \mathfrak{S}_n \quad t : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ biiettiva

$$\varrho : t \rightarrow \Pi$$

$$f_1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, f_2 = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \dots, f_n = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

Example

$$\sigma(1) = 2, \quad \sigma(2) = 1, \quad n = 2$$

$$f_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, f_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$p(\sigma) = [f_{\sigma(1)} | f_{\sigma(2)}] = [f_2 | f_1] = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Date la matrice $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ e la matrice di permutazione p :

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} c & d \\ a & b \end{bmatrix}$$

Teorema

Sia data la [fattorizzazione LU con GEPP](#) esiste una matrice di permutazione Π tale che

$$\Pi A = LU$$

La matrice Π è ottenuta raccogliendo tutti gli scambi di riga effettuati durante [l'algoritmo](#)

Sono invertibili?

Se vogliamo risolvere il sistema $Ax = b$, le sue soluzioni coincideranno con $\Pi Ax = \Pi b \iff LU = \Pi b$ e quindi il problema si risolve permutando il vettore b e risolvendo la [fattorizzazione LU](#).

Oltre che a [risolvere sistemi lineari](#) l'Algoritmo di Gauss può essere usato per calcolare il determinante e l'inversa di una matrice .

L'osservazione sta nel fatto che il metodo di gauss non cambia il determinante della matrice, ma solo il suo segno (Se avvengono degli scambi di righe).

Quindi, con la variante del massimo pivot parziale, si hanno molti scambi di righe che devono essere contati. Si ottiene quindi che $\det(A) = \det(U)$ se gli scambi sono pari, e $\det(A) = -\det(U)$ se gli scambi sono dispari.

Si può scrivere quindi $\det(A) = (-1)^s \det(U)$, con s il numero di scambi di righe.

Ottenendo la triangolare superiore U , il determinante si trova solo sulla diagonale, e quindi

$$\det(U) = (-1)^s \prod_{i=1}^n u_{ii}$$

Il costo del prodotto è di $n - 1$ operazioni, quindi è trascurabile rispetto al metodo di Gauss.

a5. Curva di Bezier

Dati P_0, P_1, \dots, P_n punti di controllo, la curva

$$B(t) = \sum_{i=0}^n \binom{n}{i} t^i (1-t)^{n-i} P_i$$

è detta curva di Bezier

Esempio

N=1

$$\begin{aligned} B_{0,1}(t) &= \sum_{i=0}^1 \binom{1}{i} t^i (1-t)^{1-i} P_i = \\ &= \binom{1}{0} t^0 (1-t)^1 P_0 + \binom{1}{1} t^1 (1-t)^0 P_1 = \end{aligned}$$

$$= 1 \cdot 1(1-t)^1 P_0 + 1t(1-t)^0 P_1 = (1-t)P_0 + tP_1$$

Inviluppo convesso di n+1 punti

L'inviluppo convesso in $n + 1$ punti in R^n è il più piccolo insieme convesso che li contiene

Proprietà delle curve di bezier

Il vettore tangente (derivata) su B in 0 è $n(v_1 - v_0)$ e ci permette di determinare la retta tangente di γ in $\gamma(t_n)$ è $\gamma(t_0) + \gamma'(t_0)t$

a6. Modello della corda elastica

1. Trovare la posizione di una corda elastica soggetta a forza esterna.
2. Trovare una funzione che ne deriva la forma.
3. Trovare il valore di questa funzione in alcuni punti.

n masse uguali e equidistanti collegate tra loro da molle.

4. Scriviamo l'equazione dell'equilibrio per singola massa
 - Ogni massa si può muovere solo in verticale
 - L'equazione dell'equilibrio è $F_y = 0$
 - Le forze con componenti verticali su m_i sono
 - La forza peso
 - La forza elastica della massa m_{i+1}
 - La forza elastica della massa m_{i-1}

$$|F| = k(\ell - \ell_0) = k\ell, \quad \ell_0 = 0$$

Se y_i è la posizione verticale di m_i allora la forza che m_{i+1} esercita su m_i ha componente verticale $k(y_{i+1} - y_i)$

La massa m_{i-1} esercita una forza $k(y_{i-1} - y_i)$

L'equazione diventa

$$\begin{cases} -mg + k(y_{i+1} - y_i) + k(y_{i-1} - y_i) = 0 \\ -mg + k(y_2 - y_1) + k(-y_1) = 0 \\ -mg + k(-y_n) + k(y_{n-1} - y_n) = 0 \end{cases}$$

Le incognite sono y_1, \dots, y_n le incognite delle molle

$$\frac{-mg}{k} + y_{i+1} - y_i + y_{i-1} - y_i = 0$$

$$\begin{cases} -y_{i-1} + 2y_i - y_{i+1} = -\frac{mg}{k} \\ 2y_1 - y_2 = -\frac{mg}{k} \\ 2y_n - y_{n-1} = -\frac{mg}{k} \end{cases}$$

Il sistema ha n equazioni e n incognite, è un sistema lineare del tipo $Ax = b$, con A tridiagonale

a7. Interpolazione

Problema dell'interpolazione polinomiale

Dati $x_0, \dots, x_n \in [a, b]$ e dati $y_0, \dots, y_n \in \mathbb{R}$ trovare un polinomio $p(x)$ di grado al più n tale che:

$$p(x_0) = y_0, \quad p(x_1) = y_1, \quad p(x_n) = y_n$$

$$\begin{cases} a_0 + a_1x_0 + a_2x_0^2 + \dots + a_nx_0^n = y_0 \\ \vdots \\ a_0 + a_1x_n + a_2x_n^2 + \dots + a_nx_n^n = y_n \end{cases} \quad \text{è un sistema lineare}$$

$$V = \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{bmatrix} \quad \text{matrice di Vandermonde}$$

Note

Il problema è ben posto se e solo se V è invertibile:

$$\det(V) = \prod_{\substack{i>j \\ i,j=0}}^2 (x_i - x_j) = (x_1 - x_0)(x_2 - x_1)(x_2 - x_0)$$

Corollario

Il problema dell'interpolazione polinomiale è ben posto se e solo se i nodi sono distinti.

Dimostrazione

Siccome è un sistema lineare, deve essere $\det(V) \neq 0$ ma:

$$\det(v) \neq 0 \iff x_i - x_j \neq 0 \iff x_i \neq x_j \iff \text{i nodi sono distinti}$$

Esempio

$x_0, x_1 \in [a, b]$, $y_0, y_1 \in R$ il polinomio di interpolazione ha grado al più 1, quindi abbiamo soluzione unica

$$p(x) = a + bx$$

$$\begin{cases} a + bx_0 = y_0 \\ a + bx_1 = y_1 \end{cases} \implies \begin{bmatrix} 1 & x_0 \\ 1 & x_1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \end{bmatrix}$$

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1 & x_0 \\ 1 & x_1 \end{bmatrix}^{-1} \begin{bmatrix} y_0 \\ y_1 \end{bmatrix}$$

$$\frac{1}{x_1 - x_0}$$

L'utilizzo principale del polinomio di interpolazione

Polinomi di Lagrange Associati a x_0, \dots, x_n Distinti

- $$L_0(x) = \frac{(x-x_1)(x-x_2)\dots(x-x_n)}{(x_0-x_1)(x_0-x_2)\dots(x_0-x_n)}$$

- ...
- $L_i(x) = \frac{(x-x_1)(x-x_2)\dots}{x_i - x_j}$

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^n \frac{x - x_j}{x_i - x_j} \quad i = 1, \dots, n$$

Teorema

Siano $x_0, \dots, x_n \in [a, b]$ distinti e siano $L_0(x), \dots, L_n(x)$ i polinomi di Lagrange allora

1. $L_i(x_j) = \delta_{ij} : \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$
2. 2 polinomi di Lagrange sono una base di $R_n[x]$
3. Dati $y_0, \dots, y_n \in R$, il polinomio di interpolazione rispetto a $\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}, \dots, \begin{bmatrix} x_n \\ y_n \end{bmatrix}$ è

$$p(x) = \sum_{i=0}^n y_i L_i(x)$$

Dimostrazione

$$1. L_i(x_j) = \frac{(x_j - x_0) \dots (x_j - x_{i-1})(x_j - x_{i+1}) \dots (x_j - x_n)}{(x_i - x_0) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)} = 0, \quad i \neq j$$

$$L_i(x_i) = \frac{(x_i - x_0) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)}{(x_i - x_0) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)} = 1, \quad i = j$$

2. Siccome $R_n[x]$ ha dimensione $n + 1$ è sufficiente dimostrare che sono linearmente indipendenti

$$\sum \alpha_i L_i(x) = 0 \implies \alpha_i = 0 \quad i = 0, \dots, n$$

valutiamo la combinazione lineare in $x_j \quad j = 0, \dots, n$

$$0 = \sum_{i=0}^n \alpha_i L_i(x_j) = \alpha_j L_j(x_j) = \alpha_j$$

3. $p(x) = \sum_{i=0}^n y_i L_i(x)$ dobbiamo dimostrare che $p(x_j) = y_j$ poiché p ha grado al più n e i nodi sono distinti

$$p(x_j) = \sum_{i=0}^n y_i L_i(x_j) = y_j L_j(x_j) = y_j$$

Algoritmo per Il Calcolo Di p(x) E La Sua Valutazione

$$\sum_{i=0}^n y_i \prod_{j \neq i} \frac{x - x_j}{x_i - x_j} = \sum_{i=0}^n y_i \frac{\prod_{j \neq i} x - x_j}{\prod_{j \neq i} x_i - x_j}$$

Interpolazione per Approssimare Funzioni

Si può utilizzare l'interpolazione polinomiale per l'approssimazione di funzioni continue.

Data $f \in C[a, b]$ e dati $x_0, \dots, x_n \in [a, b]$ il polinomio di interpolazione è l'unico polinomio di grado **al più** n tale che $p(x_i) = f(x_i) = y_i$.

Una volta approssimata una funzione $f(x)$ con un polinomio $p_n(x)$ si ha un errore:

$$r(x) = f(x) - p_n(x)$$

detto **resto dell'interpolazione**

Teorema

Sia $f \in C^{k+1}[a, b]$ e sia $p_n(x)$ il polinomio di interpolazione relativo a $x_0, \dots, x_n \in [a, b]$ distinti. Allora $\forall x \in [a, b], \exists \xi(x) \in [a, b]$:

$$r(x) = f(x) - p_n(x) = \frac{f^{(k+1)}(\xi(x))}{(n+1)!} (x - x_0) \dots (x - x_n)$$

In generale non è detto che aumentando il numero dei nodi l'approssimazione migliori (anche se la funzione da approssimare è regolare).