MATEMATICA COMPUTAZIONALE

Libro: "Fondamenti di calcolo numerico" G. Monegato, CLUT 5/10/2005

Argomento di oggi interamente preso dal libro.

Sistema di equazioni lineari

$Ax = b$ $\quad A \in \mathbb{R}^{n \times m}$ (matrici), $b \in \mathbb{R}^m$ (vettore), x ?

metodi diretti (soluz. esatta in un numero finito di passi, alterando A)

iterativi (non alterano A, costruiscono una successione di vettori, convergono sotto opportune ipotesi alla soluzione esatta, arresta il processo appena ottenuta la precisione richiesta)

$A = \begin{pmatrix} \emptyset & \cdots & \emptyset \\ \vdots & \emptyset & \vdots \\ \emptyset & \cdots & \emptyset \end{pmatrix}$ matrice sparsa = quando il numero degli elementi nulli è molto grande. Per questa è meglio usare il metodo iterativo.

Quando la matrice è piena è meglio usare il metodo diretto.

Metodo di Gauss, eliminazione

A triangolare superiore $A = \begin{pmatrix} \times & \cdots & \times \\ & \ddots & \vdots \\ \emptyset & & \times \end{pmatrix}$ $\quad Ax = b$

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots + a_{1m}x_m = b_1 \\ \quad\quad a_{22}x_2 + a_{23}x_3 + \dots + a_{2m}x_m = b_2 \\ \quad\quad\quad\quad \vdots \\ \quad\quad\quad\quad a_{mm}x_m = b_m \end{cases}$$

Partendo dall'ultima $x_m = \frac{b_m}{a_{mm}}$ si va a ritroso.

$$x_k = \frac{\left(b_k - \sum_{j=k+1}^{m} a_{kj}x_j\right)}{a_{kk}}, \quad k = m-1, \dots, 1$$

Per la triangolare inferiore sarebbe la stessa cosa $A = \begin{pmatrix} \times & & \emptyset \\ \vdots & \ddots & \\ \times & \cdots & \times \end{pmatrix}$.

In sintesi vogliamo trasformare un generico sistema in uno equivalente di forma triangolare.

1) Quando ad una equaz. sostituiamo una comb. lineare dell'equazione con un'altra dello stesso sistema, il nuovo sistema risulta equivalente al precedente.

2) Il metodo di Gauss dimostra che è sempre possibile mediante un numero finito di combinazioni lineari di quel tipo ed eventuali permutazioni di equazioni, trasformare un generico sistema in un sistema equivalente di forma triangolare.

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 \dots a_{1m}x_m = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 \dots a_{2m}x_m = b_2 \\ \vdots \end{cases}$$

assumiamo $a_{11} \neq \emptyset$

elimino x_1 dalle ultime $n-1$ equazioni, sommando alla i-esima eq. la prima moltiplicata per $m_{i1} = -\frac{a_{i1}}{a_{11}}$, $i = 2 \dots n$

$a_{21} x_1 + a_{22} x_2 + \dots a_{2n} x_n = b_2$

$-\frac{a_{21}}{a_{11}} \left(a_{11} x_1 + a_{12} x_2 + \dots + a_{1n} x_n = b_1 \right)$] somma di queste due

$\emptyset \; x_1 \quad a_{22}^{(2)} x_2 + a_{23}^{(2)} x_3 + \dots = b_2^{(2)}$ RISULTATO DELLA SOMMA

Quindi ora abbiamo:

$a_{11} x_1 + a_{12} x_2 + \dots + a_{1n} x_n = b_1$

$\emptyset \quad a_{22}^{(2)} x_2 + a_{23}^{(2)} x_3 + \dots + a_{2n}^{(2)} x_n = b_2^{(2)}$

Applicando ripetutamente questa procedura ottengo un nuovo sistema dove x_1 sarà sempre $\emptyset$.

$$\begin{cases} a_{11} x_1 + a_{12} x_2 + \dots + a_{1n} x_n = b_1 \\ \quad \emptyset \quad a_{22}^{(2)} x_2 + \dots + a_{2n}^{(2)} x_n = b_2^{(2)} \\ \quad \emptyset \quad a_{32}^{(2)} x_2 + \dots + a_{3n}^{(2)} x_n = b_3^{(2)} \\ \quad \vdots \\ \quad \vdots \end{cases}$$

Possiamo applicare la procedura alla matrice più interna, partendo dalla seconda equazione.

Lo facciamo per tutte le matrici interne finché non otteniamo una matrice triangolare inferiore, ovvero $n-1$ volte.

Algoritmo

1) elimino le variabili in $n-1$ passi

al passo k-esimo, $k = 1 \dots n-1$

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

$m_{ik} = -\frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$, $b_i^{(k+1)} = b_i^{(k)} + m_{ik} b_k^{(k)}$

2) risolvere sistema triangolare superiore

Matematica computazionale NO! 6/10/2005

Non dà risultati specifici, ma modelli di calcolo.

La soluzione trovata sarà sempre approssimata, vincoli: accuratezza, velocità di calcolo, margine di errore.

ESEMPIO

$y(q) = \operatorname{sen} \log q$ equazione esplicita dove l'errore può essere nullo.

$y = e^{q^2} \operatorname{sen} y q$ equazione implicita (q e y non sono separate), da risolviamo ma in modo approssimato. (f nota non esplicitamente).

$y(q) = \sum_{k=0}^{\infty} a_k e^{-b_k q^2}$ anche se la f è nota esplicitamente non possiamo cal

colarla (a causa della somma degli ∞ termini).

$y(q) = \text{sen}(q)$ f. nota esplicitamente e calcolabile non direttamente (ma anche in questo caso c'è un errore)

$y(q) = 3q$ f nota esplicitamente, calcolabile direttamente (anche in questo caso abbiamo un errore, perchè non sappiamo come è q e come si può rappresentare con un numero finito di bit).

Quindi avremo sempre e comunque un margine di errore.

Il computer calcola il log con le serie.

Se dovessimo calcolare: $\text{sen}\log\sqrt{q^{200}}$ bisognerebbe procedere dall'interno verso l'esterno, invece il computer ne crea una espressione approssimata tipo $\sum a_k x^k$ ma estremamente più veloce da calcolare. Bisogna quindi sempre conoscere il margine di errore.

ESEMPIO

Quanto occupa una foto da 1024 x 768 a colori?

$3 \cdot 1024 \cdot 768 \,(\cdot 1) = 2{,}4$ Mb

3 → RGB: 3 valori da 0 a 255 ($256 = 2^8$)

(·1) perchè abbiamo supposto che un pixel occupa un byte di memoria

Se devo spedire la foto con un modem a 40 Kb/s?

$$\frac{(2{,}4 \cdot 8) \cdot 1024}{40} = 500\ s \simeq 8\ \text{min}$$

ho convertito i byte in bit

Possiamo convertirla in Jpeg che si basa sulla tecnica lossy, basata sulla trasformata di Fourier.

19/10/2005

Gauss

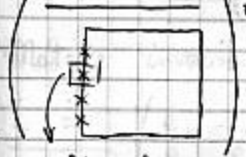
Il procedimento delle eliminazioni di Gauss può essere portato a termine senza permutare l'ordine iniziale delle eq. se A diag. dominante (per righe o colonne) o simmetrica def. positiva.

$-\frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} = m_{ik}$ $\quad \left|\frac{a_{kk}}{\neq \emptyset}\right| \leftarrow$ pivot (mai in questo caso) $\quad |a_{ii}| > \sum_{i \neq j} |a_{ij}|$ DIAG. DOMINANTE

$a_{ij} = a_{ji}$ simmetrica

A def. pos. $x^T A x > \emptyset \quad \forall x \neq \emptyset$

Tecnica del pivot parziale



$a_{22}^{(2)}$ → molto piccolo

scelgo valore ass. più grande e scambio le due righe

$a_{kk}^{(k)} = \max_i |a_{ik}^{(k)}|$

Fattorizzazione LU (lower - upper)

Interpretare il metodo di Gauss come successione finita di trasformazione di A e b, cioè moltipl. di A e b, per opportune matrici.

Riformulare l'algoritmo di Gauss in 2 parti:

1) $\underset{\sim}{G} \cdot \rightarrow GA = U$ ← triangolare sup.

2) $GAx = Gb$, $\bar{b} = Gb$ $\quad Ux = \bar{b}$

Vediamo 1) scambio eq. i-esima eq. j-esima

$P_{ij} = \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{pmatrix}$ (i-esima, j-esima)

$I = \begin{pmatrix} 1 & \emptyset \\ \emptyset & 1 \end{pmatrix}$ $\quad P_{12} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ → ho scambiato le righe di I

$A = \begin{pmatrix} 4 & 3 \\ 2 & 5 \end{pmatrix}$ $\quad P_{12}A = \begin{pmatrix} 2 & 5 \\ 4 & 3 \end{pmatrix}$

La sostituzione dell'eq. i-esima con la stessa più la j-esima moltiplicata per m_{ij}

$H_{ij} = \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & m_{ij} & & 1 \end{pmatrix}$ ← elemento

$P_1, P_2 \dots P_{n-1}$ (I se non scambio)

P_{ij}

$H_1, H_2 \dots H_{n-1}$, $H_j = H_{nj} \dots H_{j+1,j} = \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & m_{j+1,j} & 1 & \\ & & m_{j+2,j} & & 1 \\ & & \vdots & & \\ & & m_{n,j} & & \end{pmatrix}$

Di fatto Gauss dice che:

$Ax = b$

$H_1 P_1 Ax = H_1 P_1 b$] 1° passo

$H_2 P_2 H_1 P_1 Ax = H_2 P_2 H_1 P_1 b$] 2° passo

$\vdots$

$H_{n-1} P_{n-1} \dots H_1 P_1 Ax = H_{n-1} P_{n-1} \dots H_1 P_1 b$] ultimo passo

Quindi G sarà $= H_{n-1} P_{n-1} \dots H_1 P_1$ costo computazionale $\frac{n^3}{3}$ relativamente basso

y verrà utilizzata per trasformare vettori e matrici, serve memorizzare m_{ij} e eventuali permutazioni

$$\begin{pmatrix} m_{11} & m_{12} & m_{13} & \dots \\ m_{21} & m_{22} & m_{23} & \dots \\ m_{31} & m_{32} & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Non serve memorizzare tutte le matrici P, basta memorizzare vettore pivot:

pivot $(k) = r$ SCAMBIO $k \leftrightarrow r$

pivot $(k) = k$ NON SCAMBIO

$H_{n-1} P_{n-1} \dots H_1 P_1 A = H_{n-1} P_{n-1} \dots H_1 P_1 b$

$\underbrace{H_{n-1} \dots H_1}_{\bar{H}} \quad \underbrace{P_n \dots P_1 A}_{P}$

raccolta matrici

$\bar{H} P A = U$

$P A = \bar{H}^{-1} U$

$$L = \bar{H}^{-1} = \begin{pmatrix} 1 & & & \\ -\bar{m}_{21} & 1 & & 0 \\ -\bar{m}_{31} & -\bar{m}_{32} & 1 & \\ \vdots & \vdots & & \ddots \\ & & & 1 \end{pmatrix}$$

$PA = LU$

$PAx = Pb \rightarrow LUx = Pb$

$Ly = Pb$

$Ux = y$

matrice triangolare superiore che si può ricavare, dopodiché mi trovo x (le altre cose sono già note)

Ho scomposto le cose in due matrici triangolari, una superiore e l'altra inferiore.

27/10/2005

Fattorizzazione LU

Se non ci sono scambi $P = I \Rightarrow A = LU = LUD_1$ (0)

con L e D_1 triang. unitarie.

Se A è simmetrica $U_1 = L^T$ (1) se poi A è def. pos. $\Rightarrow d_{ii} > 0$ (2) $\Rightarrow$ A simmetrica def. pos. $A = L_1 L_1^T$, $L_1 = LD^{1/2}$ $\left(D^{1/2} = \sqrt{d_{ii}}\right)$

$l_{ii} = \sqrt{a_{ii}}$

$l_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}\right) / l_{jj}$, $j = 1 \dots i-1$

$l_{ii} = \left(a_{ii} - \sum_{k=1}^{i-1} l_{ik}\right)^{1/2}$

ALGORITHO CHOLESKI

lo chiede all'esame

CHOLESKI $\frac{n^3}{6}$ COMPLESSITÀ COMPUTAZIONALE

Matrice inversa

$$L = \begin{pmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ l_{31} & l_{32} & l_{33} & \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

$Y = L^{-1} = \begin{pmatrix} y_{11} & & & \\ y_{21} & y_{22} & & \\ y_{31} & y_{32} & y_{33} & \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$

inversa sempre in forma triangolare inferiore

$y_{jj} = \frac{1}{l_{jj}}$ $j = 1 \dots n$

$y_{ij} = -\left(\sum_{k=j}^{i-1} l_{ik} y_{kj}\right) / l_{ii}$

$i = j+1 \dots n-1$

Per calcolare l'inversa di una matrice generica è molto difficile, ma con le cose viste precedentemente diventa banale.

Come calcolare l'inversa

$GA = U$

$U^{-1} GA = U^{-1} U = I$ → moltiplicato per U^{-1} entrambi i membri

$U^{-1} GA = I \Rightarrow A^{-1} = U^{-1} G$ → matrice triangolare superiore

↓ se moltiplico per A^{-1} entrambi i membri

$PA = LU$ ci può servire per calcolare A^{-1}!

$P^{-1} PA = P^{-1} LU$

$A = P^{-1} LU$

$AU^{-1} = P^{-1} L$ (moltiplicato per U^{-1})

$AU^{-1} L^{-1} = P^{-1}$ (moltiplicato per L^{-1})

$AU^{-1} L^{-1} P = I$

$A^{-1} = U^{-1} L^{-1} P$ → porto A dall'altra parte che diventa A^{-1}

già conoscevo P, L, U, per le inverse abbiamo già visto gli algoritmi, quindi posso tranquillamente calcolare la matrice inversa di A

Calcolo degli autovalori di A

$Ax = \lambda x$, $x \neq \emptyset$, $\lambda \in \mathbb{C}$

x autovettore, λ autovalore

ci sono n autovettori e n autovalori

$(A - \lambda I) x = \emptyset$ → ha portato λ dall'altra parte e RACCOLTO LA x

$x \neq \emptyset$

Voglio che abbia anche una soluzione non nulla. → $\det(A - \lambda I) = \emptyset$ → vuol dire che ha più di una soluzione.

$= \lambda^n + \alpha_1 \lambda^{n-1} + \alpha_2 \lambda^{n-2} + \dots$

POLINOMIO CARATTERISTICO

Sto cercando le n radici di POL. CAR.

λ incognita nei reali o nei complessi.

Ci sono vari algoritmi, a seconda di quali autovalori vogliamo trovare.

Teorema di localizzazione (Gerschgorin)

Localizza l'area dei complessi dove posso pescare gli autovalori

$r_i = \sum_{j \neq i} |a_{ij}|$, $i = 1 \dots n$ RIGHE

$c_j = \sum_{j \neq i} |a_{ij}|$, $j = 1 \dots n$ COLONNE

$R = \bigcup_{i=1}^{n} R_i$ $\quad C = \bigcup_{j=1}^{n} C_j$

$R_i = \{ z_i \in \mathbb{C} \mid |z_i - a_{ii}| \leq r_i \}$ $\quad C_j = \{ z_j \in \mathbb{C} \mid |z - a_{jj}| \leq C_j \}$

ogni autovalore $\in$ insieme R

$\in$ insieme C

$R \cap C$ ← SOMMA DEI MODULI DEI RESTANTI

Esempio

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 4 \\ \frac{1}{2} & 0 & 4 \end{pmatrix}$$

$R_1 = \{ |z-1| \leq 0 \}$ (*)

$R_2 = \{ |z-2| \leq 4 \}$

$R_3 = \{ |z-4| \leq \frac{1}{2} \}$

3 righe

UNIONE

l'unione dei tre cerchi darà l'insieme dove cercare gli autovalori

INTERSEZIONE TRA QUESTI DUE

$C_1 = \{ |z-1| \leq \frac{1}{2} \}$

$C_2 = \{ |z-2| \leq 0 \}$ (*)

$C_3 = \{ |z-4| \leq 4 \}$ UNIONE DEI TRE CERCHI

3 colonne

(*) quando il cerchio collassa ad un unico punto $\lambda = 1$, $\lambda = 2$

l'altro autovalore va cercato nell'area risultata.

Metodo delle potenze

Serve a trovare l'autovalore di massimo modulo, ammesso che ci sia un unico autovalore $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq |\lambda_4| \dots$

$x_1\ x_2 \dots x_m$ sistema linearmente indipendente

$\lambda_i \in \mathbb{R}$ → perché se fosse complesso anche il suo coniugato sarebbe maggiore degli altri

$v_0 = \alpha_1 x_1 + \alpha_2 x_2 + \dots + \alpha_m x_m$ → generico vettore

$A v_0 = \alpha_1 A x_1 + \alpha_2 A x_2 + \dots + \alpha_m A x_m$ → moltiplico per matrice A

$A x_1 = \lambda_1 x_1$ ...

$= \alpha_1 \lambda_1 x_1 + \alpha_2 \lambda_2 x_2 + \dots + \alpha_m \lambda_m x_m =$

$A^2 v_0 = AA v_0 = A(\alpha_1 \lambda_1 x_1 + \alpha_2 \lambda_2 x_2 + \dots + \alpha_m \lambda_m x_m)$

$= \alpha_1 \lambda_1 A x_1 + \alpha_2 \lambda_2 A x_2 + \dots + \alpha_m \lambda_m A x_m$

$= \alpha_1 \lambda_1 \lambda_1 x_1 + \alpha_2 \lambda_2 \lambda_2 x_2 + \dots + \alpha_m \lambda_m \lambda_m x_m$

$= \alpha_1 \lambda_1^2 x_1 + \alpha_2 \lambda_2^2 x_2 + \dots + \alpha_m \lambda_m^2 x_m =$

$A^k v_0 = \alpha_1 \lambda_1^k x_1 + \alpha_2 \lambda_2^k x_2 + \dots + \alpha_m \lambda_m^k x_m$

$= \lambda_1^k \left(\alpha_1 x_1 + \alpha_2 \frac{\lambda_2^k}{\lambda_1^k} x_2 + \dots + \alpha_m \frac{\lambda_m^k}{\lambda_1^k} x_m \right)$

$\frac{\lambda_2}{\lambda_1} < 1$

$\lim_{k \to \infty} \frac{(v_{k+1})_j}{(v_k)_j} = \frac{\lambda_1^{k+1} \left(\alpha_1 x_1 + \alpha_2 \frac{\lambda_2^{k+1}}{\lambda_1^{k+1}} x_2 + \dots \right)}{\lambda_1^k \left(\alpha_1 x_1 + \alpha_2 \frac{\lambda_2^k}{\lambda_1^k} x_2 + \dots \right)_j} =$

I TERMINI SUCCESSIVI AL PRIMO SI SEMPLIFICANO TUTTI TRA LORO

$= \lambda_1 \left(\frac{\alpha_1 x_1 + 0}{\alpha_1 x_1 + 0} \right) = \lambda_1$ → converge all'autovalore di massimo modulo.

Metodo delle potenze

2/11/2005

Converge tanto più rapidamente quanto più $|\lambda_1| > |\lambda_2|$ (x discosta dai successivi)

*) λ autovalore per $A \Rightarrow \lambda^{-1}$ autovalore per A^{-1} perché

$Ax = \lambda x \Rightarrow A^{-1}Ax = \lambda A^{-1}x \Rightarrow x = \lambda A^{-1}x \Rightarrow \lambda^{-1}x = A^{-1}x$

Il metodo delle potenze può essere utilizzato per migliorare l'approssimazione di un dato autovalore ρ approssimato di λ autovalore di A:

$(A - \rho I)x = Ax - \rho x = \lambda x - \rho x = (\lambda - \rho)x$ ⟶ MATRICE (A) × VETTORE (x) = VETTORE (x) × SCALARE (λ)

$\lambda - \rho$ autovalori di $A - \rho I \overset{(*)}{\Rightarrow} (A-\rho)^{-1}$ autovalore di $(A - \rho I)^{-1}$

Chiamiamo $\mathcal{H} = (\lambda - \rho)^{-1}$ immaginiamo che ρ sia approssimazione ACCURATA buona

$\rho \simeq \lambda$

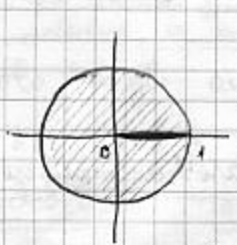
$\lambda - \rho$ è quasi $\emptyset$ e il suo reciproco è molto grande

quindi $\mathcal{H}$ autovalore di max modulo

$(A - \rho I)^{-1}$

quindi uso il metodo delle potenze e trovo una quantità più accurata per $\mathcal{H}$:

$\frac{1}{\mathcal{H}} = \lambda - \rho \qquad \lambda = \rho + \frac{1}{\mathcal{H}}$



l'autovalore x trova nel cerchio di raggio 1 vicino a $\rho = 1$ e con questo metodo si fa una migliore approssimazione (raffinamento) ⟶ METODO DELLE POTENZE INVERSE

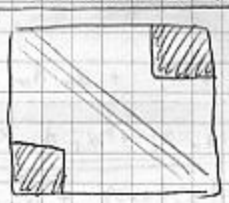
matrice con molti $\emptyset$

Torniamo a Gauss e ai sistemi iterativi su esso

$Ax = b \qquad x = ?$ con i 3 metodi iterativi ?

applicati a matrici con struttura regolare con pochi elementi non nulli

quasi tutti $\emptyset$



$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1m}x_m = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2m}x_m = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \dots = b_3 \\ \vdots \end{cases}$$

$$x_1 = \frac{b_1 - \sum_{j \neq 1} a_{1j}x_j}{a_{11}}$$

$$x_2 = \frac{b_2 - \sum_{j \neq 2} a_{2j}x_j}{a_{22}}$$

$$x_3 = \frac{b_3 - \sum_{j \neq 3} a_{3j}x_j}{a_{33}}$$

input, vettore di partenza

$x^{(0)} = (x_1^{(0)} \dots x_m^{(0)})$

Dato un vettore di partenza ⟶ devo trovare un vettore approssimato decente.

$x^{(1)}, x^{(2)}, x^{(3)} \longrightarrow \bar{x}$

$$x_1^{(1)} = \frac{b_1 - \sum a_{1j}x_j^{(0)}}{a_{11}}$$

$x_2^{(1)} = \dfrac{b_2 - \sum_{j \neq 2} a_{2j} x_j^{(0)}}{a_{22}}$ → informazione data in input

$x_3^{(1)} = \dfrac{b_3 - \sum_{j \neq 3} a_{3j} x_j^{(0)}}{a_{33}}$

$x_4^{(1)} = \dots$

$x_5^{(1)} = \dots$

ecc...

SI PUÒ SCRIVERE ANCHE: $x_2^{(1)} = \dfrac{b_2 - a_{21} x_1^{(1)} - \sum_{j \neq 2,1} a_{2j} x_j^{(0)}}{a_{22}}$ (numero della prima parola)

OVVIAMENTE PER TUTTE LE INCOGNITE

senza correzione si chiama JACOBI

con questa correzione si chiama metodo di GAUSS - SEIDEL (più facile)

→ al primo giro devo ottenere un vettore $x^{(1)} = (x_1^{(1)}, x_2^{(1)}, \dots, x_m^{(1)})$

di fatto alla fine posso scrivere: $x_1^{(k+1)} = \dfrac{b_1 - \sum_{j \neq 1} a_{1j} x_j^{(k)}}{a_{11}}$ e così via a tutti gli altri passi

$x_2^{(k+1)} = \dots$

$x_3^{(k+1)} = \dots$

ecc...

ora faccio un ulteriore raffinamento al secondo passo:

$x_1^{(2)} = \dfrac{b_1 - \sum_{j \neq 1} a_{1j} x_j^{(1)}}{a_{11}}$

per tutte le altre incognite...

dopo un po' di giri dovrei avere una buona approssimazione

Terzo metodo (del sovrarilassamento)

$x_i^{(k+1)} = x_i^{(k)} + \omega r_i^{(k)}$ → ancora è un mistero per gli studiosi quale sia l'ω perfetto

$r_i^{(k)} = \dfrac{1}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i}^{m} a_{ij} x_j^{(k)} \right]$ → $\dfrac{b_i - \sum_{j \neq i} a_{ij} x_j^{(k-1)}}{a_{ii}} + \dfrac{1}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i}^{m} a_{ij} x_j^{(k)} \right]$

Esercizio (per casa) FARLO BENE!

Verificare che scegliendo $\omega = 1$ il metodo del sovrarilassamento si riduce a metodo di Gauss - Seidel. → $x_i^{(k)} = \dfrac{b_i - \sum_{j \neq i} a_{ij} x_j^{(k-1)}}{a_{ii}}$

9/11/2005

Approfondimento sulla lezione scorsa

I metodi convergono quando la matrice A è a diagonale dominante per righe (PER RIGHE $|a_{ii}| \geq \sum_j |a_{ij}|$) o per colonne (PER COLONNE $\left(|a_{ii}| \geq \sum_k |a_{ki}|\right)$). → LO CHIEDE SEMPRE ALL'ESAME!!!

$x_2^{(1)} = \dfrac{b_2 - a_{21} x_1^{(1)} - \sum_{j \neq 1,2} a_{2j} x_j^{(0)}}{a_{22}}$ → GAUSS - SEIDEL

EQUAZIONI NON LINEARI

$f(x) = 0$

$x^{2,1} - 7x + 3 = 0$ → come la risolvo?

$x^{10} - 7x + 7x^3 - 15x^2 = 0$

$x^2 - \cot g \cos \lg x - 7 = 0$

} SICURAMENTE PIÙ DIFFICILI DELLE EQ. LINEARI

Capita spesso di trovarle in fisica ad es...

Ci vogliono algoritmi che approssimano per poterle risolvere.

$x_1, x_2, x_3 \dots x_m \to x$ $\quad f(x) = \emptyset$

si risolvano con i metodi iterativi

$x_{m+1} = g(x_m)$ $\quad g$ funzione di iterazione

PROCED. ITERATIVO

$x_1 = g(x_0)$

$x_2 = g(x_1)$

$x_3 = g(x_2)$

Noi presenteremo l'algoritmo di NEWTON e NEWTON - RAPHSON

Newton - Raphson

Si parte da x_0, e basato sulla formula precedente.

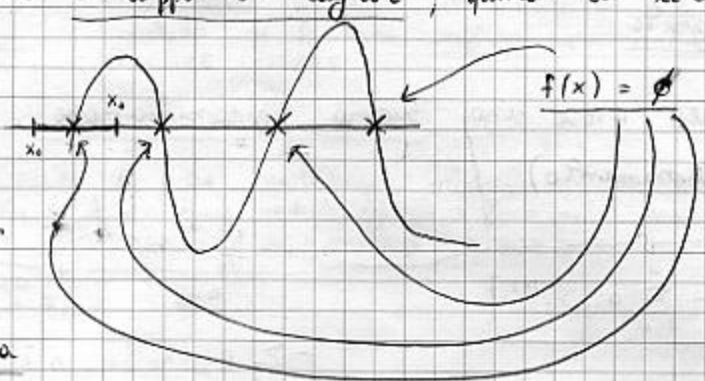
$g(x) = x - \frac{f(x)}{f'(x)}$ → derivata

$x_{m+1} = g(x_m) = x_m - \frac{f(x_m)}{f'(x_m)}$ → FORMULA

1) $f' \neq \emptyset$

2) si basa sullo sviluppo di Taylor, quindi la scelta di x_0 non deve essere casuale:

bisogna scegliere un intervallo molto piccolo in cui sono sicura che ricada una sola radice.

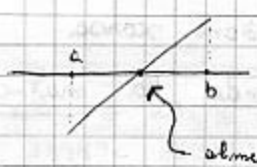


3) $f'' > \emptyset$ nell'intervallo dove sto lavorando (esclude casi in cui ci sono flessi - quindi non ci sono più radici, ma una -).

$\emptyset$) $f(a) f(b) < 0$ $\quad [a, b]$

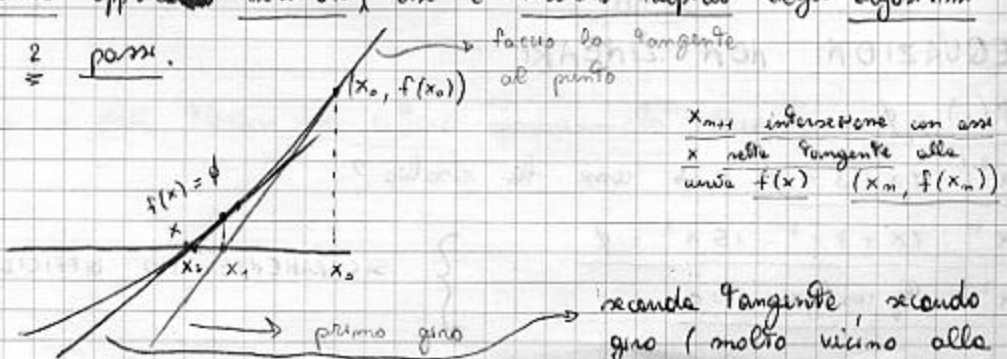
↓ ipotesi di prima

così mi assicuro di avere almeno una radice.



almeno un punto taglia l'asse x

Dopo aver fissato questi punti (INTERVALLO) applico Newton (che è molto rapido degli algoritmi lineari), si fa in 1 o 2 passi.



x_{m+1} intersezione con asse x nella tangente alla curva $f(x)$ $\quad (x_m, f(x_m))$

soluzioni x)

Metodo molto rapido, ma devo scegliere attentamente l'intervallino di partenza.
Ma quando mi devo fermare con le iterazioni?
test di convergenza
Sono due.

$|x_m - x_{m+1}| \leq x_{toll}$
↓
tolleranza

$|x_m - x_{m+1}| \leq x_{toll} |x_{m+1}|$

} DEVO APPLICARE (IMPLEMENTARE) TUTTI E DUE 'STI TEST

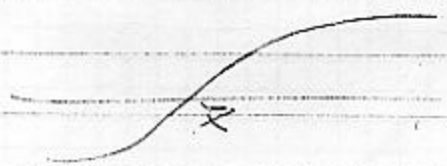
23/10/05

EQ NON LINEARI

$F(x) = 0$

$x'' + 7x + \cos x$

. Def. $f : A \subseteq \mathbb{R} \to \mathbb{R}$ $x \in A$ è p.to di accumulazione



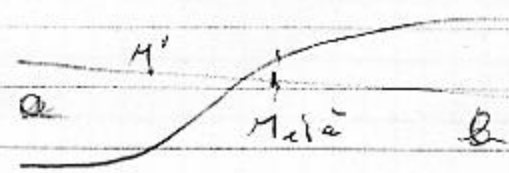
(1) $f(a) f(b) < 0$
↑ IPOTESI

$f(\bar{x}) = 0$

(2) f continua

C'è Metodo Bisezione per risolvere eq non lineare
↓
METODO BISEZIONE PER RISOLVERE EQ NON LINEARI

Divido ogni volta una parte ev.



$[a, M]$

$f(a) f(M) < 0$

ok!

Test di Convergenza

(1) $|f(x_n)| < \varepsilon$

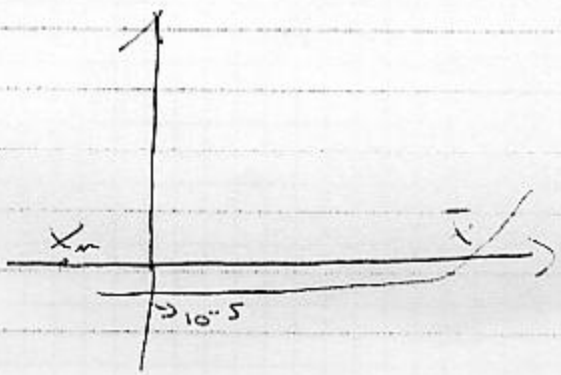
Successione $x_1\ x_2\ x_3 \ldots$ tale che
SUCCESSIVO

$f(\bar{x}) = 0$

↳ tende
TENDE A $\emptyset$

Vale il test (1)?

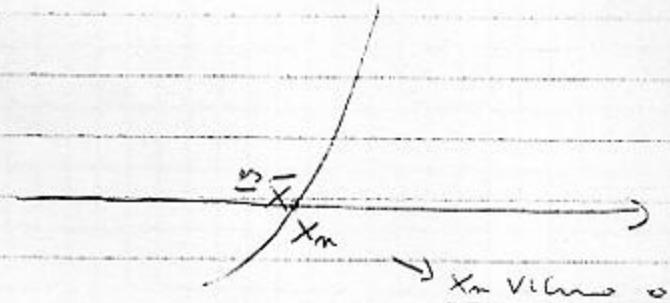
es.



$|f(x_n)| < \varepsilon$ ⇒ NO!

x_n ANCHE SE È PASSATO
anche se è passato
è lontano da $\bar{x}$
È LONTANO DA $\bar{x}$

Esempio



x_n vicino a $\bar{x}$

ma $f(x_n)$ non è passato
Non è passato
NO!
è quasi infinito
QUASI INFINITO

• Si vuole uno stimo così:

$$|x_n - x_{n+1}| \leq \text{xtoll}$$

$$|x_n - x_{n+1}| \leq \text{xtoll}\,|x_{n+1}|$$

Argomenti oss. finora

GIOVEDÌ
CONTINUATO TRASFORMATA E MINIMI QUADRATI
GENERALIZZATO Trasformata & Minimi Quadrati

• Macchine numeriche

• Prima lezione

LEZIONE PROFESSORE

ZERI ALGEBRICI

$$P_N(x) = x^N + a_1 x^{N-1} + a_2 x^{N-2} \dots a_N$$

$$\pi = \max |a_N| \qquad C = \{ |z| \leq 1 + \pi \}$$

N addd $\quad 2N-2$ moltiplic $\quad x^2 = x \cdot x$

es) grado 3

$$P_3(x) = x^3 + a_1 x^2 + a_2 x + a_3$$

$$= a_3 + x\left(a_2 + x\left(a_1 + x\right)\right)$$

DENTRO OGNI PARENTESI TORNA ABBIAMO
Dentro ogni parentesi torna abbiamo
UN TERMINE NOTO E X
un termine noto e x

Metodo Horner → per punto x fissato

$$\begin{cases} p_0 = 1 \\ p_k = x\, p_{k-1} + a_k, \quad k = 1 \dots n \end{cases}$$

- Nostro caso ADESSO CASO

$$p_1 = x + a_1$$

$$p_2 = x(x + a_1) + a_2 = x^2 + a_1 x + a_2$$

$$p_3 = x\left(x^2 + a_1 x + a_2\right) + a_3$$

- Perché no somma? PERCHÉ NO SOMMA
  → per calcolare la derivata LA DERIVATA CALCOLARE
  punto x fissato
  Horner richiedeva ogni passaggio di calcolare
  Newton richiedeva la derivata prima
  PUNTO X FISSATO PASSAGGIO DI CALCOLARE

Se voglio calcolare $p_n'(x_m)$ per Newton
DERIVATA

Costruisco polinomio aggiuntivo $q_{n-1}(x) = \dfrac{p_n(x) - p_n(x_1)}{x - x_1}$
COSTRUISCO POLINOMIO AGGIUNTIVO

Con $x \to x_1$ $\quad q_{n-1}(x) \to p_n'(x)$

- È vantaggioso rispetto a Newton
  È VANTAGGIOSO RISPETTO A NEWTON

Si dimostra che $p_{n-1}(x) = x^{n-1} + b_1 x^{n-2} + \dots b_{n-1}$

dove $b_0 = 1$

$$b_k = b_{k-1} x + a_k \qquad , k = 1 \dots n-1$$

- Supponiamo di aver trovato approssimazione grossolana per $\tilde{x}_1$ come lo trovo per le altre?

Vediamo prossima volta.

30/12/2005

LAGRANGE

$x_i^{(0)}$ approssimazione delle radici x_i.

Calcolo x_i con il metodo di Newton. Uso $p_n(x_i)$ e $p_n'(x_i)$.

Costruisco il polinomio ridotto.

$q_{n-1}(x) = \dfrac{p_n(x) - p_n(x_1)}{x - x_1}$ ha per radici le stesse radici di p_n tranne x_1. Calcolo così x_2.

Poi:

$q_{n-2}(x) = \dfrac{q_{n-1}(x) - q_{n-1}(x_2)}{x - x_2}$ e così via; fino ad ottenere un polinomio di grado 2 di cui sappiamo calcolare le radici. Le radici vanno calcolate in ordine crescente $|x_1| \le |x_2| \le \dots$ (gli errori si accumulano nel corso della procedura).

Interpolazione lagrangiana

Supponiamo di avere dei punti P, Q nel piano. Vogliamo trovare la retta L_1 (interpolante di Lagrange di grado 1) che passa per P e Q. ($L_1 = ax + b$).

Per due punti, otteniamo un sistema di due equazioni in due incognite la cui soluzione sono i coefficienti a e b. Consideriamo un ulteriore punto T. La curva L_2 che passa per P, Q e T ha equazione: $L_2(x) = a_0 + a_1 x + a_2 x^2$.

Quindi, dati n punti del piano, $L_n(x) = a_0 + a_1 x + \dots + a_n x^n$ tale che $L_n(x_i) = y_i$ (polinomio lagrangiano).

$(x_0, y_0) \dots (x_n, y_n)$

Si dimostra che il polinomio interpolante di Lagrange esiste ed è unico.

In generale, per n punti, questa procedura è la peggiore.

$$\begin{cases} L_n(x_0) = y_0 = a_0 + a_1 x_0 + \dots + a_n x_0^n \\ L_n(x_1) = y_1 = a_0 + a_1 x_1 + \dots + a_n x_1^n \\ \vdots \\ L_n(x_n) = y_n = a_0 + a_1 x_n + \dots + a_n x_n^n \end{cases}$$

sistema di $n+1$ equazioni in $n+1$ incognite $(a_0 \dots a_n)$

$$\begin{vmatrix} 1 & x_0 & x_0^2 & \dots & x_0^m \\ 1 & x_1 & x_1^2 & \dots & x_1^m \\ \vdots & & & & \\ 1 & x_m & x_m^2 & \dots & x_m^m \end{vmatrix} \neq \emptyset \iff$$ soluzione esiste ed è unica

Questa è una matrice del tipo VAN DERMONDE, ovvero è malcondizionata.

Esistono perciò altre tecniche, più stabili.

Interpolazione lagrangiana con metodo delle differenze divise (FINITE) → per le parentesi quadre (o metodo di Newton)

$L_m(x) = y_0 + (x - x_0)[x_0, x_1] + (x - x_0)(x - x_1)[x_0, x_1, x_2] + \dots + (x - x_0)\dots(x - x_m) \cdot [x_0, x_1 \dots x_m]$

se $a \to b$ $[a, b]$ è la derivata prima di $f(x)$

$[a, b] = \frac{f(b) - f(a)}{b - a}$ $\quad f(x_i) = y_i$ $\quad f'(x)$ $\quad a \to b$

Conosciamo $f(x)$ solo nei punti (x_i, y_i) e vogliamo sapere come si comporta f nei punti non assegnati.

$[x_0, x_1, x_2] = \frac{[x_1, x_2] - [x_0, x_1]}{x_2 - x_0}$ $\quad f''(x)$

Questa rappresentazione ha senso anche se i punti sono vicini fra loro.

$f(x) - L_m(x) = R_m(x)$ errore o resto dell'interpolazione lagrangiana

$R_m(x) = (x - x_0) \dots (x - x_m) \underbrace{[x, x_0, x_1, \dots, x_m]}_{\text{DIFFERENZA DIVISA}}$

Esiste un teorema che afferma $[x, x_0 \dots x_m] = \frac{f^{m+1}(\xi)}{m+1}$

ξ giace all'interno dell'intervallo, ma non è possibile conoscerne la posizione esatta.

Supponiamo f e P_m → polinomio di grado m, es. $f(x) = x^m \Rightarrow f^{m+1}(x) = \emptyset$

$\Rightarrow R_m(x) = \emptyset \Rightarrow f(x) = L_m(x)$ → se resto è $\emptyset$, la funzione è uguale al polinomio lagrangiano

In generale, l'interpolante di Lagrange non è un buono strumento per l'approssimazione di una funzione, poiché aumentano i dati di input la funzione trovata non tende a "abbracciare" quella reale.

$l_{m,k} = \frac{(x - x_0) \dots (x - x_{k-1})(x - x_{k+1}) \dots (x - x_m)}{(x_k - x_0) \dots (x_k - x_{k-1})(x_k - x_{k+1}) \dots (x_k - x_m)}$

$\lambda_m = \max_{[x_0 \dots x_m]} \sum_{k=0}^{m} |l_{m,k}(x)|$ costante di Lebesgue

$|R_m(x)| \leq (1 + \lambda_m) E_m(f)$ $\quad \lambda_m E_m \overset{?}{\longrightarrow} \emptyset$

errore di migliore approssimazione uniforme

Dalla teoria sappiamo che $\lambda_m \geq c \log m \iff \lambda_m$ non è limitata (cresce al crescere di m)

Domanda all'esame

Nel caso migliore, $\lambda_m \simeq c \log_m$ — x_i zeri di Cebicef

altrimenti $\lambda_m \simeq e^n$ — x_i equidistanti

PROGETTO → $I_{ij} \rightarrow I_{ij}^{\varepsilon} = I_{ij} + \varepsilon_{ij}$

Generazione di numeri casuali.

7/12/2005

Interpolazione di Lagrange

$$[a,b] = \frac{f(b)-f(a)}{b-a}$$

$$[a,b,c] = \frac{[b,c]-[a,b]}{c-a}$$

$$[a,b,c,d] = \frac{[b,c,d]-[a,b,c]}{d-a}$$

$$[a,a] = f(a)$$

$$[a,a,a] = \frac{f''(a)}{2}$$

$$\ell_{m,k}(x) = \frac{(x-x_0)\dots(x-x_{k-1})(x-x_{k+1})\dots(x-x_m)}{(x_k-x_0)\dots(x_k-x_{k-1})(x_k-x_{k+1})\dots(x_k-x_m)}$$

polinomio fondamentale di Lagrange

$$L_m(f,x) = \sum_{k=0}^{m} \ell_{m,k}(x)\, y_k$$

utile se molti y_k sono $= \phi$.

vantaggio: se $x_i \simeq x_{i+1}$

costante di Lebesgue

$$\lambda_m = \max_{[x_0,x_m]} \sum_{k=0}^{m} |\ell_{m,k}(x)| \geq c \log m$$

$$|f(x) - L_m(f,x)| \leq \underbrace{c}_{\text{costante}} (1+\lambda_m) \underbrace{E_m(f)}_{\text{errore di migliore approssimazione uniforme}}$$

$|f(x) - L_m(f,x)|$ = resto $R_m(f,x)$

→ il nostro scopo è che questa quantità vada a zero

se $\lambda_m E_m(f) = \phi \Rightarrow f(x) - L_m(f,x) \rightarrow \phi$

$\lambda_m E_m(f) \xrightarrow{?} \phi$

λ_m su nodi equidistanti $\lambda_m \simeq e^n$

$E_m(f) \rightarrow \phi$ regolarità di f

$E_m(f) = \min_{p \in \mathbb{P}_m[x_0,x_m]} \max |f(x) - p(x)|$ → errore di migliore approssimazione uniforme

Una domanda all'esame può essere: quali sono vantaggi e svantaggi dei due approcci di Lagrange?

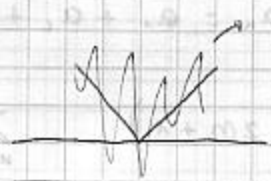
Esempio chiesto all'esame

$f(x) = x^{3/2}$ $[0,1]$

$f \in C^1$

$E_m(f) \sim \frac{1}{m}$

$\lambda_m E_m(f) \sim \log m \cdot \frac{1}{m} \rightarrow \phi$



interpolante di Lagrange

già con 5 punti c'erano fenomeni di overflow e underflow

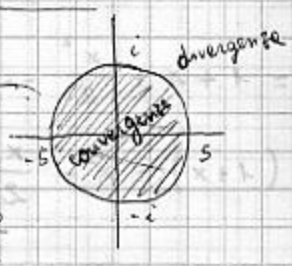
Commento sulla convergenza (fenomeno di Runge)

$f(x) = \frac{1}{1+x^2}$ RUNGE

$L_n(f) \not\to f$

posso avere tutte le derivate che voglio

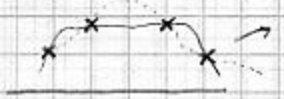
Se estendiamo lo studio nel campo dei complessi → $i^2 = -1$
avrei divisione per zero (ha delle singolarità)



dentro questo cerchio non ci sono punti di singolarità

Supponiamo che $f(x)$ continua in $[a, b]$

approssimo f con g



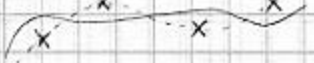
conosco poche informazioni

Come faccio a sapere di quanto mi discosto?

Approssimazione uniforme (o errore, norma)

$\|f - g\|_\infty = \max_{[a,b]} |f(x) - g(x)|$

Ho f cont: $\exists\ P_n$ t.c. $\|f - P_n\| \to \phi$?



WEIERSTRASS disse che si potevano trovare polinomi che al crescere di n si avvicinano al vero grafico della funzione.

Un esempio esplicativo: $\exists!\ E_n(f) = \min_{P \in \mathcal{P}_n} \|f - P\| \to$ polinomio di migliore approssimazione uniforme, ma non sappiamo costruirlo, lo sappiamo solo in teoria

(un unico)

— Errore quadratico medio

f quadrato integrabile

g $\int_a^b (f(x) - g(x))^2\,dx$ = errore quadratico medio

$w(x)$ → si può aggiungere questa funzione peso

$f_n(x) = \sum_{k=1}^{n} c_k \phi_n(x)$ → migliore g che possiamo scegliere

ϕ_n cont. quadrato integrabile, ortonormale

$\int_a^b w(x) (f(x) - f_n(x))^2\,dx$ minimo

$c_k = a_k = \int_a^b w(x) f(x) \phi_n(x)\,dx$

coeff. di Fourier

Risultato più che soddisfacente: metodo dei medi quadrati, con qualche riferimento al metodo uniforme.

Definizione di ortonormalità: $\int_a^b w(x) \phi_r(x) \phi_s(x)\,dx = \begin{cases} 0, & r \neq s \\ 1, & r = s \end{cases}$

14/12/2005

Approssimazione in media quadratica (non c'è su Manegatto)

$\sqrt{\int_a^b w(x) [f(x) - f_n(x)]^2\,dx}$ → trovare f_n tale che questa quantità sia minima

peso

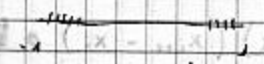
$w(x) = \frac{1}{\sqrt{1-x^2}}$, $[a, b] = [-1, 1]$

T_m pol. di Cebicev di prima specie

$T_0 = 1$, $T_1(x) = x$, $T_{m+1}(x) = 2x \cdot T_m(x) - T_{m-1}(x)$ → relazione di ricorrenza a 3 termini

zeri di T_m $(T_m = 0)$

$x_k = \cos \frac{\pi}{2m}(2k+1)$ $k = 0, \dots, m-1$ reali distinti e in $(-1, 1)$



più densi verso gli estremi

$$\int_{-1}^{1} \frac{T_m(x)\, T_n(x)}{\sqrt{1-x^2}}\, dx = \begin{cases} 0, & m \neq n \\ \frac{\pi}{2}, & m = n > 0 \\ \pi, & m = n = 0 \end{cases}$$

T_m ortogonale

$\left\{ \frac{T_0}{\sqrt{\pi}}, \frac{T_n}{\sqrt{\frac{\pi}{2}}} \right\}_n$ sistema ortonormale in $(-1, 1)$ $\xrightarrow{\text{perché}}$ $\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} \tilde{T}_n(x)\, \tilde{T}_m(x)\, dx = \begin{cases} 0, & m \neq n \\ 1, & m = n \end{cases}$

$f_m(x) = c_0 \frac{T_0}{\sqrt{\pi}} + \sum_{k=1}^{m} c_k T_k(x) \frac{\sqrt{2}}{\sqrt{\pi}}$ → realizza la minima approssimazione in media quadratica se:

$c_0 = a_0 = \frac{1}{\sqrt{\pi}} \int_{-1}^{1} f(x)\, T_0\, dx$

$c_k = a_k = \frac{\sqrt{2}}{\sqrt{\pi}} \int_{-1}^{1} f(x)\, T_k(x)\, dx$ → polinomio di quasi migliore approssimazione

Equazioni differenziali

lisate in Fisica, Ingegneria, economia, chimica. Studiano fenomeni evolutivi, per esempio come si propaga un suono, come si propaga il calore in una stanza. Sempre difficili da risolvere con l'analisi.

esempio

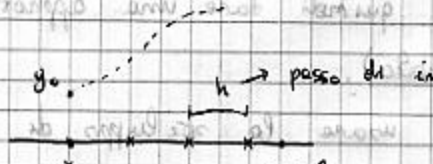
$$\begin{cases} y'(x) = y(x) \\ y(0) = 1 \end{cases}$$

$y(x)$? in $[0, l]$? → $y(x) = e^x$ (caso banale)

Cos'è una eq. diff del 1° ordine ?

$$\begin{cases} y'(x) = f(x, y(x)) \\ y(x_0) = y_0 \end{cases}$$ $[x_0, x_0 + l]$

↓ condizione iniziale



Vogliamo trovare la soluzione in generici punti equidistanti → $x_i = x_0 + i\,h$ modi

dove $h = \frac{l}{n}$

Metodo di Eulero

$y(x_{i+1}) - y(x_i) = y'(\xi)(x_{i+1} - x_i)$, ξ ?

faccio un' approssimazione grossolana → $y'(\xi) \cong y'(x_i)$

quindi:

$y(x_{i+1}) - y(x_i) = \underbrace{y'(x_i)}_{f(x_i, y(x_i))} \underbrace{(x_{i+1} - x_i)}_{h}$

per la relazione iniziale

$y(x_{i+1}) = y(x_i) + h f(x_i, y(x_i))$ → scrivendo $y(x_i) = y_i$, abbiamo →

→ $y_{i+1} = y_i + h f(x_i, y_i)$

Se poniamo $i = 0$ → $y_1 = y_0 + h f(x_0, y_0)$ → di questa conosciamo tutto: y_0 (cond. iniziale), h (fissata da noi), f (data).

Ma questa formula è affetta da un grosso errore dato da: $y'(\xi) \cong y'(x_i)$

Calcoliamola ora per $i = 1$:

$y_2 = y_1 + h f(x_1, y_1)$ → questa è due volte affetta da errore, perché uso qualcosa che era già affetto da errore (y_1).

Sempre peggio per $i = 2$, $i = 3$, $i = 4$ ...

Quindi questo metodo è molto semplice, ma ha la propagazione dell' errore (accumulazione dell' errore).

Ci sono metodi più accurati.

Metodo di Eulero - Cauchy

Basato su questa approssimazione grossolana → $y'(\xi) \cong \frac{y_{i+1} - y_{i-1}}{2h} = y'(x_i)$

↓ più accurata della precedente

$y_{i+1} = y_{i-1} + 2h\, y'(x_i) = y_{i-1} + 2h f(x_i, y_i)$

Calcoliamola per $i = 1$:

$y_2 = y_0 + 2h f(x_1, y_1)$ → y_0 lo conosciamo (cond. iniziale), ma y_1 ?

Quindi questo metodo di Eulero - Cauchy non è auto partente (ha bisogno di un valore ausiliario per innescare il processo).

Dobbiamo quindi fare una approssimazione per y_1 (non col metodo di Gauss, che non è accurato).

Si deve usare lo sviluppo di Taylor ricordando: $y'(x) = f(x, y(x))$.

Dobbiamo calcolare le derivate parziali → f_x, f_y.

Questi metodi sono superati, hanno + di 100 anni.

Metodi passo passo (Step by Step)

Classe di metodi più utile per risolvere le eq. diff:

$$y_{i+k} = \sum_{j=0}^{k-1} \{ a_j y_{i+j} + h\, b_j f(x_{i+j}, y_{i+j}) \} + h\, b_k f(x_{i+k}, y_{i+k})$$

$b_k = \emptyset$ esplicita (predictor) → dai valori precedenti riesco a prevedere il valore corrente

$b_k \neq \emptyset$ implicita (corrector)

Due famiglie di metodi.

Metodo di Milne (corrector)

$$y_{i+1} = y_{i-3} + \frac{4}{3} h (2y'_{i-2} - y'_{i-1} + 2y'_i)$$

$$y_{i+1} = y_{i-1} + \frac{h}{3} (y'_{i-1} + 4y'_i + y'_{i+1})$$

$$y(x)_{i+1} = f(x_{i+1}, y(x_{i+1}))$$

Non è autopartente.

Metodo di Runge-Kutta (esplicito)

$$y_{i+1} = y_i + \frac{1}{2} (K_1 + K_2) \leftarrow \text{formula}$$

$$K_1 = h f(x_i, y_i)$$

$$K_2 = h f(x_i + h, y_i + K_1)$$

prima volta che dentro la f non compaiono solo x_i, y_i

È autopartente.

MATEMATICA COMPUTAZIONALE

6/10/2005

Matematica computazionale in generale

Non dà risultati specifici ma modelli di calcolo.

La soluzione trovata sarà sempre approssimata, vincoli: accuratezza, velocità di calcolo, margine di errore.

Esempio

$y(q) = \text{sen} \log q$ equazione esplicita, dove l'errore può essere nullo.

$y = e^{y q^2} \text{sen} y q$ equazione implicita (q e y non sono separate). La risolviamo ma in modo approssimato (f nota non esplicitamente)

$y(q) = \sum_{k=0}^{\infty} a_k e^{-b_k q^2}$ anche se la f è nota esplicitamente, non possiamo calcolarla (a causa della somma degli ∞ termini)

$y(q) = \text{sen} q$ f nota esplicitamente e calcolabile non direttamente (ma anche in questo caso c'è un errore)

$y(q) = 3q$ f nota esplicitamente calcolabile direttamente (anche in questo caso abbiamo un errore, perchè non sappiamo come è q e come si può rappresentare con un numero finito di bit)

Quindi avremo sempre e comunque un margine di errore.

Il computer calcola il log con le serie.

Se dovessimo calcolare $\text{sen} \log \sqrt{q^{2+x}}$ bisognerebbe procedere dall'interno verso l'esterno, invece il computer ne crea un'espressione approssimata tipo $\sum a_n x^n$ ma estremamente più veloce da calcolare. Bisogna quindi sempre conoscere il margine di errore.

Esempio

Quanto occupa una foto da 1024 x 768 a colori?

$3 \cdot 1024 \cdot 768 (\cdot 1) = 2{,}4$ Mb

3 → RGB; valori da 255; perchè abbiamo supposto che un pixel occupa un byte di memoria

da 0 a 255 $256 = 2^8$

Se devo spedire la foto con un modem a 40 kbit/s?

$\frac{(2{,}4 \cdot 8) \cdot 1024}{40} = 500 \text{ sec} \simeq 8 \text{ min}$

ho convertito i byte in bit

Possiamo convertirla in Jpeg che si basa sulla tecnica lossy, basata sulla trasformata di Fourier.

20/10/2005

Matematica computazionale in grafica

Risoluzione in megapixel: quanti milioni di pixel.

Metodi per comprimere l'immagine → trattiamo l'immagine

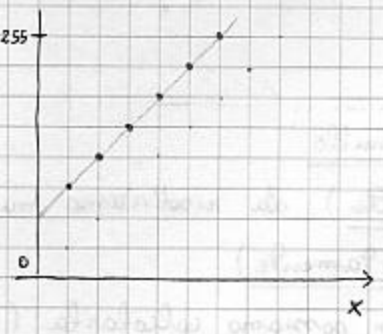
Non si perdono nel numero di pixel, ma nella qualità dell'immagine.

Per esempio si vedono dei blocchi nell'immagine.

Trasformata di Fourier

Algoritmo per comprimere basato sulla trasformata di Fourier.

Ammettiamo di aver immagine e di vedere solo una riga di shell. Esso corrisponde a vedere una riga di pixel (formato da 3 matrici R, G, B).



Un'immagine la cui componente rossa da sinistra a destra tende ad aumentare.

PIXEL	R
1	23
2	24
3	25
4	26
⋮	⋮

È sprecato usare memoria per dirgli tutti i pixel, quando in realtà è una retta.

$R = a + bx$

$R = 22 + x$ $\begin{cases} a = 22 \\ b = 1 \end{cases}$ lo possiamo fare con l'equazione, fornendo termine noto e coeff. angolare.

Numero di parametri da dare molto più basso.

Se noi zoomiamo continuamente vediamo i pixel che formano il pezzo di immagine, e non si può più andare oltre. Sappiamo le informazioni dei pixel, ma non tra pixel e pixel. Se usiamo le equazioni possiamo stimarlo, l'equazione è continua e può stimare anche tra pixel e pixel (rappresentazione vettoriale invece che discretizzata → stessa cosa che accade ai fonts true type).

Il problema è che le equazioni non sono così facili come quelle di una retta.

Per esempio: $R = a + bx + cx^2$ (parabola) → siamo andati a complicare, ma non è sufficiente).

Possiamo quindi incrementare l'ordine: $R = a + bx + cx^2 + dx^3 + ex^4 + \dots$

La compressione funziona facendo la rappresentazione tramite una funzione invece che pixel per pixel, cercando di usare il numero più basso di coefficienti.

Si possono usare per esempio degli esponenziali: $a + be^x + ce^{2x} + de^{3x} + ee^{4x} + \dots$

Non è questo il problema, ma quanti coefficienti uso, devo usarne il meno possibile.

Devo trovare una rappresentazione sparsa (con il più basso numero di coefficienti).

$f(x) = a_0 + a_1 \varphi_1(x) + a_2 \varphi_2(x) + a_3 \varphi_3(x) + \dots$ → combinazione lineare

come si scelgono $\varphi(i)$: primo criterio → sparsità

Ma noi dobbiamo sapere subito quanto valgono $a_0, a_1 \dots$ i coeff. Facciamo quindi il processo di ANALISI: noti $f(x_1), f(x_2) \dots \rightarrow a_0, a_1, a_2 \dots$ dobbiamo trovare subito i coeff. (in tempo reale → neanche 2 sec per avere il Jpeg), in una videocamera digitale devo fare 24 foto al secondo.

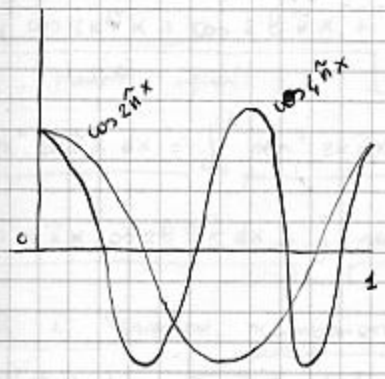
Per poi visualizzare l'immagine dobbiamo fare la sintesi: $a_0, a_1, a_2 \dots \rightarrow f(x_1), f(x_2) \dots$

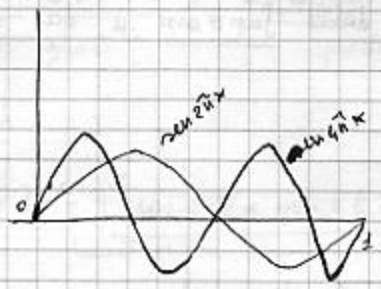
ma non è necessario che accada velocissima come l'analisi, ma deve essere comunque veloce.

la compressione deve dare un'immagine equivalente con una funzione → semplicità e velocità dei due algoritmi (ANALISI e SINTESI).

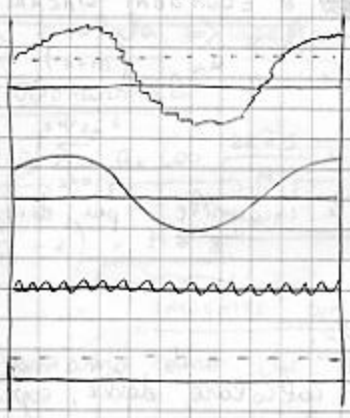
Sistema φ

$\varphi \equiv \{ \cos 2\tilde{\pi}x,\ \cos 4\tilde{\pi}x,\ \cos 6\tilde{\pi}x,\ \cos 8\tilde{\pi}x,\ \text{sen}\, 2\tilde{\pi}x,\ \text{sen}\, 4\tilde{\pi}x,\ \text{sen}\, 6\tilde{\pi}x \ldots \}$ $x \in [0,1]$





a che serve rappresentarle così?



→ se dovessi rappresentare questi dovrei usare pixel per ogni minima perturbazione ma mi accorgo che ora è l'unione di due funzioni + semplici

0,5

→ $\cos 2\tilde{\pi}x$ (1)

→ tanti $\text{sen}\, 40\tilde{\pi}x$ (0,1)

→ alza al livello del cos (0,5)

$f(x) = a_0 + a_1 \cos 2\tilde{\pi}x + b_{10} \text{sen}\, 20\tilde{\pi}x$

$a_0 = 0,5$

$a_1 = 1$

$b_{10} = 0,1$

l'equalizzatore dello stereo subisce lo stesso principio, onde alta e bassa frequenza, separazione dei segnali ad alta e bassa frequenza (ma non usa la trasformata di Fourier).

Sistema di Fourier

$f(x) = a_0 + \sum_{k=1}^{\infty} a_k \cos 2k\tilde{\pi}x + \sum_{k=1}^{\infty} b_k \text{sen}\, 2k\tilde{\pi}x, \quad x \in [0,1]$

$f(0) = a_0 + \sum_{k=1}^{\infty} a_k \cos 2k\tilde{\pi}\,\phi + \sum_{k=1}^{\infty} b_k \text{sen}\, 2k\tilde{\pi}\,\phi$

$= a_0 + \sum_{k=1}^{\infty} a_k$

UGUALE ALL'INIZIO

$f(1) = a_0 + \sum_{k=1}^{\infty} a_k$

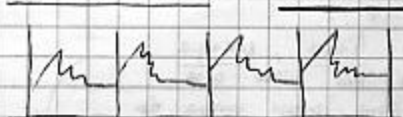
AGGIUNGONO UN GIRO CHE NON SERVE

$f(x+1) = a_0 + \sum_{k=1}^{\infty} a_k \cos 2k\tilde{\pi}(x+1) + \sum_{k=1}^{\infty} b_k \text{sen}\, 2k\tilde{\pi}(x+1)$

$= a_0 + \sum_{k=1}^{\infty} a_k \cos(2k\tilde{\pi}x + 2k\tilde{\pi}) + \sum_{k=1}^{\infty} b_k \text{sen}(2k\tilde{\pi}x + 2k\tilde{\pi})$

$= a_0 + \sum_{k=1}^{\infty} a_k \cos 2k\tilde{\pi}x + \sum_{k=1}^{\infty} b_k \text{sen}\, 2k\tilde{\pi}x$

$f(x+1) = f(x)$ PERIODICA



Per comprimere sostituiamo un sistema discreto (pixel) a un sistema funzionale (con pochi coeff.), ANALISI e SINTESI e viceversa molto rapidi. Fourier è in grado di soddisfare questi requisiti.

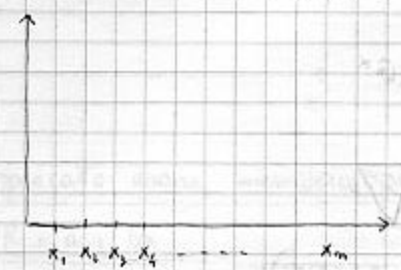
26/10/2005

Funzione di compressione

$f(x) = a_0 \varphi_0(x) + a_1 \varphi_1(x) + \dots + a_m \varphi_m(x) + \dots$

ANALISI → trovare i coefficienti (non immediato) $a_0, a_1, \dots a_m$

SINTESI → trovare $f(x)$ dalle funzioni e dai coefficienti



$$\left.\begin{aligned} f(x_0) &= a_0 \varphi_0(x_0) + a_1 \varphi_1(x_0) + \dots + a_m \varphi_m(x) \\ f(x_1) &= a_0 \varphi_0(x_1) + a_1 \varphi_1(x_1) + \dots + a_m \varphi_m(x_1) \\ &\vdots \\ f(x_m) &= a_0 \varphi_0(x_m) + a_1 \varphi_1(x_m) + \dots + a_m \varphi_m(x) \end{aligned}\right\}$$

SISTEMA DI EQUAZIONI LINEARI con $a_0, a_1, \dots, a_m$ $(m+1)$ incognite

Si risolve con Gauss in tempo $O(m^3)$. Più aumentano le incognite, più diventa complicato ed è difficile avere risultato in tempo reale.

Dobbiamo scegliere φ_m che permette di abbattere questo ostacolo.

Fourier permette di passare da $O(m^3) \to O(m^2)$, arriviamo a calcolare delle espressioncine. Però ancora non è sufficiente, per esempio per le foto.

Allora esiste la trasformata veloce di Fourier ($O(n \log_2 n)$ di complessità).

Ma ancora non è sufficiente per le operazioni in tempo reale.

La trasformata di Fourier viene implementata a livello hardware (su chip) alla velocità della luce. Prima era fatto via software ed era lento. Dal ponte un 84 86 fu fatto nell' hardware. Una tecnologia nuova (10 anni) permette di farlo in $O(N)$, (wavelet) → algoritmo Jpeg2000.

Fourier

$f(x) = a_0 + a_1 \cos 2\pi x + a_2 \cos 4\pi x + \dots + a_k \cos 2k\pi x + \dots + b_1 \operatorname{sen} 2\pi x + b_2 \operatorname{sen} 4\pi x + \dots + b_n \operatorname{sen} 2n\pi x + \dots$

$x \in [0, 1]$ → polinomio trigonometrico.

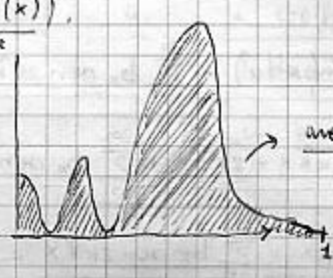
Rappresentiamo le frequenze oscillatorie delle funzioni, più sono lisce meglio è.

1) Si applica a funzioni periodiche. ($f(x+1) = f(x)$).

2) $\int_0^1 f^2(x)\,dx$

↓

in fisica, energia che può avere una funzione, oppure energia dei pixel nella macchina fotografica



→ area ha un valore finito

$$\int_0^1 f^2(x)\,dx = \sum_{k\geq 0} a_k^2 + b_k^2$$

finita — finita, quindi a_k e b_k devono finire, la luce della macchina fotografica non ci abbaglia

a_k e b_k devono essere $\emptyset$ o tendere a $\emptyset$, tali che li posso trascurare nella somma (ecco la compressione).

$$\int_0^1 f^2(x)\,dx = \int_0^1 \left[\sum_k a_k \cos 2k\tilde{\pi}x + b_k \operatorname{sen} 2k\tilde{\pi}x\right]^2 dx \approx \int_0^1 \left[\sum_k \left(a_k^2 \cos^2 2k\tilde{\pi}x + b_k^2 \operatorname{sen}^2 2k\tilde{\pi}x\right) + \sum_{\substack{k,\ell \\ k\neq\ell}} \cdot\right.$$

quadrati

$$\left. a_k a_\ell \cos 2k\tilde{\pi}x \cdot \cos 2\ell\tilde{\pi}x + \sum_{\substack{k,\ell \\ k\neq\ell}} b_k b_\ell \operatorname{sen} 2k\tilde{\pi}x \operatorname{sen} 2\ell\tilde{\pi}x + \sum_{k,\ell} a_k b_\ell \cos 2k\tilde{\pi}x \operatorname{sen} 2\ell\tilde{\pi}x\right]$$

prodotti uguali — prodotti uguali — prodotti misti

$$\int_0^1 \cos^2 2k\tilde{\pi}x\,dx = \int_0^1 \operatorname{sen}^2 2k\tilde{\pi}x = 1 \longrightarrow \text{REGOLA}$$

$$\int_0^1 \cos 2k\tilde{\pi}x \cos 2\ell\tilde{\pi}x\,dx = \int \operatorname{sen} 2k\tilde{\pi}x \operatorname{sen} 2\ell\tilde{\pi}x\,dx = \int_0^1 \cos 2k\tilde{\pi}x \operatorname{sen} 2\ell\tilde{\pi}x = \emptyset \longrightarrow \text{REGOLA}$$

quindi i termini scompaiono tutti (vale per ogni base ortogonale - ortonormale) e abbiamo:

$$\int_0^1 f^2(x)\,dx = \sum_{k\geq 0} a_k^2 + b_k^2$$

Energia limitata $\Rightarrow$ coeff. $\to \emptyset$

3) f. ortonormali

$$f\left(\frac{j}{N}\right) = \sum_{k=0}^{N/2-1} a_k \cos\frac{2k\tilde{\pi}j}{N} + \sum_{k=0}^{\frac{N}{2}-1} b_k \operatorname{sen}\frac{2k\tilde{\pi}j}{N}$$

$j = \emptyset, \dots, N-1$, $f(x_j) = \frac{j}{N}$

↓ è valore con N è uguale a quello con $\emptyset$ perciò non si mette

$2\left(\frac{N}{2}+1\right) = N+2$

INCOGNITE CON N EQUAZIONI (NON SI PUÒ FARE) PIÙ INCOGNITE PERFETTO ALLE EQ.

Però possiamo dire che $b_0 = \emptyset$, ed anche che $b_{\frac{N}{2}} = \emptyset$ quindi abbiamo tolto due incognite e possiamo risolvere il sistema.

$$\begin{cases} a_k = \frac{1}{N}\sum_{j}^{N-1} f_j \cos\frac{2kj\tilde{\pi}}{N} \\ b_k = \frac{1}{N}\sum_{j}^{N-1} f_j \operatorname{sen}\frac{2kj\tilde{\pi}}{N} \end{cases}$$

FOURIER IN FORMA ANALITICA

Quante operazioni sono necessarie per trovare a_k, b_k?

$N-1$ SOMMATORIE — N MOLT. DENTRO SOMMATORIA — 1 PRODOTTO FUORI

per un coefficiente → in totale $2N$.

$2N \cdot \frac{N}{2} = \frac{2N^2}{2} = O(2N^2) \to O(N^2)$.

↑ COEFF. — SOMMATORIA PRINCIPALE

$$\begin{cases} a_k = \frac{1}{N}\sum_{j=0}^{N/2} f_j \cos\frac{2kj\tilde{\pi}}{N} \\ b_k = \frac{1}{N}\sum_{j=0}^{N/2} f_j \operatorname{sen}\frac{2kj\tilde{\pi}}{N} \end{cases}$$

TRASFORMATA VELOCE DI FOURIER

Bisogna usare i complessi:

$$C_k = \frac{1}{N}\sum_{j=0}^{N-1} f_j e^{-i\frac{2\tilde{\pi}jk}{N}}$$

dove $e^{ix} = \cos x + i \operatorname{sen} x$

$(a+ib) \pm (c \pm id) = (a \pm c) + (b \pm d)\, i$ → REGOLA

$(a+ib) \cdot (c+id) = ac + iad + ibc - bd$ → REGOLA

$= (ac - bd) + (ad + bc)\, i$

$i^2 = -1$

$e^{-ix} = \cos(-x) + i\,(\operatorname{sen}(-x)) = \cos x - i \operatorname{sen} x$ → REGOLA

$\omega = e^{-i\frac{2\pi}{N}}$ → SE VALE QUESTA → $C_k = \frac{1}{N} \sum_{j=0}^{N-1} f_j\, \omega^{jk}$

$C_k = \frac{1}{N} \sum_{j=0}^{N-1} f_j\, e^{-i\frac{2\pi jk}{N}}$ ← PERCHÉ ERA COSÌ

$C_k^{(N)} = \frac{1}{N} \sum_{j=0}^{N-1} f_j\, \omega^{jk}$ → dividiamolo in due pezzi (numero di punti è una potenza di 2)

il conto viene fatto su N punti

$= \frac{1}{N} \sum_{j=0}^{\frac{N}{2}-1} \left[f_j\, \omega^{jk} + f_j + \frac{N}{2}\, \omega^{(j+\frac{N}{2})k} \right] =$

$= \frac{1}{N} \sum_{j=0}^{\frac{N}{2}-1} \left[f_j\, \omega^{jk} + f_{j+\frac{N}{2}}\, \omega^{jk} \cdot \omega^{\frac{N}{2}k} \right] =$

$\omega^{\frac{N}{2}k} = e^{-i\frac{2\pi}{N}\cdot\frac{N}{2}\cdot k} = e^{-i\pi k} = \cos(\pi k) - i \operatorname{sen}(\pi k) \Rightarrow$ (il seno si annulla)

$\begin{cases} k \text{ pari} \to 1 \\ k \text{ dispari} \to -1 \end{cases} \Rightarrow (-1)^k$

$= \frac{1}{N} \sum_{j=0}^{\frac{N}{2}-1} \left[f_j + f_{j+\frac{N}{2}} (-1)^k \right] \omega^{jk}$ quindi cambia a seconda che k sia pari o di spari

$\begin{cases} C_{2m_1}^{(N)} = \frac{1}{N} \sum_{j=0}^{\frac{N}{2}-1} \left[f_j + f_{j+\frac{N}{2}} \right] \omega^{2m_1 k} & \text{pari} \\ C_{2m_1+1}^{(N)} = \frac{1}{N} \sum_{j=0}^{\frac{N}{2}-1} \left[f_j - f_{j+\frac{N}{2}} \right] \omega^{j}\, \omega^{2m_1 j} & \text{dispari} \end{cases}$

$\begin{cases} C_{2m_1}^{(N)} = \frac{1}{2} \left[\frac{1}{N/2} \sum_{j=0}^{\frac{N}{2}-1} \left(f_j + f_{j+\frac{N}{2}} \right) \omega_1^{m_1 j} \right] \\ C_{2m_1+1}^{(N)} = \frac{1}{2} \left[\frac{1}{N/2} \sum_{j=0}^{\frac{N}{2}-1} \left(f_j - f_{j+\frac{N}{2}} \right) \omega^{j}\, \omega_1^{m_1 j} \right] \end{cases}$

3/11/2005

Fourier

$C_k = \frac{1}{N} \sum_{j=0}^{N-1} f_j\, \omega^{jk}$ $\quad k = 0, \dots, m-1$

$\omega := e^{-i\frac{2\pi}{N}}$

$f_j^{(1)} = f_j^{(0)} + f_{j+\frac{N}{2}}^{(0)}$ $\quad j = 0 \dots \frac{N}{2} - 1$

$f_{j+\frac{N}{2}}^{(1)} = \left(f_j^{(0)} - f_{j+\frac{N}{2}}^{(0)} \right) \omega^j$

$C_{2m_1} = \frac{1}{2} \left[\frac{1}{N/2} \sum_{j=0}^{\frac{N}{2}-1} f_j^{(1)}\, \omega_1^{m_1 j} \right]$

$C_{2m_1+1} = \frac{1}{2} \left[\frac{1}{N/2} \sum_{j=0}^{N/2-1} f_{j+\frac{N}{2}}^{(1)}\, \omega_1^{m_1 j} \right]$

(è la somma posta in due)

$m_1 = 0 \dots \frac{N}{2} - 1$

$\omega_1 = \omega^2 = e^{-\frac{i\,2\pi\cdot 2}{N}} = e^{-i\frac{2\pi}{N/2}}$

Tutti quei calcoli davano $2N^2$ → COMPLESSITÀ

Così non abbiamo dimostrato nulla.

Mi riduco da n calcoli a $\frac{N}{2}$ calcoli pari e $\frac{N}{2}$ calcoli dispari, ho guadagnato pochissimo $\frac{1}{4} N^2$ contro $2N^2$ di quella intera.

Le due somme si possono ancora dividere $\frac{N}{4} - 1$ $\quad \frac{N}{4} - 1$ $\quad \frac{N}{4} - 1$ $\quad \frac{N}{4} - 1$

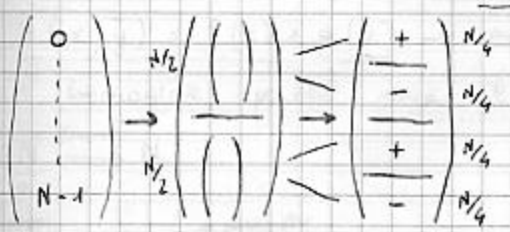
Queste si possono ancora dividere in due somme di $\frac{N}{8} - 1$ termini, e così via ...

In ogni passaggio salvo i coefficienti in un modo diverso.

Quante operazioni si vogliono per risolvere la trasformata?

Noi dobbiamo ridurci alla sommatoria di un solo termine, che significa il termine stesso.

Al primo passaggio faccio $\frac{3}{2}N$ operazioni.



Anche nel passaggio dopo dovremo fare $\frac{3}{2}N$ operazioni, consideriamo però l'altezza dell'albero che andiamo a creare, $\log_2 N \longrightarrow \frac{3}{2} N \cdot \log_2 N \longrightarrow O(N \log N)$.

Sembra che l'algoritmo sia nato nel 1965 da Cooley e Tukey, la trasformata era già usata ma con grossi dati c'erano problemi. Già ai tempi di Gauss questo algoritmo fu sviluppato.

Esempio

$$\begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \end{pmatrix} \rightarrow \begin{pmatrix} f_0 + f_4 \\ f_1 + f_5 \\ f_2 + f_6 \\ f_3 + f_7 \\ f_0 - f_4 \\ (f_1 - f_5)\,\omega \\ (f_2 - f_6)\,\omega^2 \\ (f_3 - f_7)\,\omega^2 \end{pmatrix} = \begin{pmatrix} f_0^{(1)} \\ f_1^{(1)} \\ f_2^{(1)} \\ f_3^{(1)} \\ f_4^{(1)} \\ f_5^{(1)} \\ f_6^{(1)} \\ f_7^{(1)} \end{pmatrix} \longrightarrow \begin{pmatrix} f_0^{(1)} + f_2^{(1)} \\ f_1^{(1)} + f_3^{(1)} \\ f_0^{(1)} - f_2^{(1)} \\ (f_1^{(1)} - f_3^{(1)})\,\omega_4^0 \\ f_4^{(1)} + f_6^{(1)} \\ f_5^{(1)} + f_7^{(1)} \\ f_4^{(1)} - f_6^{(1)} \\ (f_5^{(1)} - f_7^{(1)})\,\omega^2 \end{pmatrix} = \begin{pmatrix} f_0^{(2)} \\ f_1^{(2)} \\ f_2^{(2)} \\ f_3^{(2)} \\ f_4^{(2)} \\ f_5^{(2)} \\ f_6^{(2)} \\ f_7^{(2)} \end{pmatrix} \rightarrow$$

$$\rightarrow \begin{pmatrix} f_0^{(2)} + f_1^{(2)} \\ f_0^{(2)} - f_1^{(2)} \\ f_2^{(2)} + f_3^{(2)} \\ f_2^{(2)} - f_3^{(2)} \\ f_4^{(2)} + f_5^{(2)} \\ f_4^{(2)} - f_5^{(2)} \\ f_6^{(2)} + f_7^{(2)} \\ f_6^{(2)} - f_7^{(2)} \end{pmatrix} = \begin{pmatrix} f_0^{(3)} \\ f_1^{(3)} \\ f_2^{(3)} \\ f_3^{(3)} \\ f_4^{(3)} \\ f_5^{(3)} \\ f_6^{(3)} \\ f_7^{(3)} \end{pmatrix}$$

$\underbrace{\frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdots \frac{1}{2^{\log_2 N}}}_{N \text{ passi}} = \frac{1}{N}$

Possiamo anche calcolare prima i pari, poi i dispari:

$$\begin{matrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \end{matrix} \rightarrow \begin{matrix} C_0 \\ C_2 \\ C_4 \\ C_6 \\ \hline C_1 \\ C_3 \\ C_5 \\ C_7 \end{matrix} \rightarrow \begin{matrix} C_0 \\ C_4 \\ C_8 \\ \hline C_2 \\ C_6 \\ C_{10} \\ \hline C_1 \\ C_3 \\ C_5 \\ C_9 \end{matrix}$$

Quindi i coefficienti saranno fatti scambiati, per riconoscerli dobbiamo scrivere il codice binario dei coefficienti:

$$\begin{pmatrix} 0&0&0\\ 0&0&1\\ 0&1&0\\ 0&1&1\\ 1&0&0\\ 1&0&1\\ 1&1&0\\ 1&1&1 \end{pmatrix} \rightarrow \begin{pmatrix} 0&0&0\\ 1&0&0\\ 0&1&0\\ 1&1&0\\ 0&0&1\\ 1&0&1\\ 0&1&1\\ 1&1&1 \end{pmatrix} \rightarrow \begin{pmatrix} 0\\ 4\\ 2\\ 6\\ 1\\ 5\\ 3\\ 7 \end{pmatrix} \quad \begin{matrix} k=0\\ k=4\\ k=2\\ k=6\\ k=1\\ k=5\\ k=3\\ k=7 \end{matrix}$$

Questa è esattamente l'ANALISI di Fourier.

Dobbiamo fare la sintesi e trovare la funzione, esiste la formula inversa:

$$f_j = \sum_{k=0}^{N-1} c_k \, \omega^{-jk} \qquad \cos\frac{2jk\pi}{N} + i \operatorname{sen}\frac{2jk\pi}{N}$$

(differenza nel segno; complesso coniugato)

$FT(f)$ ← trasformata di Fourier

$IFT = FT(\bar{c})$ ← inversa della trasformata di Fourier; complesso coniugato di c

Per fare l'inversa non bisogna modificare l'algoritmo, ma fare il complesso coniugato.

$$f_j = \sum_{k=0}^{N-1} c_k \cdot \omega^{-jk} =$$

$$= \sum_{k=0}^{N-1} \left(c_k^R + i\,c_k^I\right)\left(\cos + i \operatorname{sen}\right) =$$

(R = reale, I = immaginario; $\cos\frac{2kj\pi}{N}$ stessa cosa col seno)

$$= \sum_{k=0}^{N-1} c_k^R \cos + i\,c_k^R \operatorname{sen} + i\,c_k^I \cos - c_k^I \operatorname{sen} =$$

$$= \sum_{k=0}^{N-1} \left(c_k^R \cos - c_k^I \operatorname{sen}\right) + i\left(c_k^R \operatorname{sen} + c_k^I \cos\right)$$

$$f_j = \sum_{k=0}^{N-1} \bar{c}_k \cdot \omega^{jk} =$$

$$= \sum_{k=0}^{N-1} \left(c_k^R - i\,c_k^I\right)\left(\cos - i \operatorname{sen}\right) =$$

$$= \sum_{k=0}^{N-1} c_k^R \cos - i\,c_k^R \operatorname{sen} - i\,c_k^I \cos - c_k^I \operatorname{sen} =$$

$$= \sum_{k=0}^{N-1} \left(c_k^R \cos - c_k^I \operatorname{sen}\right) - i\left(c_k^R \operatorname{sen} + c_k^I \cos\right)$$

→ con c normale e con $\bar{c}$ (complesso coniugato) la parte reale è uguale e cambia solo un segno nella parte immaginaria.

Anche se le immagini hanno valori reali, noi lavoriamo con i complessi che hanno parte reale (quella data) e parte immaginaria nulla. Quando facciamo la cosa inversa è un guaio.

10/11/2005

Fourier

Noi abbiamo fatto Fourier per una riga, la dobbiamo fare per un'immagine

(matrice), ad es: $f(x) = a_0 + a_1 \cos 2k\tilde{\pi}x + a_2 \cos 4k\tilde{\pi}x + \dots + b_1 \operatorname{sen} 2k\tilde{\pi}x + b_2 \operatorname{sen} 4\tilde{\pi}kx + \dots =$

$= \sum_i \alpha_i \varphi_i(x)$

$g(x) = c_0 + c_1 \cos 2k\tilde{\pi}y + c_2 \cos 4k\tilde{\pi}y + \dots + d_1 \operatorname{sen} 2k\tilde{\pi}y + d_2 \operatorname{sen} 4k\tilde{\pi}y + \dots = \sum_e \beta_e \varphi_e(y)$

$h(x,y) = f(x) + g(y)$ → non funziona se le uniamo così! Troppo semplice per l'immagine, perché mica il colore si modifica uniformemente!

$N \xrightarrow{\text{Fourier}} N$

↑ punti

$N^2 \longrightarrow 2N$ (componente x + componente y) ← parametri

↑ valori per ogni immagine (punti = N × N = parametri)

Noi vogliamo riassumere N^2 parametri in $2N$ (compressione); ma il modello così è troppo semplice.

Proviamo a fare: $h(x,y) = f(x) \odot g(y)$ → così avremo N^2 prodotti, ovvero:

(non proprio un per)

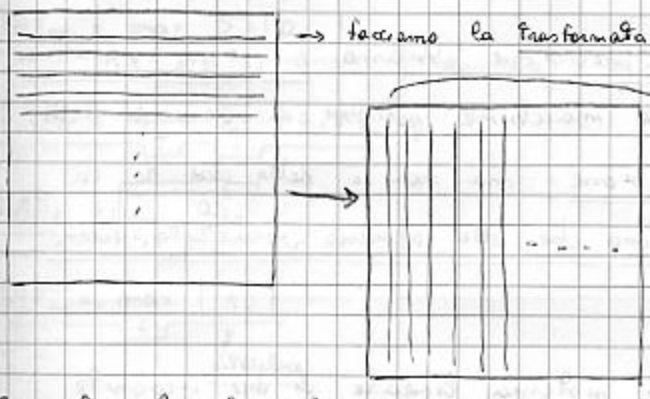
$h(x,y) = \sum_{k,e} \gamma_{k,e} \varphi_k(x) \varphi_e(y)$

coefficienti incognite (N^2) della rappresentazione funzionale

Ma questo non è il modello più generale, quello più generale è: $h(x,y) = \sum_{k,e} \xi_{k,e} \cdot \varphi_{k,e}(x,y)$ → in pratica facciamo → $\varphi_{k,e}(x,y) = \varphi_k(x) \cdot \varphi_e(y)$ (LI ABBIAMO SEPARATI)

cosa che non vale per esempio → $\operatorname{sen} 2k\tilde{\pi} \underbrace{xy}$

qua non si possono separare



→ facciamo la trasformata di Fourier per ogni riga e ne facciamo una nuova matrice

→ poi la facciamo per colonne e abbiamo finito

Per la trasformata inversa è la stessa cosa (prima per righe applichiamo l'inverse e poi per colonne)

Possiamo fare anche per colonne e per righe in tutti e due i casi.

Possiamo dividere Fourier anche non solo in 2, ma in 3 (resto 0, resto 1, resto 2), oppure per 4 (resto 0, 1, 2, 3) ecc..., quindi non è detto che i punti devono essere per forza una potenza di 2. L'efficienza però è migliore con una potenza di 2.

MALCONDIZIONAMENTO

$\begin{cases} x - y = 1 \\ x - 1{,}0001\, y = 0 \end{cases}$ $\begin{cases} x = 1{,}0001\, y \\ 0{,}0001\, y = 1 \end{cases}$ $\begin{cases} y = 10^5 \\ x = 100'001 \end{cases}$ $\begin{cases} x = 100'000 \\ y = 100'001 \end{cases}$

$Ax = b$

$$A = \begin{vmatrix} 1 & -1 \\ 1 & -1,00001 \end{vmatrix}$$

$$\begin{cases} x - y = 1 \\ x - 0,99999\,y = 0 \end{cases} \rightarrow \begin{cases} x = 0,9999\,y \\ y = x - 1 \end{cases} \rightarrow \begin{cases} x = 0,9999\,y \\ y = -1 + 0,9999\,y \end{cases}$$

$$\rightarrow \begin{cases} x = -9999 \\ y = -\dfrac{1}{0,0001} = -100'000 \end{cases}$$

$x = 99'999$

$y = -100'000$

$Ax = b$

$$A = \begin{vmatrix} 1 & -1 \\ 1 & -0,99999 \end{vmatrix}$$

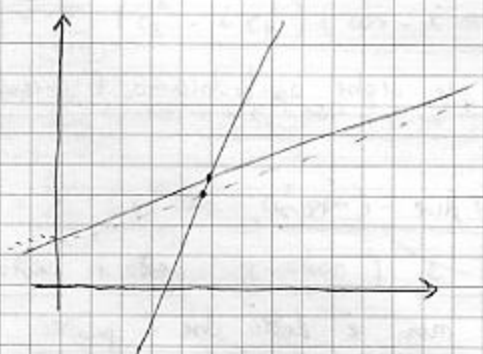
→ LE DUE MATRICI SONO VERAMENTE MOLTO SIMILI, MA IL PROBLEMA È CHE LE SOLUZIONI SONO COMPLETAMENTE DIVERSE.

Questo vuol dire che commettendo un piccolissimo errore abbiamo dei risultati diversissimi. Quindi non sappiamo se i nostri risultati siano veri e quanto lontani dalla verità.

32 bit singola precisione → errore $\sim 10^{-6}$

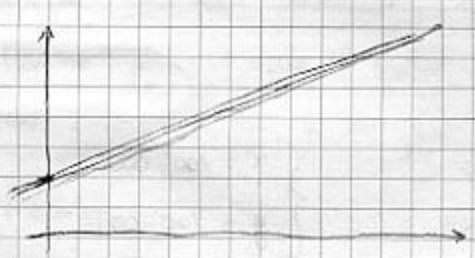
64 bit doppia precisione → errore $\sim 10^{-16}$

Se noi facciamo un programma in singola precisione abbiamo 6 cifre significative, ma se noi ci trasferiamo su altre macchine potrebbe accadere di tutto. Il problema si risolverebbe con la doppia precisione, ma non è detto che ho la soluzione giusta, quindi devo fare un programma che dia almeno risultato vicino alla realtà.



Nel caso di un sistema lineare di due incognite, dove troviamo l'intersezione delle due rette, se le spostiamo di poco il risultato non cambia molto.

Ma non era il nostro caso.



Le nostre rette erano quasi parallele:

$$\begin{pmatrix} 1 & -1 \\ 1 & -1,00001 \end{pmatrix} \qquad \begin{pmatrix} 1 & -1 \\ 1 & -0,99999 \end{pmatrix}$$

$$\begin{cases} x - y = 1 \\ x - 1.01y = 0 \end{cases}$$

Dobbiamo avere strumenti che ci dicono se una matrice è malcondizionata o ben condizionata.

Norma

Grandezza di uno scalare, valore assoluto. → $\|v\|$

Serve a misurare la grandezza del vettore.

$\|v\| > 0$ se $\underline{v} \neq \underline{0}$

$\|v\| = 0 \iff \underline{v} = \underline{0}$

$\|\alpha \cdot \underline{v}\| = |\alpha| \cdot \|v\|$

$\|\underline{u} + \underline{v}\| \leq \|u\| + \|v\|$

Come si calcola la norma di un vettore?

$$\|\underline{u}\|_\infty = \max_i |u_i|$$

$$\|\underline{u}\|_1 = \sum_i |u_i|$$

$$\|u\|_2 = \sqrt{\sum_i u_i^2}$$

le norme si definiscono anche per le matrici:

$\|\alpha \cdot A\| = |\alpha| \cdot \|A\|$

$\|A + B\| \leq \|A\| + \|B\|$

Come si calcolano le norme di matrici?

$$\|A\|_\infty = \max_i \sum_j |a_{ij}|$$

$$\|A\|_1 = \max_j \sum_i |a_{ij}|$$

(cambiano i e j ma max e Σ)

$$\|A\|_F = \sqrt{\sum_{i,j} a_{ij}^2}$$

$$\|A\|_2 = \max_i |\lambda_i|$$

(λ autovalore)

la norma mi dice quanto è grande la matrice, quanto questa matrice amplifica il vettore con cui dobbiamo fare l'operazione.

$Ax = b$ → diciamo che commettiamo un certo errore su esso

$A(x + \Delta x) = b + \Delta b$ → se la sottraggo a quella prima ho:

$A\Delta x = \Delta b$

allora è vero anche (su matrici compatibili) $\Delta x = A^{-1}\Delta b$ → questo mi dice quant'è l'errore che ho commesso

$\|\Delta x\| = \|A^{-1}\Delta b\| \leq \|A^{-1}\| \cdot \|\Delta b\|$

Ricordandomi, dico che $\|b\| = \|Ax\| \leq \|A\| \cdot \|x\|$

$\|\Delta x\| \leq \|A^{-1}\| \cdot \|\Delta b\|$

$\|A\| \cdot \|x\| \geq \|b\|$

$$\frac{\|\Delta x\|}{\|A\| \cdot \|x\|} \leq \frac{\|A^{-1}\| \cdot \|\Delta b\|}{\|b\|}$$

$$\frac{\|\Delta x\|}{\|x\|} \leq \|A\| \cdot \|A^{-1}\| \cdot \frac{\|\Delta b\|}{\|b\|}$$

errore relativo su x — $\gamma(A)$ — questo da' errore relativo su b

a noi interessa questa parte (INDICE DI CONDIZIONAMENTO)

$\gamma(A) = \|A\| \cdot \|A^{-1}\|$

$1 = \|I\| = \|A \cdot A^{-1}\| \leq \|A\| \cdot \|A^{-1}\|$ → norma della matrice identica

$\|A\| \cdot \|A^{-1}\| \geq 1$ → l'errore non si può ridurre, ma solo amplificare

$\frac{\|\Delta b\|}{\|b\|}$ → errore relativo rispetto a b, non da' altro che le sue cifre significative

ES: $\Delta b = 0{,}001$ $\quad b = 10{,}000000000\ldots$

Supponiamo di scrivere $\gamma(A) = 10^9$, la formula sarà:

$\frac{\|\Delta x\|}{\|x\|} \leq 10^9 \cdot \frac{\|\Delta b\|}{\|b\|}$ → P cifre significative

Delle P cifre significative non ce ne mangiamo 9.

Con la singola precisione si possono sfollare matrici malcondizionate fino a 10^6 cifre significative. Altrimenti dobbiamo passare alla doppia precisione.

Per vedere ciò basta una addizione (BEN CONDIZIONATA) e una sottrazione (MAL CONDIZIONATA).

Con l'addizione, anche se troncheremo la parte decimale, il risultato è simile, nella sottrazione il risultato ha meno cifre significative degli operatori!

Quindi un metodo è stabile se non propaga gli errori del calcolatore.

16/11/2005

Fourier

Si da' una rappresentazione funzionale, di una discretizzata.

$f(x) = a_0 + a_1 \cos 2\pi\tilde{\nu}x + a_2 \cos 4\pi\tilde{\nu}x + \ldots$

$f(x) = \sum_i a_i \varphi_i(x)$

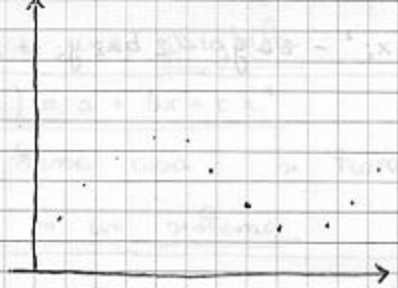
L'obiettivo dell'analisi di Fourier è trovare i coefficienti in un numero finito di passi.

$f(x_1) = a_0 \varphi_0(x_1) + a_1 \varphi_1(x_1) + a_2 \varphi_2(x_1) + \ldots$

$f(x_2) = a_0 \varphi_0(x_2) + a_1 \varphi_1(x_2) + a_2 \varphi_2(x_2) + \ldots$

$\vdots$

Questo sistema è risolvibile in modo analitico (più veloce).



→ conosco la funzione in un set di punti (dove posso passare infinite curve, non sappiamo cosa accade in mezzo - problema di interpolazione -).

Imposto che la funzione date io conosco i punti valga: $f(x_1)$, $f(x_2)$ ...
Cosa accade in mezzo dipende da cosa abbiamo scelto per φ_i.
Potrebbe essere una retta (quasi corretto se i punti sono molto fitti). Molto più probabilmente è una curva. Seno e coseno sono delle curve con infinite derivate, quindi sono molto usate. Le funzioni scelte non devono essere troppo lisce. (lisce: che non hanno bruschi cambiamenti).
In natura ci sono comportamenti lineari, quadratici (gravità), cubici...
Se una funzione può essere approssimata alla serie di Taylor, allora può essere ricondotta ad un polinomio, ma ci sono dei problemi quando il grado della funzione è grande: 1° è che nei punti che non sappiamo ha un comportamento oscillatorio.
Fare una funzione di grado molto alto è complicata e spesso non dà il risultato corretto, potrebbe bastarmi $f(x) = a_0 + a_1 x$ e mi servono due punti per trovarla; come lo scelgo? Usando il metodo dei minimi quadrati!

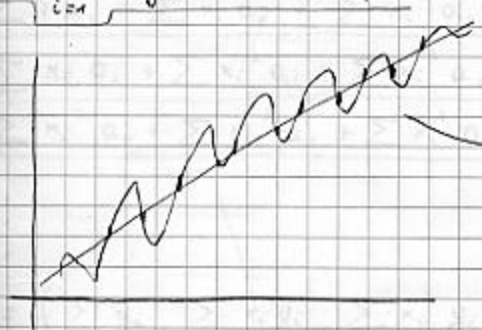
Minimi quadrati

$$S := \sum_{i=1}^{N} (y_i - a - b x_i)^2$$

definito come

$$y_i \equiv f(x_i)$$

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



→ si fa la somma dei quadrati delle parti ammesse

I parametri a e b migliori sono quelli per cui S è minimo.

Come facciamo a trovare il minimo di una funzione? Facciamo la derivata e vediamo dove sono i punti di massimo o minimo.
E a due variabili? Imponiamo che la derivata sia nulla con quei parametri:

$$\frac{\partial f(x,y)}{\partial x} = \phi$$

$$\frac{\partial f(x,y)}{\partial y} = \phi$$

DERIVATE PARZIALI → E POI DOBBIAMO CALCOLARE IL DET. HESSIANO:

$$\left| \frac{\partial^2 f(x,y)}{\partial x^2} \right| > 0$$

$S(a,b)$ → $\frac{\partial S}{\partial a} = \phi$ $\frac{\partial S}{\partial b} = \phi$ → imporre le derivate rispetto ad a e b uguali a ϕ

$$S(a,b) := \sum_{i=1}^{N} (y_i - a - b x_i)^2$$

$$S(a,b) := \sum_{i=1}^{N} (y_i - a - bx_i)^2 = \sum_{i=1}^{m} (y_i^2 + a^2 + b^2 x_i^2 - 2ay_i - 2bx_i y_i + 2abx_i)$$

$$\begin{cases} \frac{\partial S}{\partial a} = \sum_{i=1}^{n} 2a - 2y_i + 2bx_i = \phi \\ \frac{\partial S}{\partial b} = \sum_{i=1}^{n} 2bx_i^2 - 2x_i y_i + 2ax_i = \phi \end{cases}$$

$$\begin{cases} Na + \sum x_i b = \sum y_i \\ \sum_i x_i a + \sum x_i^2 b = \sum x_i y_i \end{cases}$$

$Ah = g$ $\qquad A := \begin{vmatrix} N & \sum x_i \\ \sum x_i & \sum x_i^2 \end{vmatrix}$; $\qquad g = \begin{vmatrix} \sum y_i \\ \sum x_i y_i \end{vmatrix}$

RISOLVIAMO IL SISTEMA; RISCRIVENDOLO COSÌ:

$$\begin{cases} S_0 a + S_1 b = T_0 \\ S_1 a + S_2 b = T_1 \end{cases}$$

dove $S_i = \sum_k x_k^i$ $\qquad T_i = \sum_k x_k^i y_k$

$$a = \frac{(T_0 - S_1 b)}{S_0}$$

$$b = \left(T_1 - \frac{S_1}{S_0}(T_0 - S_1 b)\right)\frac{1}{S_2} \Rightarrow$$

$$\Rightarrow b = \left(S_0 T_1 - S_1 T_0 + S_1^2 b\right)\frac{1}{S_2 S_0} \Rightarrow$$

$$\Rightarrow b\left(S_2 S_0 - S_1^2\right) = S_0 T_1 - S_1 T_0 \Rightarrow$$

$$\Rightarrow b = \frac{(S_0 T_1 - S_1 T_0)}{(S_2 S_0 - S_1^2)}$$

$$a = \left[T_0 - S_1 \frac{(S_0 T_1 - S_1 T_0)}{S_1 S_0 - S_1^2}\right]\frac{1}{S_0} \Rightarrow$$

$$\Rightarrow a = \frac{\left(S_2 S_0 - S_1^2\right) T_0 - S_1 (S_0 T_1 - S_1 T_0)}{S_0 (S_2 S_0 - S_1^2)} \Rightarrow$$

$$\Rightarrow a = \frac{S_2 T_0 - \cancel{S_1^2 T_0} - S_1 T_1 + \cancel{S_1^2 T_0}}{S_2 S_0 - S_1^2}$$

quindi: $\begin{cases} a = \frac{S_2 T_0 - S_1 T_1}{S_2 S_0 - S_1^2} \\ b = \frac{S_0 T_1 - S_1 T_0}{S_2 S_0 - S_1^2} \end{cases}$

Chi ci dice che è un max o un min? Dobbiamo andare a vedere l'hessiano:

$$\begin{vmatrix} \frac{\partial^2 S}{\partial a^2} & \frac{\partial^2 S}{\partial a \partial b} \\ \frac{\partial^2 S}{\partial b \partial a} & \frac{\partial^2 S}{\partial b^2} \end{vmatrix}$$ → o calcoliamo l'hessiano, o troviamo un metodo che ci dice che è il max.

$S = \sum_{i=1}^{N} (y_i - a - bx_i)^2$ → sceglieremo sempre a e b tale che $S \to \infty$.

Quindi la quantità non ha massimo, quindi è per forza un minimo.

Se noi abbiamo una parabola:

$f(a, b, c; x) = a + bx + cx^2$

si fa la stessa cosa, si trovano le tre incognite facendo le derivate parziali e mettendole in un sistema.

17/11/2005

Minimi quadrati

$$S_i := \sum_{i=1}^{N} \left(y_i - a_0 - a_1 x_i - a_2 x_i^2 - a_3 x_i^3 - \dots \right)^2$$

REGOLA

$$\frac{d\, g^2(x)}{dx} = 2 g(x) \cdot \frac{dg(x)}{dx}$$

$$\frac{\partial S}{\partial a_0} = -2 \sum_{i=1}^{N} \left(y_i - a_0 - a_1 x_i - a_2 x_i^2 - a_3 x_i^3 - \dots \right) =$$

$$\frac{\partial S}{\partial a_1} = -2 \sum_{i=1}^{N} \left(y_i - a_0 - a_1 x_i - a_2 x_i^2 - a_3 x_i^3 - \dots \right) x_i =$$

$$\frac{\partial S}{\partial a_2} = -2 \sum_{i=1}^{N} \left(y_i - a_0 - a_1 x_i - a_2 x_i^2 - a_3 x_i^3 - \dots \right) x_i^2 =$$

$$= -2 \sum_{i=1}^{N} y_i + 2 a_0 \sum 1 + 2 a_1 \sum x_i + 2 a_2 \sum x_i^2 + 2 a_3 \sum x_i^3 + \dots = \phi$$

$$= -2 \sum_{i=1}^{N} x_i y_i + 2 a_0 \sum x_i + 2 a_1 \sum x_i^2 + 2 a_2 \sum x_i^3 + 2 a_3 \sum x_i^4 + \dots = \phi$$

$$= -2 \sum_{i=1}^{N} x_i^2 y_i + 2 a_0 \sum x_i^2 + 2 a_1 \sum x_i^3 + 2 a_2 \sum x_i^4 + 2 a_3 \sum x_i^5 + \dots = \phi$$

Alla fine avremo:

$$\begin{cases} \sum 1\, a_0 + \sum x_i\, a_1 + \sum x_i^2\, a_2 + \sum x_i^3\, a_3 + \dots = \sum y_i \\ \sum x_i\, a_0 + \sum x_i^2\, a_1 + \sum x_i^3\, a_2 + \sum x_i^4\, a_3 + \dots = \sum x_i\, y_i \\ \sum x_i^2\, a_0 + \sum x_i^3\, a_1 + \sum x_i^4\, a_2 + \sum x_i^5\, a_3 + \dots = \sum x_i^2\, y_i \\ \vdots \end{cases}$$

ogni volta si comincia con un grado in più nella x

$$\underline{h} = \left(\sum y_i, \sum x_i y_i, \sum x_i^2 y_i, \dots \right)^T$$

$$\underline{\beta} \equiv \left(a_0, a_1, a_2, a_3, \dots \right)^T$$

Avremo una matrice:

$$A \cdot \underline{\beta} = \underline{h}$$

$$S_i = \sum_k x_k^i$$

$$T_i = \sum_k y_k x_k^i$$

$$\begin{vmatrix} S_0 & S_1 & S_2 & S_3 & \dots \\ S_1 & S_2 & S_3 & S_4 & \dots \\ S_2 & S_3 & S_4 & S_5 & \dots \\ S_3 & S_4 & S_5 & S_6 & \dots \\ \vdots & & & & \end{vmatrix}$$

→ ma questa matrice è tendenzialmente mal condizionata, e più mi va avanti col grado peggio è

Noi potremmo scegliere un polinomio di grado $n-1$, rispetto agli n punti che abbiamo, dimodochè i parametri saranno esattamente n: $a_0, a_1, a_2 \dots a_n$.

polinomio $N-1$

$$f(x) = a_0 + a_1 x + \dots + a_{n-1} x$$

$$\begin{cases} f(x_1) = y_1 = a_0 + a_1 x_1 + \dots + a_{n-1} x_1 \\ \vdots \\ f(x_n) = y_m = a_0 + a_1 x_m + \dots + a_{n-1} x_m \end{cases}$$

Riprendiamo: (tende al polinomio di interpolazione)

$$S = \sum \left(y_i - \sum_{j=0}^{n-1} a_j x_i^j \right)^2 \longrightarrow$$ se va a ϕ vuol dire che la soluzione coincide perfettamente con i punti dati

Quando la funzione oscilla molto, vuol dire che è malcondizionata.

Nel malcondizionamento anche un errore piccolissimo si propaga e porta a risultati assolutamente senza senso.

Malcondizionamento nella macchina

$a = m \cdot 10^{\text{mantissa}}$ $\quad 0 \le m < 1$ $\quad 0,\dots$

$327,4 \longrightarrow 0,3274 \cdot 10^3$

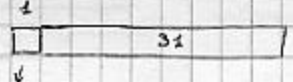
$0,0022 \longrightarrow 0,22 \cdot 10^{-2}$

Ma non esiste solo la base dieci.

$a = m \cdot B^s$ nel sistema di base B

Nella macchina si usano i transistor (ON / OFF) per rappresentare gli interi.

1	2	3 $\longrightarrow 2^n$
0	00	000
1	01	001
	10	010
	11	011
		100
		101
		110
		111

Si riservano 32 bit: 

si arriva a $+2^{32-1} - 1$

$2\,147\,483\,647$

Ma molti numeri non rientrano nel range.

$\longrightarrow$ altro metodo per rappresentarli

MANTISSA COMPRESA TRA 0,1 e 1

$$a = 2^s \left(x_1 2^{-1} + x_2 2^{-2} + x_3 2^{-3} + \dots + x_p 2^{-p} \right)$$

dove $|s| \le 2^{q-1}$

Quindi il numero più alto sarà: $2^{2^{q-1}}$ (quando tutti i bit della caratteristica sono pari a 1).

$\longrightarrow 2^{127} \simeq 10^{38}$

(in valore assoluto)

Qual è il numero più piccolo? → $s = -2^{q-1}$ e tutte le x sono a $\emptyset$.

$x_1 = x_2 = \dots = x_p = \emptyset$

Il numero più piccolo dopo $\emptyset$ è quello dove solo x_p è $= 1 \rightarrow 2^s \cdot 2^{-p}$

con $s = \emptyset$

$1 + \varepsilon = 1$

↑ perché se è molto piccolo (tipo 0,0000000100...) il calcolatore si perde un sacco di cifre

Io devo trovare l'ε più piccolo per cui $1 + \varepsilon > 1$ (zero macchina).

Esso è 2^{-p} (ovvero tutte le x a $\emptyset$ e x_p a 1).

In termini reali questo è $2^{-23} \simeq 10^{-7}$

$1 + \varepsilon = 1$

$\varepsilon < \varepsilon_0 \simeq 10^{-7}$

Ma 10^{38} non è ancora sufficiente.

Altro modo: → aggiunto 5 bit alla caratteristica

1	11	20

32 BIT $\quad 2^{1023} \simeq 10^{308}$ MAX

$2^{-20} \simeq 10^{-6}$ ZERO MACCHINA

Ma oggi si usa doppia precisione:

1	11	52

64 BIT $\quad 2^{1023} \simeq 10^{308}$ MAX

$2^{-52} \simeq 10^{-16}$ ZERO MACCHINA

Underflow: quando il numero è più piccolo del valore assoluto di 10^{-308}, la soluzione è che si mette il numero a $\emptyset$.

Per l'overflow la soluzione non è così semplice.

Consideriamo tre casi:

s	2^s	① $x_1 = 1, x_2 = \dots = x_{p-1} = \emptyset$ $\quad 2^s \cdot 2^{-1}$	② $x_1 = \dots = x_{p-1} = \emptyset$, $x_p = 1$ $\quad 2^s \cdot 2^{-p}$	③ $x_1 = \dots = x_p = 1$ $\quad 2^s \left(2^{-1} + 2^{-2} + \dots + 2^{-p}\right)$
0	1	2^{-1}	2^{-p}	$1 - 2^{-p}$
1	2	1	$2^{-(p-1)}$	$2 - 2^{-(p-1)}$
2	4	2	$2^{-(p-2)}$	$4 - 2^{-(p-2)}$
3	8	4	$2^{-(p-3)}$	$8 - 2^{-(p-3)}$

$2^s \cdot 2^{-p} \left(1 + 2 + \dots + 2^{p-1}\right) = 2^s 2^{-p} \left(2^p - 1\right) = 2^s \frac{2^p - 1}{2^p} = 2^s \left(1 - 2^{-p}\right)$

[0, 1] [0, 2] [0, 4] [0, 8]

② ÷ ③

① ÷ ③

0 1 2 3 4 5 6 7 8

Poiché il primo bit è sempre 1, non si rappresenta per niente nelle macchine, ma esso rimane virtualmente a disposizione della mantissa. (NOTAZIONE IMPLICITA 1)

Quindi i bit della mantissa passano da 52 a 53 e ho 2^{-53}.

Vediamo come gli errori si propagano:

$\sqrt{1+\delta} - 1 \qquad \delta < \varepsilon_0 \qquad \sqrt{1+\delta} - 1 = \emptyset$ → ma noi sappiamo che non è vero

$$\frac{\left(\sqrt{1+\delta} - 1\right)\left(\sqrt{1+\delta} + 1\right)}{\left(\sqrt{1+\delta} + 1\right)} = \frac{1+\delta-1}{\sqrt{1+\delta}+1} = \frac{\delta}{\sqrt{1+\delta}+1} \xrightarrow{\delta \to \emptyset} \frac{\delta}{1+1} = \frac{\delta}{2}$$

24/11/2005

Soluzione dei minimi quadrati

RUGBY2002@LIBERO.IT

Polinomio con grado basso.

$$S := \sum_{i=1}^{n} \left(y_i - f(\underline{a}; x_i)\right)^2 \qquad f(\underline{a}) := a_0 + a_1 x + \dots + a_m x^m$$

La matrice dovrà avere determinante maggiore di zero.

A volte capita che minimizzando una funzione non si riesce a trovare la soluzione, ma dobbiamo approssimare.

Newton e bisezione sono dei metodi iterativi dove lo spazio in cui si poteva trovare lo zero veniva sempre ridotto.

Col Newton si fa prima, bisogna trovare le derivate parziali.

$S = f(x; \Theta)$

$S = \sum_{i=1}^{n} \left(y_i - \log(a + x_i) + x_i\right)^2$

$y = \log(a + x) + x$

$$\frac{\partial S}{\partial a} = -2 \sum_{i=1}^{n} \left(y_i - \log(a + x_i) + x_i\right) \cdot \frac{1}{a + x_i} = \emptyset$$ → eq. non lineare

Ma non sappiamo se è un MAX o un MIN,

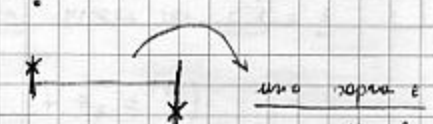
Potrebbe però essere un MIN locale.

Allora è meglio usare un metodo globale.

Un metodo ad hoc è:

abbiamo bisogno di tre punti in questo caso

al contrario del metodo a cui servono 2 punti



uno sopra e uno sotto lo zero

Piano piano si restringe lo spazio dove trovare i tre punti.

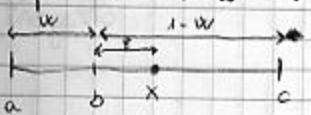
Non parliamo solo di minimizzazione di funzioni perchè la massimizzazione è la minimizzazione col segno meno.

Metodo della sezione aurea

Metodo lento ma non richiede l'uso delle derivate. È un metodo robusto (funziona abbastanza spesso). Più lento di quello della bisezione.

$f(b) < \{f(a), f(c)\}$ → condizione di partenza

Dopo mettiamo un punto x:

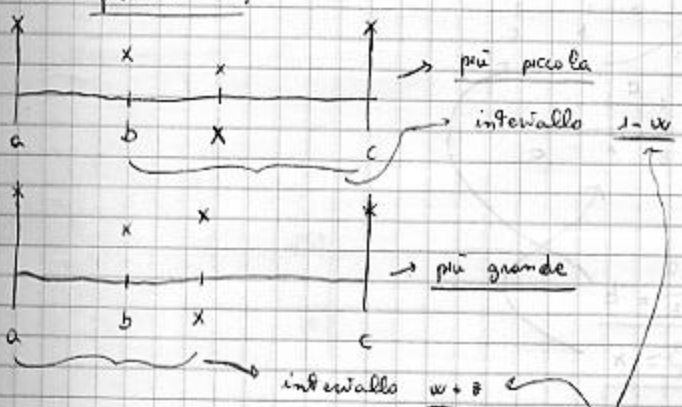


$w := \frac{b-a}{c-a}$ → dice quanto b sia vicino ad a

$z := \frac{x-b}{c-a}$

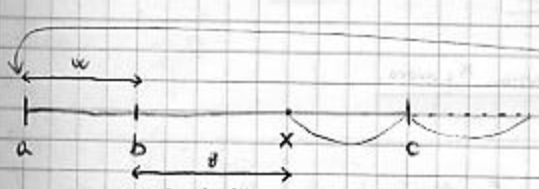
$1 - w = \frac{c-b}{c-a}$

Dobbiamo vedere se la funzione nel punto x è più piccola o più grande che nel punto b.



Dobbiamo cercare di evitare intervalli troppo piccoli, altrimenti rischiamo di garantire che dall'altra parte non ci siano dei minimi. Bisogna ridurre l'intervallo, ma con prudenza; quindi $1 - w = w + z \rightarrow z = 1 - 2w$

cosa vuol dire?



Ma quanto w chi ce lo ha detto?

Ammettiamo che sia stato calcolato come noi calcoliamo z.

distanza di b da a → w
distanza di x da b → $\frac{z}{1-w}$

} relative! → dobbiamo imporle uguali $w = \frac{z}{1-w} \Rightarrow$

$\Rightarrow \boxed{w = \frac{1-2w}{1-w}}$ → dobbiamo trovare quanto vale w

$w(1-w) = 1 - 2w \Rightarrow$

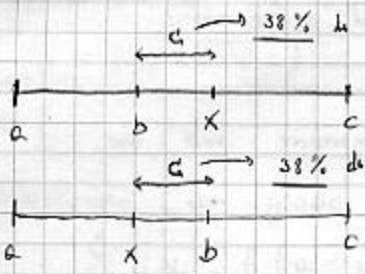
$w - w^2 = 1 - 2w \Rightarrow$

$w^2 - 2w - w + 1 = 0$

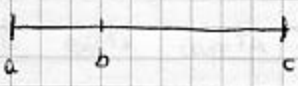
$w^2 - 3w + 1 = 0 \Rightarrow w = \frac{3 \pm \sqrt{9-4}}{2} = \frac{3 \pm 5}{2}$ → scartiamo la soluzione positiva perché supererebbe l'intervallo.

$w = \frac{3-\sqrt{5}}{2} = 0,38197\ldots = G$ → numero usalente a Pitagora

Nel grafico bc scegliamo x al 40% vicino a b
al 60% vicino a c

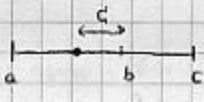


Algoritmo

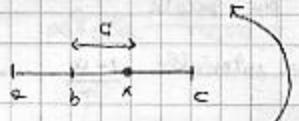


$f(b) < \{f(a), f(c)\}$ (min)

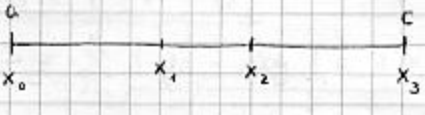
$(b-a) > c-b \qquad x = b - G(b-a)$



$(c-b) > (b-a) \qquad x = b + G(c-b)$



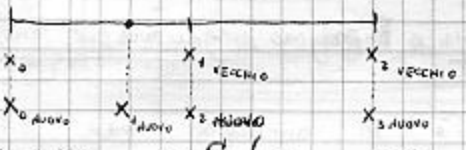
Usiamo 4 punti:



$x_1 = x$; $x_2 = b$

$x_1 = b$; $x_2 = x$

CASO $f(x_1) < f(x_2)$:



$x = x_{1\,\text{VECCHIO}} - G\left(x_{2\,\text{VECCHIO}} - x_{0\,\text{VECCHIO}}\right) = (1-G)\,x_{1\,\text{VECCHIO}} + G\,x_{0\,\text{VECCHIO}}$

Ora si fa la stessa cosa con l'intervallo $x_{0\,\text{NUOVO}}$ $x_{2\,\text{NUOVO}}$.

CASO $f(x_2) < f(x_1)$:

$x = x_{2\,\text{VECCHIO}} + G\left(x_{3\,\text{VECCHIO}} - x_{2\,\text{VECCHIO}}\right) = (1-G)\,x_{2\,\text{VECCHIO}} + G\,x_{3\,\text{VECCHIO}}$

Andiamo avanti con l'intervallo giusto.

- finora a, b, c

$x_0 = a$; $x_3 = c$

$b-a > c-b \Rightarrow x_1 = b - G(b-a)$; $x_2 = b$

$c-b > b-a \Rightarrow x_2 = b + G(c-b)$; $x_1 = b$

calcola f_0, f_1, f_2, f_3

loop fino a che $x_3 - x_0 > \varepsilon$

$f_1 < f_2 \Rightarrow x_1^{nuovo} = x_0$; $x_3^{nuovo} = x_2$; $x_2^n = x_1$; $x_1^n = G x_0 + (1-G) x_1$ → procedimento ricorsivo

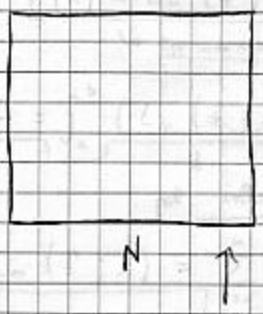
$f_2 < f_1 \Rightarrow x_0^n = x_1$; $x_3^n = x_3$; $x_1^n = x_2$; $x_2^n = (1-G) x_2 + G x_3$

calcola f_0, f_1, f_2, f_3

ridefinisci x_0, x_1, x_2, x_3

1/12/2005

Progetto



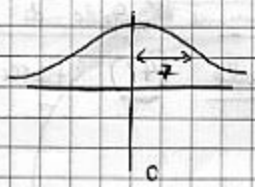
matrice

$i, j = 1 \ldots 256$ (immagini)

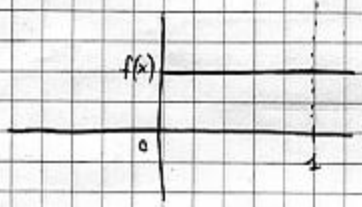
$I_{ij} \rightarrow I_{ij}^{\varepsilon} = I_{ij} + \varepsilon_{ij}$

livelli di grigio

generiamo numeri casuali (rand) e lo aggiungiamo al pixel



numero a caso estratto dalla distribuzione gaussiana



$0 \le \eta_{ij} \le 1$ $\quad -10 \le \varepsilon_{ij} \le 10$

$\varepsilon_{ij} = 20 (\eta_{ij} - 0,5)$

0 - 255 → toni di grigio

i num. casuali generati non potrebbero essere consentiti allora min di 0 → li mettiamo a $\emptyset$, mag. di 255 → li mettiamo a 255.

Arrotondiamo I_{ij} ottenuto all'intero più vicino → 7,4 ---> 7.

I_{ij} , I_{ij}^{ε}

FT ↓ ↓

C_{ij} , C_{ij}^{ε} → coefficienti dopo la trasformata

Fourier

Fourier toglie il disturbo dal segnale: si buttano tutti i coefficienti piccoli ($= \emptyset$) di $9 \cdot 10^{-3}$

$P_{ij}^{\varepsilon}(q)$ → rappresentiamo meglio i coefficienti ingrossati (dell'imm. originale)

coeff. dopo quelli più piccoli di q sono posti $= \emptyset$

si pongono uguali a $\emptyset$

$$P_{ij}^{\varepsilon}(q) := \begin{cases} C_{ij}^{\varepsilon}, & |C_{ij}^{\varepsilon}| > q \\ \emptyset, & |C_{ij}^{\varepsilon}| \not> q \end{cases}$$

$|a| := \sqrt{R_a^2(a) + I_m^2(a)}$

reale, immaginaria

$E(q) := \sum_{i,j=1}^{N} \left(P_{ij}^{\varepsilon}(q) - C_{ij} \right)^2$

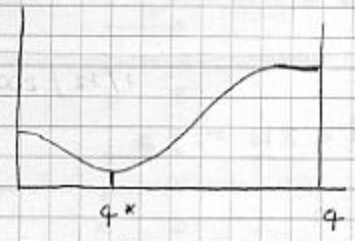
errore dipende da q; imm. con errore; imm. vera

Vediamo che succede per $E(0)$:

mi tengo tutti i coefficienti.

Se $E(\infty)$:

azzero tutto perché saranno tutti più piccoli. (unico colore)



errore rispetto a q

voglio q che rende più piccolo $E(q) := \sum_{i,j=0}^{n} \left(P_{ij}^{\epsilon}(q) - c_{ij} \right)^2$

devo trovare il minimo di questo

?

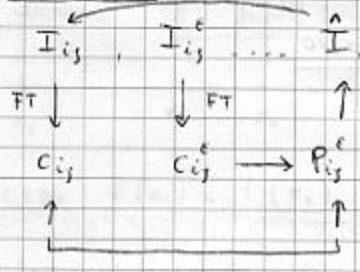
lo fa con l'algoritmo "regolabile" che nel Nonegato non c'è

si può usare un qualunque metodo di minimizzazione

Dopo che abbiamo ottenuto q ottimale (q^*) facciamo la trasformata inversa:

$P_{ij}^{\epsilon}(q) \rightarrow \hat{I}_{ij}^{\epsilon}$

RIASSUNTO:



Pacchetto: ImageJ → legge le imm già in formato binario (non lo usa nessuno!)

Java linguaggio più usato. Max 2 persone.

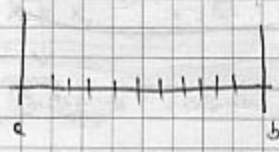
Formule di quadratura

Per approssimare gli integrali.

Non è sempre immediato risolvere l'integrale, come le derivate.

$I = \int_a^b f(x)\,dx$ $\qquad$ $I = I_n + E_n$

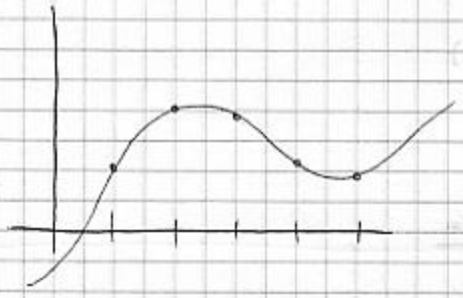
I_n: integrale approssimato; E_n: errore



$I_n = \sum_{k=1}^{n} w_k \cdot f(x_k)$

le formule di quadratura servono a scegliere i nodi e i pesi

$I = \sum_{k=1}^{n} w_k f(x_n) + E_n$



polinomio interpolante

$$f(x) \approx L_n(f;x)$$

↳ approssimo al pol. interpolante

$$I = \int_a^b (L_n(f;x) + R_n(f))\,dx$$

↓

pol. interpolante di Lagrange

$$L_n(f;x) = \sum_{k=1}^{n} f(x_k)\cdot \ell_{n,k}(x) \longrightarrow L_n(f;x)$$

$$L_n(f;x) = \sum \ell_{n,k}(x)\cdot f(x_k)$$

$$\ell_{n,k}(x) = \begin{cases} 1 & k = \ell \\ 0 & k \neq \ell \end{cases}$$

$$\ell_{n,1}(x) \to \ell_{n,1}(x_1) = 1,\ \ell_{n,1}(x_2) = 0 \ \dots\ \ell_{n,1}(x_n) = 0$$

$$\ell_{n,2}(x) \to \ell_{n,2}(x_1) = 0,\ \ell_{n,2}(x_2) = 1 \ \dots\ \ell_{n,2}(x_n) = 0$$

$$\ell_{n,n}(x) \to \ell_{n,n}(x_1) = 0,\ \ell_{n,n}(x_2) = 0 \ \dots\ \ell_{n,n}(x_n) = 1$$

$$L_n(f;x_\ell) = \sum \ell_{n,k}(x_\ell)\cdot f(x_k)$$

$$= f(x_\ell)$$

↓

verifica la proprietà di interpolazione

$$\ell_{n,k}(x) = \frac{(x-x_1)\dots(x-x_{k-1})\cdot(x-x_{k+1})\cdot\dots\cdot(x-x_n)}{(x_k-x_1)\cdot\dots\cdot(x_k-x_{k-1})\cdot(x_k-x_{k+1})\cdot\dots\cdot(x_k-x_n)}$$

↳ tutti tranne x_k

può valere 0 o 1

quindi torniamo all'integrale:

$$I = \int_a^b \left[\sum_{k=1}^{n} f(x_k)\cdot \ell_{n,k}(x) + R_n(f,x)\right]dx =$$

$$= \sum_{k=1}^{n} f(x_k)\int_a^b \ell_{n,k}(x)\,dx + \int_a^b \ell_{n,k}(x)\,dx + \int_a^b R_n(f;x)\,dx =$$

$\int_a^b \ell_{n,k}(x)\,dx$ = w_k perché 2 volte ?

$\int_a^b R_n(f;x)\,dx$ = E_n

$$= \sum_{k=1}^{n} w_k f(x_k) + E_n$$

w_k ↓ pari

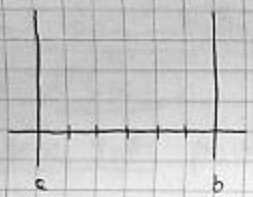
Una formula di quadratura (scelta di un insieme di nodi + pesi) ha un grado di precisione d dove:

$R_n = 0 \qquad n \le d$

Mettiamoci in una situazione in cui i nodi sono equidistanti:

$$I = \sum_{k=1}^{n} w_k f(x_k) + E_n \qquad w_k = \int_a^b \ell_{n,k}(x)\,dx$$

FORMULE DI NEWTON - COTES



NUMERI DI COTES $C_k = \frac{w_k}{(b-a)}$

$\sum_{k=1}^{n} C_k = 1$

$(b-a) = \int_a^b 1 \, dx \simeq \sum_{k=1}^{n} w_k$

$\lim_{n} \sum_{k=1}^{n} |C_k| = \infty$

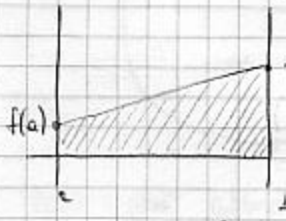
↓

dipende dall'interpolazione /\/\/\/\ (oscillante)

n L_{n-1}

Le formule di Cotes hanno senso per un numero di punti piccolo.

Esempio (2 punti)



posso solo tracciare una retta (area)

$I = \frac{1}{2}(f(a) + f(b)) \cdot (b-a)$

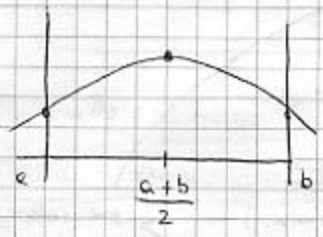
$I = \frac{h}{2}(f(a) + f(b))$ dove $h = (b-a)$

quindi $I = w_1 f(x_1) + w_2 f(x_2)$ $w_1, w_2 = \frac{h}{2}$

funziona malissimo perchè abbiamo approssimato a una retta

TRAPEZI $w_k = \left\{ \frac{h}{2}, \frac{h}{2} \right\}$

Esempio (3 punti)

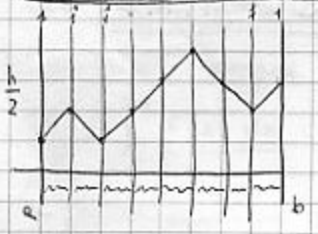


$I = \frac{h}{3}\left\{ f(a) + 4 f\left(\frac{a+b}{2}\right) + f(b) \right\}$ dove $h = \frac{b-a}{2}$

CAVALIERI - SIMPSON

$w_k = \left\{ \frac{h}{3}, \frac{4}{3}h, \frac{h}{3} \right\}$ dove $h = (b-a)/2$

Formule quadrative composte → utilizzando questi esempi si risolvono



$I = \sum_{i=1}^{m} I_i = \sum_{i=1}^{m} \frac{h}{2}\left(f(a + (i-1)h) + f(a + ih) \right)$

avremo le medie tra tutti i punti

$I = \frac{h}{2}\left[f(a) + 2 f(a+h) + 2 f(a+2h) + \dots + 2 f(a + (m-1)h \right]$

$= \frac{h}{2}\left[f(a) + 2\sum_{i=1}^{m+1} f(a + ih) + f(b) \right]$