Elementi di calcolo numerico

Rovesti Gabriel

Attenzione

Il file non ha alcuna pretesa di correttezza; di fatto, è una riscrittura attenta di appunti, slide, materiale sparso in rete, approfondimenti personali dettagliati al meglio delle mie capacità. Credo comunque che, per scopo didattico e di piacere di imparare (sì, io studio per quello e non solo per l'esame) questo file possa essere utile. Semplice si pone, per davvero ci prova.
Thank me sometimes, it won't kill you that much.

Gabriel

Sommario

Capitolo 1: Sistema floating-point e propagazione degli errori; costo computazionale

Lezione 1 e 2 - Numeri e floating point

Parliamo dei numeri con segno. In particolare per un numero si intende una parte con segno, una parte intera ed una parte frazionaria, solitamente realizzata con una codifica binaria.
Le cifre e gli indici corrispondono alle potenze, positive e negative.
Noi ci focalizzeremo sulla base *b=10,* nonostante siano rimaste altre basi come la base 60 (sessagesimale).
All'interno del calcolatore si usa invece la base 2, corrispondente ai 2 possibili stati assunti da una macchina.
In merito ad un numero sappiamo essere composto da una parte intera e una serie frazionaria, definita come serie convergente. Ciò è visibile come:

$$sign(x)\left(\underbrace{\sum_{j=0}^{m} c_j b^j}_{\text{parte intera}} + \underbrace{\sum_{j=1}^{\infty} c_{-j} b^{-j}}_{\text{parte frazionaria}}\right)$$

La serie della parte frazionaria (che sta tra 0 ed 1) è convergente e ciò si dimostra per confronto per serie a termini non negativi, da cui viene maggiorata.
Genericamente tutto si riduce alla serie geometrica di ragione $a = b^{-1}$.
In generale la successione delle somme parziali con a^{n+1} diverge con a > 1, mentre per a < 1 la serie della parte frazionaria è maggiorata da una serie geometrica convergente e quindi converge.

Per vedere che la parte frazionaria sta effettivamente tra 0 ed 1, ciò viene visto in base 10 si considera come esempio 0.99 periodico in base 10. Usando la serie geometrica e raccogliendo la cifra massima si dimostra che si ottiene proprio 1.
Alcune osservazioni:

- *i numeri irrazionali hanno parte frazionaria infinita in qualsiasi base*, dato che i numeri con parte frazionaria finita in una base sono necessariamente razionali, perché somma della parte intera che è numero naturale e di una combinazione lineare;
- i numeri razionali possono avere una base finita od infinita, per esempio:
  $\frac{1}{3}$ = $(0{,}100)_3$ (caso finito) Ma in base 10 diventa: $(0{,}333\ldots)_{10}$ = 1/3 (caso infinito)

Fondamentale nel calcolo numerico è il concetto di *errore* e in particolare il *troncamento* dei numeri, tagliando *n* cifre nella parte frazionaria ma mantenendo tutta la parte intera.
Introduciamo quindi due concetti chiave: *l'approssimazione* e l'errore stesso.

In merito al troncamento abbiamo:

$$\tilde{x}_n = sign(x)\left(\sum_{j=0}^{m} c_j b^j + \sum_{j=1}^{n} c_{-j} b^{-j}\right)$$

La quantità errore è il modulo della distanza tra due reali, *a* ed ˜*a*, (quindi una differenza, vista come errore assoluto). Quindi:

$$\text{ERRORE} = |a - \tilde{a}| = |\tilde{a} - a|$$

Vediamo di stimarla ed approssimarla correttamente.

Scritto da Gabriel

L'errore di troncamento non è altro che il resto della serie geometrica per avere una stima (usando i conti di $a = b^{-1}$) dell'effettivo errore di troncamento generabile dai calcoli. Si nota quindi che esso non supera b^{-n} , dove il b varia a seconda delle cifre utilizzate.
Attenzione: l'errore è stato solo stimato, che è una situazione tipica del calcolo numerico.
Quante cifre devo prendere dopo la virgola per non superare una certa soglia di tolleranza lo determina la risoluzione di una disuguaglianza logaritmica, che porta come risultato $n \geq -\frac{log(\varepsilon)}{log(b)}$
Il fatto di avere una stima permette di controllare l'errore, che dovrà necessariamente essere ≤ ε, che sarebbe la soglia di tolleranza.

La tecnica generalmente più usata è *l'arrotondamento*, che corrisponde alla serie infinita, considerando l'arrotondamento per *difetto* (quindi prima cifra va da 0 a 4 nella base 10) oppure per *eccesso* (da 5 a 9 base 10). Noi ci limitiamo a vedere basi pari. L'esempio è:

Si approssima

$$x = sign(x)\left(\sum_{j=0}^{m} c_j b^j + \sum_{j=1}^{\infty} c_{-j} b^{-j}\right)$$
$$= sign(x)\left(\sum_{j=0}^{m} c_j b^j + 0, c_{-1}c_{-2}\dots c_{-n}\dots\right)$$

con

$$\tilde{x}_n^{arr} = sign(x)\left(\sum_{j=0}^{m} c_j b^j + 0, c_{-1}c_{-2}\dots \tilde{c}_{-n}\right)$$

dove

$$\tilde{c}_{-n} = \begin{cases} c_{-n} & \text{se } c_{-(n+1)} < \frac{b}{2} \quad \longrightarrow DIFETTO \\ c_{-n}+1 & \text{se } c_{-(n+1)} \geq \frac{b}{2} \quad \longrightarrow ECCESSO \end{cases}$$

Il massimo errore nel caso di basi pari è metà del massimo errore di troncamento ad n cifre, quindi:

$$|x - \tilde{x}_n^{arr}| \leq \frac{b^{-n}}{2}$$

Ad esempio nel caso del numero 0,44:

- nel caso del troncamento tutti i numeri dell'intervallo [0.44 e 0.449) tendono a 0.45
- nel caso dell'arrotondamento approssima tutti i numeri dell'intervallo (0.435,0.445) seguendo le regole di prima dell'arrotondamento per eccesso/per difetto.

In generale è meglio l'arrotondamento perché considera un intervallo più ampio e in generale è effettivamente migliore.

Floating point

Un numero viene scritto come numero con segno, mantissa (vista come serie da 1 ad inf delle cifre) moltiplicato per una potenza della base.
L'esponente può essere positivo, nullo o negativo. Si adotta la convenzione che la prima cifra dopo la virgola *d1* sia diversa da 0 in maniera tale da non avere infinite rappresentazioni dello stesso numero.
Questa è la classica scrittura *floating point*, utile ad introdurre il concetto di *precisione macchina* ma anche di definizione dell'insieme dei *reali macchina*.
Per esempio:
$x = 1278.3405$ diventa $+0{,}12783405\ldots * 10^4$ oppure: $x = -0.0003267 = -(0.3267\ldots) * 10^{-3}$

Semplicemente si sposta la virgola con un opportuno spostamento della base.
La mantissa sta in 0 ed 1 (con 0 escluso), avendo in particolare la mantissa compressa tra 0.1 ad 1 (entrambi inclusi).

In generale un numero reale ha mantissa infinita, perché comprendiamo anche gli irrazionali; i numeri con mantissa finita sono invece i razionali (un esempio può essere 1/3 di prima con base 3 e base 10)

A questo punto si è in grado di definire *l'insieme dei reali-macchina* che, in un sistema di calcolo, lavora un insieme di numeri con una quantità finita di cifre di mantissa (definito come *d*) e un esponente che varia in un intervallo finito di numeri (*p* che dipende dalla sua base *b*). Il fatto che il numero di cifre di mantissa sia finito e l'intervallo di esponenti sia finito permette la memorizzazione utilizzando sequenze di bit in base 2.
A questa lezione segue modello di rappresentazione a 64 bit dei reali-macchina.
Ora vogliamo stimare l'errore in cui l'approssimazione avviene per arrotondamento della mantissa al numero di cifre disponibili. Quindi:

Definiamo arrotondamento a t cifre di un numero reale scritto in notazione floating-point

$$x = sign(x)(0, d_1 d_2 \dots d_t \dots) \cdot b^p$$

il numero

$$fl^t(x) = sign(x)(0, d_1 d_2 \dots \tilde{d}_t) \cdot b^p$$

dove la mantissa è stata arrotondata alla t-esima cifra

$$\tilde{d}_t = \begin{cases} d_t & \text{se } d_{(t+1)} < \frac{b}{2} \\ d_t + 1 & \text{se } d_{(t+1)} \geq \frac{b}{2} \end{cases}$$

Come detto, l'errore massimo sarà $\leq \frac{b^{-t}}{2}$, da cui sappiamo che non supera $\leq \frac{b^{p-t}}{2}$,
Subito si nota che l'errore dipende da *p*, cioè dall'ordine di grandezza del numero.
Numeri grandi in modulo avranno errori grandi, numeri piccoli in modulo avranno errori piccoli.

Allontanandosi oppure avvicinandosi agli estremi, è accettabile un errore variabile con l'ordine di grandezza del numero? Questo succede se ci spostiamo dall'*errore assoluto*, quindi dati due numeri *a* e *˜a* in modulo, *all'errore relativo*, definito invece come il rapporto dell'errore assoluto per il modulo di *a*. In generale l'errore relativo è il più importante in campo sperimentale e viene infatti espresso in percentuale.
Quindi:

$$|a - \tilde{a}| \quad \longrightarrow \quad \text{ERRORE ASSOLUTO}$$

$$\frac{|a - \tilde{a}|}{|a|}, a \neq 0 \quad \longrightarrow \quad \text{ERRORE RELATIVO}$$

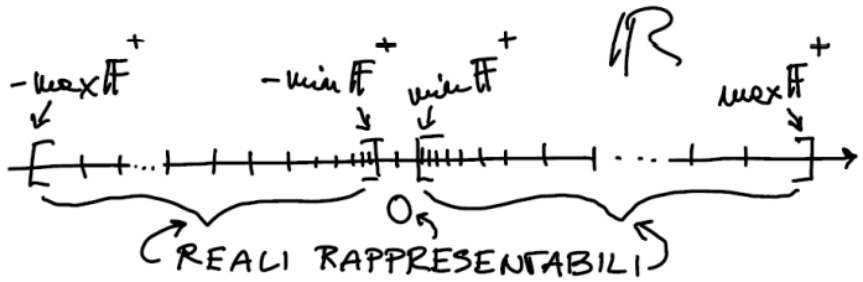
Per questo motivo andremo ad eseguire una stima da sotto, con $\frac{1}{|x|} \leq \frac{1}{|\alpha|}$
Di fatto viene poi eseguito un calcolo in cui, usando la tecnica di arrotondamento, mettiamo a frazione la stima da sopra e da sotto, si calcola il massimo errore relativo di arrotondamento possibile, cosiddetta precisione di macchina, che sarebbe $\frac{b^{1-t}}{2}$. Esso dipende solo da *b* e da *t.*
Un esempio del concetto del floating point sono *overflow* ed *underflow*, arrotondando a seconda della situazione a t cifre di mantissa. Nell'intervallo di rappresentazione ogni "tacchetta" nella rappresentazione è un intorno di precisione e, unendo tutti questi intorni, ogni numero se non è numero macchina viene arrotondato per difetto.

Lezione 3 - Struttura del sistema floating point e distribuzione

Trattiamo l'insieme dei reali macchina e la rappresentazione dei numeri, discutendo poi un modello di rappresentazione a 64 bit in MATLAB. In questo caso trattiamo una base *b* generica, prima di passare al modello a 64 bit. I *reali macchina* sono dei numeri con una quantità finita di cifre di mantissa e con un intervallo discreto di esponenti a disposizione che sposta la virgola a destra o a sinistra dando l'intervallo massimo e minimo dei numeri in modulo. Avendo $L < 0$ e $U > 0$ (estremi inferiore e superiore dell'intervallo), il numero rappresentato sarà nell'intervallo *L, L+1, … , -1, 0, 1, 2, …. U-1, U*.
Cercheremo quindi di capire:

- quanti sono i reali-macchina?
- quali reali sono approssimabili per arrotondamento con i reali macchina?
- come sono distribuiti nell'asse reale i reali-macchina?

In modo molto schematico, sintetizziamo con un disegno



I reali macchina sono l'insieme finito di tacche sull'asse reale, considerando lo 0, anch'esso reale macchina, i numeri stanno *nell'unione di due intervalli simmetrici*.
I due intervalli sono in particolare:

$$\mathbb{F}^+ \subset [\min \mathbb{F}^+, \max \mathbb{F}^+]$$

$$\mathbb{F}^- \subset [-\max \mathbb{F}^+, -\min \mathbb{F}^+]$$

Questi intervalli nel continuo sono i reali approssimabili tramite i reali-macchina, sapendo che l'errore relativo sarà <= a $b^{(1-t)}/2$. I numeri al di fuori di questi due intervalli sono troppo grandi oppure troppo piccoli per poter essere rappresentati e parliamo di overflow (intorno di ∞) ed underflow (intorno di 0).
In questi casi il numero rappresentato è fuori dall'intervallo rappresentabile e, nei vecchi linguaggi, questi errori potevano determinare l'arresto dello stato di calcolo del programma, dando degli errori oppure visualizzando *Inf* (*infinito)* oppure *NaN (Not a Number)*.
La struttura di un reale macchina è data da μ = segno * mantissa * b^p

Per cercare di prevenire underflow/overflow, conviene calcolare min F^+ e max F^+.
Il *minimo numero positivo* si otterrà *moltiplicando* la *minima mantissa* per la *potenza della base con il minimo esponente*, quindi:

$$\underline{\text{mantissa minima}} = (0, 10\ldots0)_b = 1 \cdot b^{-1}$$

$$\underline{\text{esponente minimo}} = L$$

E quindi: $$\min \mathbb{F}^+ = b^{L-1}$$

Invece il *massimo numero positivo* è ottenuto *moltiplicando* la *mantissa massima* per la *potenza della base col massimo esponente*, quindi:

$$\underline{\text{mantissa massima}} = (0, \underbrace{b-1}\,\underbrace{b-1}\ldots\underbrace{b-1})_b$$

avendo le t cifre uguali alla cifra massima, cioè b - 1.
Andiamo a calcolare la serie geometrica di ragione $b^{(-1)}$:

$$\underline{\text{mantissa max}} = \sum_{j=1}^{t} (b-1)\, b^{-j}$$ e con un po' di passaggi si ottiene: $$= 1 - b^{-t}$$

A questo punto, avremo che l'esponente massimo e l'estremo vero e proprio sono dati da (qui a destra):

$$\underline{\text{esponente massimo}} = U$$

Ad esempio, in base *10* con *16* cifre di mantissa, avendo esponenti minimo $L = -307$ e massimo
$U = +308$, avremo quindi intervalli con estremi grandissimi e piccolissimi:

$$\boxed{\max \mathbb{F}^+ = (1 - b^{-t})\, b^U}$$

$$\min \mathbb{F}^+ = 10^{-1-307} = 10^{-308}$$

$$\max \mathbb{F}^+ = (1 - 10^{-16}) \cdot 10^{308} \approx 10^{308}$$

Importante ricordarsi che la rappresentazione avviene per arrotondamento e gli unici numeri rappresentabili dal calcolatore sono proprio i *reali macchina* (numero di tacchette intuitivamente). La *cardinalità* (quindi il numero degli elementi) è data dal numero delle possibili mantisse moltiplicato per il numero dei possibili esponenti. Quindi:

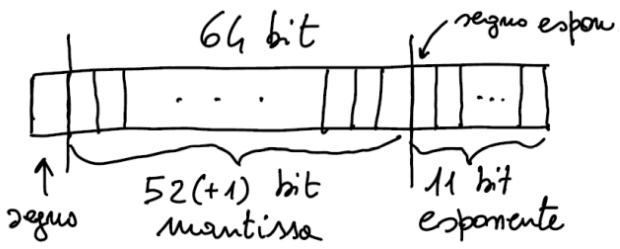
$$\text{card}(\mathbb{F}^+) = \text{numero possibili mantisse} \cdot \text{numero possibili esponenti}$$

che sarà uguale a:

$$(b-1) \cdot b^{t-1} \cdot (U - L + 1)$$

avendo *b* scelte per ogni cifra e *b-1* scelte per la prima cifra di mantissa. Andando fuori dagli intervalli detti siamo fuori dall'intervallo di rappresentazione.

Nel modello simil-Matlab di prima si ottiene una cardinalità data da un numero nell'ordine di $10^{(19)}$.
Nel caso del modello simil-Matlab, in quanto i numeri rappresentati non sono in base 2. Si sfrutta lo standard IEEE dedicando ai numeri una sequenza di 64 bit.
Qui avremo 1 bit riservato al segno, 52 (+1, quindi come fossero 53, perché la prima cifra di mantissa deve essere non nulla) bit alla mantissa ed 11 bit dedicati all'esponente.
Graficamente questa è la situazione:



In ciascuna delle caselle possiamo rappresentare gli stati dei bit 0/1 (i segni sono dati da "+" o "-" scegliendo uno dei due) e la precisione di macchina con un ordine di grandezza espresso da $10^{(-16)}$, in cui gli errori saranno $\leq$ a questa quantità. La precisione macchina è data da:

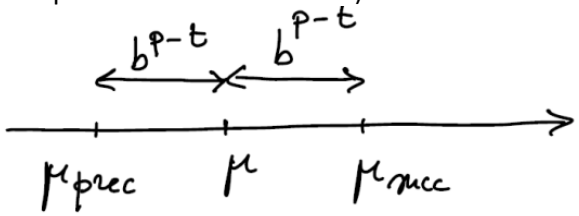
$$\varepsilon_M = \frac{2^{1-53}}{2} = 2^{-53} \approx 10^{-16}$$

Per il range degli esponenti, invece, avremo che il numero massimo (immagine sotto) e relativi estremi (immagine successiva) sono:

$$\max |p| = \sum_{j=0}^{9} 2^j = \frac{2^{10}-1}{2-1} = 1023$$

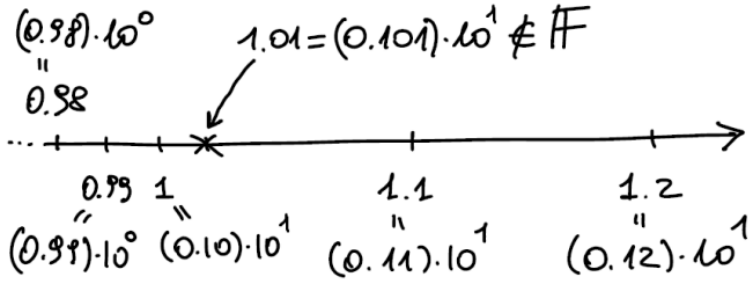
$$\max \mathbb{F}^+ \approx 2^{1023} \approx 10^{308}$$
$$\min \mathbb{F}^+ = 2^{-1024} \approx 10^{-308}$$

Attenzione che potremmo avere comportamenti strani/imprevisti; questo perché stiamo lavorando in base 2, mentre appunto stiamo rappresentando tutto in base 10. L'interfaccia dei numeri è data da
F (10, 16, -307, 308), avendo l'ordine di grandezza dei parametri chiave corretto.
Resta da analizzare come sono distribuiti i reali macchina, sapendo che questi numeri sono a *densità variabile* e *non sono distribuiti uniformemente*, avendo i numeri piccoli più vicini, mentre i numeri grandi più distanti tra di loro (si vede quindi che la densità varia).



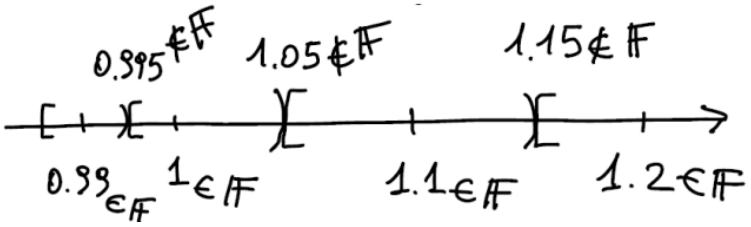
Questo si può capire calcolando la distanza tra reali-macchina consecutivi (con lo stesso esponente). La distanza è $b\hat{}(p-t)$ e i numeri differiscono del minimo possibile, in particolar modo di 1 nella t-esima cifra di mantissa. Questa distanza è assoluta; a noi interessa invece una distanza relativa, sapendo che varia (grazie a *p*, quindi l'ordine di grandezza del numero in base *b*) quando si passa per una potenza della base.

Esempio. $b = 10$ *e* $t = 2$ *(due cifre di mantissa)*

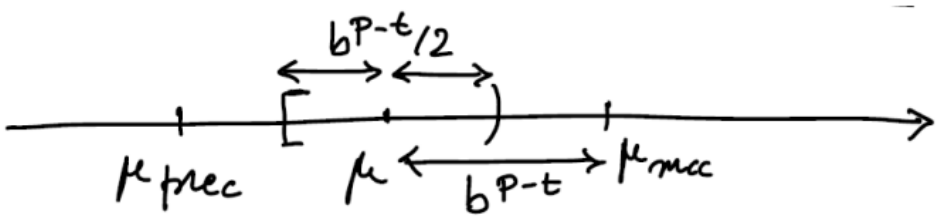


Chi è il reale-macchina successivo ad 1? Pensando a due cifre decimali, si potrebbe rispondere 1.01. In realtà non va bene, perché 1.01 ha 3 cifre di mantissa e in questo esempio t=2.
Il reale macchina successivo ad 1 è in realtà *1.1*. Arrotondando i numeri da 0.99, 1, 1.1, avremo graficamente:

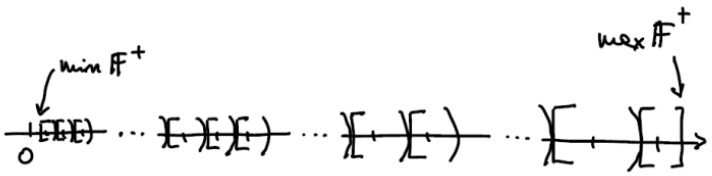
Pensando invece a quali reali sono approssimati da 0,99, 1 e 1,1 abbiamo graficamente



I reali macchina hanno come centro un intorno di approssimazione a *t* cifre di mantissa, avendo un errore assoluto <= $b^{(p-t)}/2$.

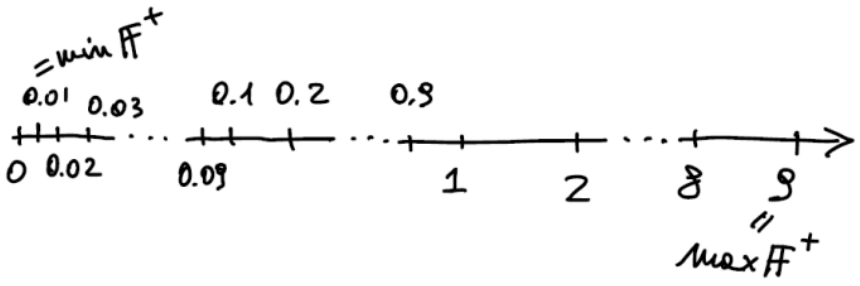


I numeri vengono arrotondati per eccesso o per difetto da una parte o dall'altra. Passando per le potenze della base gli intorni diventano asimmetrici.
L'intorno è asimmetrico e, a sx o a dx, si allunga a seconda di un fattore *b*. Tutti i reali di questi intorni vengono approssimati dal reale-macchina centro dell'intorno per arrotondamento a *t* cifre di mantissa. Nel caso di errore relativo sappiamo che non supererà la precisione di macchina ε, sapendo che è indipendente dal parametro *p*.
L'unione di tutti questi intervalli che coprono il continuo degli intervalli di rappresentazione per arrotondamento (partendo dal discreto) è il seguente:



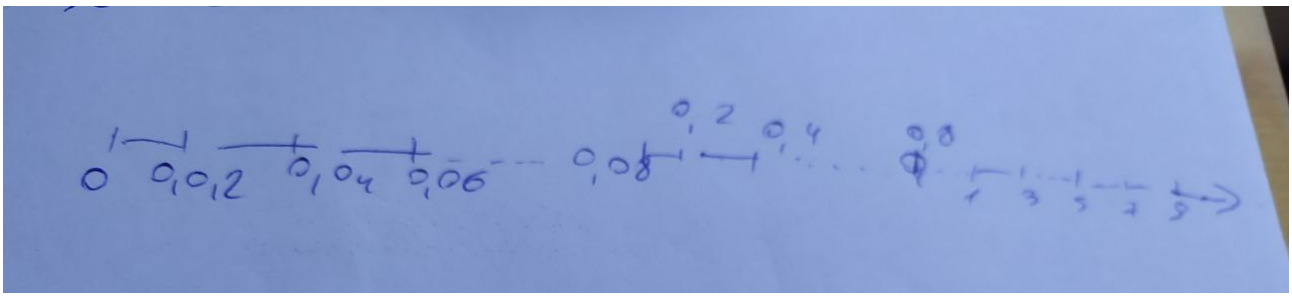
Abbiamo quindi due esempi conclusivi:

Disegnare $\mathbb{F}(10, 1, -1, 1)$
Abbiamo una sola cifra di mantissa (il minimo possibile) ed esponenti $p = -1, 0, 1$.
Disegniamo $\mathbb{F}^+$ (scrivendo per semplicità i reali-macchina in notazione standard)



Avendo una sola cifra, l'intervallo rappresentato è molto limitato e "povero", dove la precisione macchina arriva al massimo fino al 50% e si ha un range di esponenti molto piccolo, che parte dai centesimi.

Un esempio più preciso può essere l'intervallo **F** (10, 2, -2, 2) per esercizio (facendo opportuni zoom sui vari ordini di grandezza). Riporto l'esempio molto semplice (nella parte finale dell'immagine avanza appunto di 2 partendo da 1, nel caso non si leggesse):



Questo è un intervallo più ricco del precedente, calcolando i giusti valori di max e di min di errori e di arrotondamento vero.
In questo caso avendo due cifre di mantissa, partendo dai millesimi, avrò i decimillesimi, poi ai centesimi coi millesimi, i decimi coi centesimi, ecc.

Questo esempio dovrebbe far capire la ricchezza del sistema floating-point a 64 bit (precisione doppia). Analogamente esiste la precisione singola (32 bit) o precisione quadrupla (128 bit).
La precisione viene scelta dal linguaggio, permettendo di scegliere tra singola, doppia, quadrupla e solitamente vengono implementate a livello hardware (circuiti elettronici). Eventualmente è possibile aumentare il livello di precisione, implementando l'operazione aritmetica non più a livello hardware ma software, usando un algoritmo e quindi cresce il costo dell'operazione (risultando pesante in caso di grandi quantità di dati).
Tendenzialmente l'errore di misura è molto più "realistico", arrivando a $10^{(-4)}$, $10^{(-3)}$o misure simili; chiaramente $10^{(-16)}$ è un esempio ideale. Operazioni come la sottrazione può far perdere gran parte della precisione, unita ad altre operazioni di calcolo eseguite. Tutto questo studio conduce alla ***stabilità*** degli algoritmi numerici, considerando appunto la propagazione di errori in algoritmi instabili.

Lezione 4 - Operazioni aritmetiche con numeri approssimati

Tutti i calcoli e le varie operazioni sono sempre affetti da errori. Dato un insieme di reali-macchina il modo in cui il processore realizza l'operazione tra due reali rappresentabili x, y è indicata come si vede sotto (caso moltiplicazione tra due reali rappresentabili *x* ed *y*):

$$x \circledast y = fl^t(fl^t(x) \bigstar fl^t(y))$$

In generale per le ***operazioni macchina*** si ha l'arrotondamento dei due reali, viene eseguita l'operazione tra gli arrotondamenti e il risultato viene a sua volta arrotondato. In questi casi però non sono più valide la proprietà associativa e distributiva.
Le operazioni macchina quindi sono così esprimibili:

$$\bigstar = \begin{cases} \pm & \text{addizione, sottrazione} \\ * & \text{moltiplicazione} \\ / & \text{divisione} \end{cases}$$

L'esempio viene fatto con la moltiplicazione con **F** (10, 16, -307, 308) in cui poniamo:

$$a = 10^{200}\,,\ b = 10^{150}\,,\ c = 10^{-50}$$

Con le limitazioni degli esponenti reali si ha un overflow e:

Scritto da Gabriel

$$(a \otimes b) \otimes c = \text{overflow}$$

perché

$$a \otimes b = 10^{350} \quad \text{e} \quad 350 > 308$$

invece

$$a \otimes (b \otimes c) = 10^{200} \otimes 10^{100} = 10^{300}$$

Quindi

$$(a \otimes b) \otimes c \underbrace{\neq}_{\text{NO ASSOCIATIVA}} a \otimes (b \otimes c) = a \times b \times c$$

Infatti si verifica che la proprietà associativa (anche la distributiva) e similmente la commutativa possono generare errori di arrotondamento (dati dal rimescolamento degli operandi).
Sempre con il sistema **F** (10, 16, -307, 308) si esegue:

$$1 \oplus 10^{-16} = 1$$

Non esiste più un unico elemento neutro nell'addizione, avendo 17 cifre significative ed eseguendo l'arrotondamento, che porta quindi ad 1. Nel caso invece:

$$1 \oplus 10^{-15} = 1 + 10^{-15}$$

In questo caso invece le cifre significative sono 16 e non vengono fatti particolari arrotondamenti.
Si può quindi dare una seconda caratterizzazione della precisione di macchina (da noi non dimostrata):

$$\varepsilon_M = \min\{\mu \in \mathbb{F}^+ : 1 \oplus \mu > 1\}$$

Dato che non si può superare la precisione di macchina, l'errore chiave da stimare è <u>l'errore relativo</u>:

$$\varepsilon_{x \star y} = \frac{|(x \star y) - (fl^t(x) \star fl^t(y))|}{|(x \star y)|}$$

supponendo *x* ed *y* entrambi non nulli, tutto in funzione degli errori relativi sui dati ε_x ed ε_y:

$$\varepsilon_x = \frac{|x - \tilde{x}|}{|x|}, \quad x \neq 0$$

$$\varepsilon_y = \frac{|y - \tilde{y}|}{|y|}, \quad y \neq 0$$

da cui utilizzando i dati approssimati:

$$\varepsilon_{x \star y} = \frac{|(x \star y) - (\tilde{x} \star \tilde{y})|}{|(x \star y)|}, \quad x \star y \neq 0$$

Nel caso dei dati approssimati quindi diremo ***<u>stabile</u>*** un'operazione aritmetica per cui l'errore sul risultato ha lo stesso ordine di grandezza dell'errore (massimo) sui dati.
Iniziamo con la ***<u>moltiplicazione:</u>***

$$\varepsilon_{xy} = \frac{|xy - \tilde{x}\tilde{y}|}{|xy|}, \quad x, y \neq 0$$

Usando la stessa tecnica di stima che si usa per dimostrare che il limite del prodotto di due successioni o funzioni è il prodotto dei limiti, aggiungendo e togliendo a numeratore ad esempio ˜xy.

$$\varepsilon_{xy} = \frac{|\,xy - \tilde{x}\tilde{y} + \tilde{x}\tilde{y} - \tilde{x}\tilde{y}\,|}{|\,xy\,|}$$

$$= \frac{|\,\overbrace{y(x-\tilde{x})}^{=a} + \overbrace{\tilde{x}(y-\tilde{y})}^{=b}\,|}{|\,xy\,|}$$

$$\leq \frac{|\,y(x-\tilde{x}) + \tilde{x}(y-\tilde{y})\,|}{|\,xy\,|} \qquad (\bigstar)$$

$$||\,a\,| - |\,b\,|| \leq |\,a+b\,| \leq |\,a\,| + |\,b\,|$$

Usiamo quindi la *disuguaglianza triangolare* per operare correttamente questa stima otteniamo:

$$\varepsilon_{xy} \leq \frac{|\,y\,||\,x-\tilde{x}\,|}{|\,xy\,|} + \frac{|\,\tilde{x}\,||\,y-\tilde{y}\,|}{|\,xy\,|} = \varepsilon_x + \frac{|\,\tilde{x}\,|}{|\,x\,|}\varepsilon_y$$

La moltiplicazione è *stabile*, perché l'errore relativo sul risultato è maggiorato da una quantità che è dello stesso ordine di errore sui dati. Siccome $x \approx \sim x$, qualitativamente:

$$\varepsilon_{xy} \leq \varepsilon_x + \frac{|\,\tilde{x}\,|}{|\,x\,|}\varepsilon_y \approx \varepsilon_x + \varepsilon_y$$

Per esprimere questo fatto possiamo usare la notazione: $\varepsilon_{xy} \lesssim \varepsilon_x + \varepsilon_y$

La stima quindi, grazie anche alla *disuguaglianza triangolare* diventa:

$$\frac{|\,\tilde{x}\,|}{|\,x\,|} = \frac{|\,\overbrace{x}^{=a} + \overbrace{\tilde{x}-x}^{=b}\,|}{|\,x\,|} \leq \frac{|\,x\,| + |\,\tilde{x}-x\,|}{|\,x\,|} = 1 + \varepsilon_x$$

Passiamo ora alla *divisione*, essendo che è la moltiplicazione del reciproco, basta analizzare la stabilità dell'ordine di reciproco. Quindi:

$$\varepsilon_{\frac{1}{y}} = \frac{\left|\,\frac{1}{y} - \frac{1}{\tilde{y}}\,\right|}{\left|\,\frac{1}{y}\,\right|} = \frac{\frac{|\,\tilde{y}-y\,|}{|\,\tilde{y}y\,|}}{\left|\,\frac{1}{y}\,\right|} = \frac{|\,\tilde{y}-y\,|}{|\,y\,|} \cdot \frac{|\,y\,|}{|\,\tilde{y}\,|} \approx \varepsilon_y \qquad \left(\text{questo perchè } \frac{|\,\tilde{y}-y\,|}{|\,y\,|} = \varepsilon_y.\right)$$

Assumendo che l'errore relativo sia minore di 1, situazione ragionevole. Usando la stima da sotto nella disuguaglianza triangolare (usandola nel senso opposto, cioè):

$$|\,a+b\,| \geq ||\,a\,| - |\,b\,||$$

avremo:

$$\left|\,1 + \frac{(\tilde{y}-y)}{y}\,\right| \geq \left|\,1 - \frac{|\,\tilde{y}-y\,|}{|\,y\,|}\,\right| = |\,1-\varepsilon_y\,| = 1-\varepsilon_y \qquad \left(\text{perchè } \varepsilon_y < 1\right)$$

da cui si ottiene

$$|\,\tilde{y}\,| \geq |\,y\,|(1-\varepsilon_y)$$

e quindi

$$\frac{|\,y\,|}{|\,\tilde{y}\,|} \leq \frac{|\,y\,|}{|\,y\,|(1-\varepsilon_y)} = \frac{1+\varepsilon_y}{(1+\varepsilon_y)(1-\varepsilon_y)} = \frac{1+\varepsilon_y}{1-\varepsilon_y^2} \approx 1+\varepsilon_y$$

L'ordine del reciproco ha stabilità, in quanto anche qui maggiorato dall'errore su "y", considerando che l'errore relativo è <= 1. Parliamo quindi di *addizione* e di *sottrazione*.
Andremo quindi ad analizzare gli errori sui dati della somma algebrica "x + y".

Scritto da Gabriel

$$\begin{aligned}
\varepsilon_{x+y} &= \frac{|\,(x+y)-(\tilde{x}+\tilde{y})\,|}{|\,x+y\,|}, \quad x+y \neq 0 \\
&= \frac{|\,x-\tilde{x}+y-\tilde{y}\,|}{|\,x+y\,|}, \quad a = x-\tilde{x} \text{ e } b = y-\tilde{y} \\
&\leq \frac{|\,x-\tilde{x}\,|}{|\,x+y\,|} + \frac{|\,y-\tilde{y}\,|}{|\,x+y\,|}, \quad \text{DISUGUAGLIANZA TRIANGOLARE} \\
&= \frac{|\,x\,|}{|\,x+y\,|} \cdot \frac{|\,x-\tilde{x}\,|}{|\,x\,|} + \frac{|\,y\,|}{|\,x+y\,|} \cdot \frac{|\,y-\tilde{y}\,|}{|\,y\,|} \\
&= w_1\varepsilon_x + w_2\varepsilon_y
\end{aligned}$$

Attenzione che i pesi dipendono "x" e da "y", ma non dipendono dagli errori, avendo inoltre ragionato con una somma pesata degli errori sui dati con pesi ω_1 ed ω_2.
Per quanto riguarda l'addizione avremo (considerando che nell'addizione i pesi sono <= 1):

$$\text{ADDIZIONE}(sign(x) = sign(y))$$

in questo caso $|\,x+y\,| \geq |\,x\,|, |\,y\,|$ si pensi per semplicità al caso $x, y > 0$, è chiaro che $x+y > x$ e $x+y > y$)
Quindi

$$w_1 = \frac{|\,x\,|}{|\,x+y\,|} \leq 1 \quad , \quad w_2 = \frac{|\,y\,|}{|\,x+y\,|} \leq 1$$

cioè

$$\varepsilon_{x+y} \leq \varepsilon_x + \varepsilon_y$$

ovvero l'addizione è STABILE (l'errore relativo sul risultato è maggiorato da una quantità che è dell'ordine degli errori sui dati)

Nel caso invece della <u>*sottrazione*</u> avremo che $|x + y| < |y|$, quindi max $\{w_1, w_2\} > 1$, quindi la sottrazione può farci perdere precisione, ma quanta?
Può succedere che $|x|$ e $|y|$ siano molti vicini in termini relativi, cioè che $|x + y| << |x|, |y|$ ed in queste situazioni i pesi w_1, w_2 sono >> 1 e la sottrazione diventa <u>instabile</u>.
I casi instabili quindi non sono quelli in cui $|x + y|$ è "piccolo", ma quelli in cui è piccolo rispetto a $|x|, |y|$.
Ad esempio, sono analoghi:

$$\left.\begin{array}{l} |x|, |y| \approx 1\,, \; |x+y| \approx 10^{-6} \\ |x|, |y| \approx 10^{6}, \; |x+y| \approx 1 \end{array}\right\} \Rightarrow w_1, w_2 \approx 10^{6}$$

Detto a parole, due numeri dell'ordine delle unità che distano di qualche milionesimo sono altrettanto vicini, in termini relativi, a due numeri dell'ordine del milione che distano di qualche unità.
Da questo si può notare che la sottrazione è <u>potenzialmente instabile</u> (questo non accade sempre comunque). Se $|x|$ed $|y|$ sono *distanti* in termini relativi, la sottrazione perde *poca* precisione; se invece gli stessi sono *vicini* si perderà *molta* precisione (definito anche come *cancellazione numerica*) e ci può portare ad un algoritmo instabile.
Si fronteggia in 2 modi:

1) cercare di riscrivere espressioni ed algoritmi in modo tale da evitare sottrazioni instabili (ad esempio nella formula risolutiva delle equazioni di secondo grado, aggirabile riscrivendo la formula con una semplice manipolazione algebrica);
2) aumentando la precisione in funzione della grandezza di w_1 e di w_2.

In campo sperimentale, quindi, questo aumenta la precisione, andando ad un sistema floating point a precisione estesa. Si consideri questo esempio per fissare le idee:

Consideriamo la funzione

$$f(x) = \frac{((1+x)-1)}{x}, \quad x \neq 0$$

è evidente che $f(x) = 1$ (è la funzione costante 1) però, calcolando in Matlab $f(10^{-15})$ si ottiene $f(10^{-15}) = 1.11\ldots$ cioè l'errore relativo sul risultato è $> 11\%$ (un errore enorme rispetto all'arrotondamento che non supera $\varepsilon_M \approx 10^{-16}$)
Vedremo che la spiegazione sta nella sottrazione a numeratore, visto che $1 + 10^{-15}$ è vicinissimo ad 1.
Invece $f(2^{-50}) = 1$, pur essendo $2^{-50} \approx 10^{-15}$ perché?
Tratteremo entrambi i casi nella prossima lezione.

Alcuni commenti:
$2^{(-50)}$ non è un numero macchina, in base 2, e con $1 + 2^{(-50)}$ avrò 51 cifre significative, ma ne ho 53 a disposizione. Dato che si moltiplica 0, va bene e non si fa influenzare dall'arrotondamento.
Nel caso invece di $1 + 2^{(-53)}$arrotonderei ad 1 ed il risultato verrebbe 0, in quanto, similmente a prima, avrei bisogno di 54 cifre significative.

Lezione 5 – Esempi di instabilità della sottrazione

Nella scorsa lezione abbiamo analizzato la risposta delle operazioni aritmetiche agli errori sui dati. In particolare, dati x ed y in R, ed ~x circa x , ~y circa y, l'errore può essere espresso dalla disuguaglianza:

$$\varepsilon_{x \bigstar y} \leq w_1 \varepsilon_x + w_2 \varepsilon_y$$

I pesi w_1 e w_2 sono maggiori di 0 i quali possono dipendere da x,y (ma <u>non</u> dagli errori). Nel caso delle altre operazioni (moltiplicazione, divisione, addizione), w_1 e w_2 sono circa 1 oppure <= 1, dunque sono considerate *stabili*.
Nel caso della sottrazione i pesi:

$$w_1 = \frac{|x|}{|x+y|}, \quad w_2 = \frac{|y|}{|x+y|}$$

Se avessimo $|x| = -|y|$ possono essere invece grandi. Questo accade in particolare se x ed y sono vicini in termini relativi, rendendo l'operazione di sottrazione instabile e distruggendo completamente il risultato rendendolo privo di significato (quando $w_1, w_2 > max\{1/\varepsilon_x , 1/\varepsilon_y\}$), attendendosi un errore relativo > 100%.
Noi ci focalizzeremo su errori relativi, lavorando su sistemi floating point in base 10.
Alla luce di queste considerazioni, seguono esempi che dimostrano l'instabilità della sottrazione.

<u>Esempio 1</u>
Si consideri **F** (10, 4, L, U) (rappresentando con L ed U numeri da noi scelti), con x = 0.10016 ed y = -0.10012. Allora $\sim x = 0.1002$ (espressi entrambi come $fl^{(4)}$, arrotondamento a 4 cifre FP) ed ~y = -0.1001.
Eseguendo l'operazione macchina è:

$$\begin{aligned} x \oplus y &= fl^4(fl^4(x) + fl^4(y)) \\ &= fl^4(0,1002 - 0,1001) \\ &= 10^{-4} \end{aligned}$$

Invece $x + y = 4 * 10^{(-5)}$ e l'errore relativo sarà:

$$\frac{|(x+y)-(x+y)|}{|x+y|} = \frac{|4 \cdot 10^{-5} - 10^{-4}|}{4 \cdot 10^{-5}} = \frac{6 \cdot 10^{-5}}{4 \cdot 10^{-5}} = \frac{3}{2} = 150\%$$

Scritto da Gabriel

Si vede come avremo una perdita di precisione di ben 3 ordini di grandezza. Il problema è dai numeri scelti, dato che sono estremamente vicini tra di loro, con |x| ed |y| che sono circa $10^{(-1)}$ per ordine di grandezza. Infatti, se calcoliamo i pesi:

$$w_1 = \frac{|x|}{|x+y|} \approx \frac{10^{-1}}{4 \cdot 10^{-5}} = \frac{10^4}{4} = 2500$$

dove analogamente anche w_2 è circa 2500.
I fattori di amplificazione degli errori sono nell'ordine di $10^{(3)}$ e spiegano perché l'errore superi il 100%: si considera inoltre che questi fattori siano > $1/\varepsilon_r$. Se avessimo avuto una cifra di mantissa in più non occorreva neppure arrotondare.

Esempio 2

Considerando invece **F**(10,8,L,U) con $t = 8$ ed $\varepsilon_r = 10^{(-7)}/2$
Vogliamo calcolare in floating point la somma algebrica $a + b + c$, dove (si guardi soprattutto l'ordine di grandezza più che i numeri in sé):
a = $0.23371258 * 10^{(-4)}$, b = $0.33678429 * 10^{(2)}$ e c = $-0.33677811 * 10^{(2)}$
Il risultato della somma sarà: $0.64137126 * 10^{(-3)}$
Tuttavia non vale la proprietà associativa, come si vede da:

$$\begin{aligned} I &= (a \oplus b) \oplus c \\ &= 0,33678452 \cdot 10^2 \oplus (-0,33677811 \cdot 10^2) \\ &= 0,64100000 \cdot 10^{-3} \end{aligned}$$

$$\begin{aligned} II &= a \oplus (b \oplus c) \\ &= 0,23371258 \cdot 10^{-4} \oplus 0,61800000 \cdot 10^{-3}) \\ &= 0,64137126 \cdot 10^{-3} \end{aligned}$$

Si verifica inoltre che *I* è diverso da *II* e si verifica che l'arrotondamento vero di *II* è $fl^{(8)}(a+b+c)$.
Il risultato è quindi il meglio ottenibile da un'aritmetica floating point, con l'errore relativo di *I* dato da:

$$\frac{|I-(a+b+c)|}{|a+b+c|} \approx \frac{4 \cdot 10^{-7}}{0,6 \cdot 10^{-3}} = \frac{2}{3} \cdot 10^{-3} \approx 0,07\%$$

La perdita di precisione perde ben 4 ordini di grandezza, con un fenomeno di cancellazione di cifre significative, rispetto ad $\varepsilon_r = 10^{(-7)}/2$.
Ora calcolo l'ordine di grandezza dei pesi:

$$w_1 = \frac{|x|}{|x+y|} = \frac{|a+b|}{|a+b+c|} \approx \frac{0,3 \cdot 10^2}{0,6 \cdot 10^{-3}} = \frac{1}{2} \cdot 10^5 = 50000$$

e anche qui analogamente w_2 è circa 50000.
Nell'espressione di calcolo *II non ci sta perdita di precisione*: la spiegazione sta nel fatto che i numeri a, b, c entrano con 8 cifre significative e non occorre arrotondarli, cioè $\varepsilon_a = \varepsilon_b = \varepsilon_c = 0$ quindi la sottrazione "b+c" non perde precisione (invece ne perderebbe se b oppure c fossero arrotondati).
Nel caso invece della prima espressione, si ha un arrotondamento che, come si vede dal valore del peso, ben influisce sul risultato. *In generale in una sottrazione basta che uno dei due dati possegga errore e si perde precisione*, come nel caso della prima espressione appena descritto.

Esempio 3

Si consideri la funzione seguente (dove f(x) coincide (≡) con 1):

$$f(x) = \frac{(1+x)-1}{x} \quad , \quad x \neq 0$$

dove il Matlab esegue le operazioni in questo modo, avendo poi:

$$\tilde{f}(x) = \big((1 \overbrace{\oplus}^{\text{ADD}} x) \overbrace{\oplus}^{\text{SOTTR}} (-1)\big) \overbrace{\oslash}^{\text{DIV}} x$$

$$\tilde{f}(10^{-15}) = 1,11\ldots$$

dove l'errore relativo nel calcolo è maggiore dell'11%, con la differenza tra 1 e 1.11.
Anche qui, 1 e $(1+x)$ sono estremamente vicini, con il secondo che viene arrotondato, mentre il primo no. Questi piccolissimi errori di arrotondamento *vengono però amplificati dal peso* w_1 nella sottrazione $(1+x)-1$, $x = 10^{(-15)}$.

Calcoliamo poi:

$$w_1 = \frac{|\,1+x\,|}{|\,(1+x)-1\,|} = \frac{1+x}{x} \approx 10^{15}$$

Nel caso invece della sottrazione $(1+x)$ – 1 con $x = 2^{(-50)}$ è sempre dell'ordine di $10^{(15)}$ e quindi il risultato sarà esattamente uguale ad 1, perché sia $2^{(-50)}$ che $1 + 2^{(-50)}$ sono entrambi reali-macchina in base 2, pertanto non vengono arrotondati e anche il fattore di amplificazione non ha effetto.

<u>Esempio 4</u>
Consideriamo un'equazione di secondo grado del tipo
$az^{(2)} + bz + c = 0$, dove sappiamo che le soluzioni si calcolano come: $z_{\pm} = \dfrac{-b \pm \sqrt{\Delta}}{2a}$

Attenzione: in aritmetica floating point la radice viene sempre arrotondata, dato che in essa vi è sempre un errore, dipendente sempre dal peso nel caso della sottrazione.
Prendiamo in oltre per semplicità $b > 0$, con la sottrazione $\Delta^{(1/2)} - b$ il primo dato sicuramente affetto da un errore al massimo dell'ordine della precisione di macchina, considerando che nella sottrazione si perde precisione a causa di $\Delta^{(1/2)}$, già arrotondata di per sé e accade quando vicina a "b" in termini relativi.

Posti quindi x come $\Delta^{(1/2)}$ e $y = -b$, avremo:

$$w_1 = \frac{|\,x\,|}{|\,x+y\,|} = \frac{\sqrt{\Delta}}{|\,\sqrt{\Delta}-b\,|} = \frac{\sqrt{\Delta}\cdot(\sqrt{\Delta}+b)}{|\,\sqrt{\Delta}-b\,|\cdot(\sqrt{\Delta}+b)} = \frac{\sqrt{\Delta}\cdot(\sqrt{\Delta}+b)}{|\,\Delta-b^2\,|} = \frac{\sqrt{\Delta}\cdot(\sqrt{\Delta}+b)}{|\,b^2-(b^2-4ac)\,|}$$

Ma $\sqrt{\Delta} \approx b$ quindi

$$w_1 = \frac{\sqrt{\Delta}\cdot(\sqrt{\Delta}+b)}{|\,4ac\,|} \approx \frac{2b^2}{|\,4ac\,|} = \frac{b^2}{2\cdot|\,ac\,|}$$

e analogamente

$$w_2 = \frac{|\,-b\,|}{|\,\sqrt{\Delta}-b\,|} = \frac{b\cdot(\sqrt{\Delta}+b)}{|\,4ac\,|} \approx \frac{b^2}{2\cdot|\,ac\,|}$$

Possiamo quindi perdere molta precisione, ma successivamente basterà arrotondare la sottrazione. Quindi appunto la soluzione $z+$ subisce fortissimi arrotondamenti e perdita di precisione a causa del rapporto che può diventare molto grande.

<u>Ulteriore esempio</u>
$z^{(2)} + 100z - 1 = 0$, con un $\Delta = 10004$, arrotondato poi a 10000 (avendo $0.10004 * 10^{(5)}$). Quindi viene calcolato $\Delta^{(1/2)} = 100$. Le soluzioni portano quindi ad un possibile errore relativo del 100%.
Per effetto dei coefficienti di amplificazione (quindi si vede come siano i pesi a determinare, oltre all'operazione stessa, la precisione del risultato, influenzati dall'algoritmo di calcolo).

$$w_1, w_2 \approx \frac{b^2}{2\,|\,ac\,|} \approx \frac{10^4}{2} = 5000$$

Consideriamo poi un ulteriore esempio:

Consideriamo $\mathbb{F}(10, 8, L, U)$ cioè passiamo a $t = 8$ cifre di mantissa con $\varepsilon_M = \frac{10^{-7}}{2}$
In questo caso si può verificare che

$$\tilde{z}_+ = \frac{-100 + \sqrt{10004}}{2} = \frac{-100 + 100,02000}{2} = 10^{-2}$$

mentre il valore "esatto" è

$$z_+ = 0.0099990002$$

per cui abbiamo un errore relativo

$$\frac{|\, z_+ - \tilde{z}_+ \,|}{|\, z_+ \,|} \approx 10^{-4}$$

Commettiamo un errore dello 0.01%, errore che potrebbe essere comunque accettabile in molti contesti applicativi, ma è 2000 volte più grande della precisione di macchina (poteva esserlo fino a 5000 volte, ma va tenuto presente che la precisione di macchina è una soglia massima e che gli errori relativi sono di solito più piccoli di essa). Anche qui la perdita dei tre ordini di grandezza proviene dall'influsso ricevuto dai pesi stessi. Facciamo un altro esempio considerando:

$$10^{-2}z^2 + 10^4 z + 10^{-2} = 0$$

in $\mathbb{F}(10, 16, L, U)$ (sostanzialmente la precisione dell'interfaccia del Matlab).

con un errore relativo totale, addirittura del 100%.

Con $t = 16$ cifre di mantissa si calcola

$$\tilde{z}_+ = \frac{-10^4 + \sqrt{10^8 - 4 \cdot 10^{-4}}}{2 \cdot 10^{-2}} = -(0, 9999894 \ldots 46) \cdot 10^{-6}$$

mentre la soluzione "esatta" (arrotondata a 16 cifre) è

$$z_+ = -(0.1000000000001000) \cdot 10^{-5}$$

con errore relativo

$$\frac{|\, z_+ - \tilde{z}_+ \,|}{|\, z_+ \,|} \approx \frac{1.1 \cdot 10^{-5} \cdot 10^{-6}}{10^{-6}} = 1.1 \cdot 10^{-5}$$

Si vede quindi che la formula delle equazioni di secondo grado sia molto instabile. Invece:

Considerando nuovamente il caso $b > 0$

$$z_+ = \frac{\sqrt{\Delta} - b}{2a} = \frac{(\sqrt{\Delta} - b)(\sqrt{\Delta} + b)}{2a(\sqrt{\Delta} + b)} = \frac{\Delta - b^2}{2a(\sqrt{\Delta} + b)} = \frac{-4ac}{2a(\sqrt{\Delta} + b)} = -\frac{2c}{\sqrt{\Delta} + b}$$

usando una formula stabile eliminando la sottrazione: $z_+ = -\frac{2c}{\sqrt{\Delta} + b}$

avendo poi una formula stabilizzata, ottenuta combinando operazione sempre è possibile eliminare la sottrazione):

$$\begin{cases} z_1 = -sign(b)\frac{2c}{|\,b\,|+\sqrt{\Delta}} \\ z_2 = -sign(b)\frac{|\,b\,|+\sqrt{\Delta}}{2a} \end{cases}$$

stabili (non

La perdita di precisione quindi non è completamente eliminabile, data anche la sottrazione nel caso $b^{(2)} - 4ac$; tuttavia può essere approssimata grazie all'uso di un algoritmo che cambia l'ordine d'errore delle soluzioni stesse, avvicinandolo all'errore relativo.

Lezione 6 - Propagazione degli errori, condizionamento delle funzioni

Ci interessa in questa lezione l'effetto degli errori di arrotondamento su una variabile generica e la costruzione di un algoritmo iterativo che sfrutta una successione convergente alla quantità approssimabile.
Qual è l'effetto sul valore di *f* della variabile indipendente *x*?
Consideriamo quindi una funzione dove I è intervallo, con $\sim x \approx x, x \neq 0$, con il solito errore relativo ε_x.

Cerchiamo di stimare $\varepsilon_{f(x)}$

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

Supponendo f derivabile in I possiamo utilizzare la formula di Taylor al primo ordine centrata in x

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

da cui ricaviamo

$$f(\tilde{x}) - f(x) \approx f'(x) \cdot (\tilde{x} - x) \iff \frac{|f(\tilde{x}) - f(x)|}{|f(x)|} \approx \frac{|f'(x)|}{|f(x)|} \cdot |\tilde{x} - x| = \frac{|f'(x)| \cdot |x|}{|f(x)|} \cdot \frac{|\tilde{x} - x|}{|x|}$$

che possiamo riassumere con

$$\varepsilon_{f(x)} \approx condf(x)\varepsilon_x$$

dove

$$condf(x) = \frac{|f'(x)| \cdot |x|}{|f(x)|}$$

Il cosiddetto *cond* rappresenta *l'indice di condizionamento* di f in x ed è la quantità che misura la risposta della funzione ad errori e quantifica l'amplificazione dell'errore sulla variabile.
Generalmente il condizionamento viene definito come riguarda il rapporto tra errore commesso sul risultato di un calcolo e incertezza sui dati in ingresso. Abbiamo il primo esempio di *problema instabile*, definibile in modo empirico e non formale. Cominciamo quindi dal considerare:

$$f(x) = \frac{(1+x) - 1}{x}, \quad x \neq 0$$

Questa è la funzione costante 1, che risponde in modo ottimale agli errori su *x*, dato che ovunque vale 1 e l'errore relativo vale 0. In aritmetica floating-point il calcolo è instabile se usiamo la formula scritta nel modo di prima. Per x → 0, il peso w_1 tende a $+\infty$, a causa della piccolezza di |x|, crescendo la sottrazione ma avendo condizionamento ottimale.
Attenzione però: in sé *la funzione è perfettamente stabile*, tuttavia *è l'algoritmo ad essere instabile*.
Quello appena analizzato è un caso limite, ma ora diamo esempi di casi meno estremi:

- $f(x) = \frac{1}{x}, \quad x \neq 0$

$$condf(x) = \frac{\left|-\frac{1}{x^2}\right| \cdot |x|}{\left|\frac{1}{x}\right|} = 1$$

questa è una funzione stabile $\forall x$, in effetti lo sappiamo già, è l'operazione di reciproco

- $f(x) = 1 - x$

$$condf(x) = \frac{|(-1)||x|}{|1-x|} = \frac{|x|}{|1-x|}$$

ora $condf(x) \to +\infty, \quad x \to 1$ quindi la funzione è instabile per $x \approx 1$.
Di nuovo non è una sorpresa, si tratta di una sottrazione instabile per $x \approx 1$, qui $condf(x) = w_2$

- $f(x) = 1 - \sqrt{1-x^2}, \quad 0 < |x| \leq 1$

$$f'(x) = -\frac{1}{2\sqrt{1-x^2}} \cdot (-2x) = \frac{x}{\sqrt{1-x^2}}$$

(derivata di una funzione composta)

dove il condizionamento (con qualche passaggio algebrico):

$$\begin{aligned} condf(x) &= \frac{|x|}{\sqrt{1-x^2}} \cdot \frac{|x|}{|f(x)|} \\ &= \frac{x^2}{\sqrt{1-x^2}} \cdot \frac{1}{1-\sqrt{1-x^2}} \\ &= \frac{x^2}{\sqrt{1-x^2}} \cdot \frac{1+\sqrt{1-x^2}}{(1+\sqrt{1-x^2})(1-\sqrt{1-x^2})} \\ &= \frac{x^2(1+\sqrt{1-x^2})}{\sqrt{1-x^2}(1-(1-x^2))} \\ &= \frac{1+\sqrt{1-x^2}}{\sqrt{1-x^2}} \end{aligned}$$

Avremo quindi che la funzione è ben condizionata, avendo che *condf(x)* → 2.
La formula di calcolo è quella che fa perdere precisione; sfrutteremo il solito trucco per rendere la funzione ed il calcolo stesso stabile, creandosi un altro algoritmo stabile.
Ciò vale quando x è piccolo in modulo, quindi x ≈ 0.
Tuttavia il calcolo sull'espressione:

$$f(x) = 1 - \sqrt{1-x^2}$$

è *stabile* nella sottrazione *interna* alla radice, *instabile* invece nella sottrazione *esterna*.
Questo succede perché, per x → 0, i numeri si allontanano in termini relativi e si vede da:

$$w_2 = \frac{\sqrt{1-x^2}}{1-\sqrt{1-x^2}} = \frac{\sqrt{1-x^2} \cdot (1+\sqrt{1-x^2})}{x^2} \to +\infty$$

La formula quindi si stabilizza per x ≈ 0:

$$\begin{aligned} f(x) &= 1 - \sqrt{1-x^2} \\ &= \frac{(1-\sqrt{1-x^2}) \cdot (1+\sqrt{1-x^2})}{1+\sqrt{1-x^2}} \\ &= \frac{1-(1-x^2)}{1+\sqrt{1-x^2}} \\ &= \frac{x^2}{1+\sqrt{1-x^2}} \end{aligned}$$

Questo succede perché tutte le operazioni coinvolte sono stabili e la funzione in sé strutturalmente risponde agli errori sulla variabile, che non dipende da come è scritta (quindi qual è l'algoritmo di calcolo) ma solo dall'indice di condizionamento *condf(x)*.
Dunque sta a noi scegliere un algoritmo stabile nel caso di una funzione ben condizionata e quindi errori piccoli, come arrotondamenti, possono dare grossi errori nei risultati.

Ci si aspetta che se una funzione sia ben condizionata, *condf(x)* non sia grande. Questo accade ad esempio:

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

è stabile per $x \approx 0$, infatti $condf(x) \approx 2$ (cioè non è grande) per $x \approx 0$.
Viceversa, se una funzione è instabile ci aspettiamo che gli algoritmi siano a loro volta instabili.
Tutti questi concetti sono quindi empirici, perché ragioniamo in termini puramente astratti.

Scritto da Gabriel

Nel caso sottostante ci si aspetta di avere errori, avendo una funzione in sé mal condizionata:

Per concludere con l'esempio, osserviamo che $f(x) = 1 - \sqrt{1-x^2}$ è ben condizionata per $x \approx 0$, ma è mal condizionata per $|x| \approx 1$ visto che

$$condf(x) = \frac{1+\sqrt{1-x^2}}{\sqrt{1-x^2}} \to +\infty$$

per $|x| \to 1$.

In questa seconda parte, abbiamo il calcolo di una successione che *converge* alla quantità che vogliamo approssimare tramite un *algoritmo iterativo*.
Affrontiamo quindi il calcolo di π in un sistema floating-point a 64 bit con un errore della precisione di macchina (avendo 15 cifre corrette nell'interfaccia in base 10).
Consideriamo quindi due successioni convergenti a π.
La prima si può ottenere dal fatto che la cosiddetta serie armonica ha come somma: $\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$

Detta poi la somma come $1/k^{(2)}$, avremo: $\sum_{k=1}^{\infty} \frac{1}{k^2} = \lim_{n\to\infty} S_n = \frac{\pi^2}{6}$

$$\lim_{n\to\infty} \sqrt{6 \cdot S_n} = \pi$$

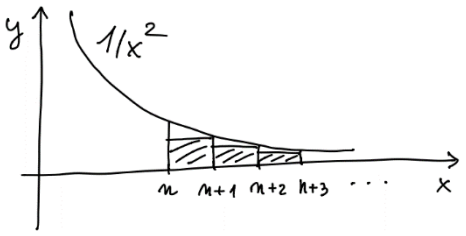
Il calcolo della serie anche qui è stabile perché coinvolge solo operazioni stabili, ma la successione sotto radice è inutilizzabile per calcolare π con grande precisione, dato che la successione in sé converge lentamente.
Per il criterio di confronto integrale con serie:

$$\frac{\pi^2}{6} - S_n = \sum_{k=n+1}^{\infty} \frac{1}{k^2}$$

si può maggiorare nel modo seguente

$$\sum_{k=n+1}^{\infty} \frac{1}{k^2} < \int_n^{\infty} \frac{1}{x^2} dx = \frac{1}{n}$$

schematizzato poi da:

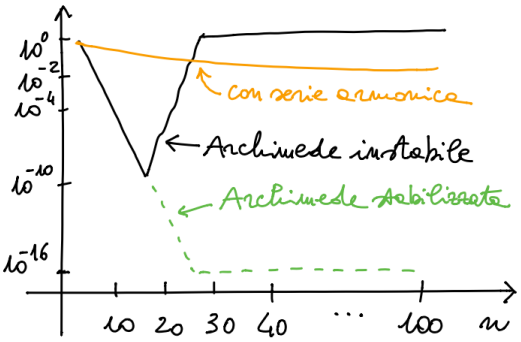


Avremo quindi una stima approssimata con: $\frac{\pi^2}{6} - S_n < \frac{1}{n}$

Per approssimare quindi correttamente l'errore in precisione macchina, dovremmo sommare $10^{(16)}$.
Nel caso invece di una successione definita per ricorrenza, cosiddetta "successione di Archimede", avremo che il limite della successione per $x \to \infty$ tende a π.
Mostriamo quindi com'è fatta questa successione (nota: il prof ha corretto qui e mette x_2 e non x_0)

$$x_0 = 2$$
$$x_{n+1} = 2^{n-\frac{1}{2}} \cdot \sqrt{1 - \sqrt{1-(4^{1-n})x_n^2}}, \quad n = 2, 3, 4, \ldots$$

Calcolando in Matlab l'errore relativo in scala logaritmica e ciclando in maniera regolare, ottiene in maniera grafica:



L'errore decresce linearmente fino a circa $10^{(-10)}$ dopodiché comincia a crescere, dato che la successione con lo schema descritto comincia ad allontanarsi da π, avendo poi un decadimento esponenziale, come si vede da qui:

$$log_{10}(r_n) = -\alpha\, n + \beta \quad \text{con } \alpha > 0$$

cioè in scala normale

$$r_n = 10^{-\alpha n+\beta} = \gamma\, \theta^n \quad \text{con} \quad \gamma = 10^{\beta} \quad \text{e} \quad \theta = 10^{-\alpha} < 1$$

Dunque l'algoritmo è instabile e questa instabilità, da un certo *n* in poi, distrugge la convergenza della successione.

$$1 - \sqrt{1 - \alpha_n} \quad \text{con } \alpha_n = (4^{1-n})x_n^2$$

Di fatto la prima sottrazione risulta stabile perché a_n è un infinitesimo, invece la seconda (quella ad esponente a destra) diventa sempre più instabile al crescere di n, avendo una rapida convergenza ad 1.
Il peso w_2 di questa sottrazione sarà sicuramente affetto da errore (dato dalla radice a numeratore), in cui:

$$w_2 = \frac{|\,y\,|}{|\,x+y\,|} = \frac{\sqrt{1-\alpha_n}}{1-\sqrt{1-\alpha_n}}$$

che sarà circa → $\approx \frac{2}{\alpha_n} = 2 \cdot \frac{4^{n-1}}{x_n^2} \approx \frac{1}{2\pi^2} 4^n$

La stima dell'errore è dato da una combinazione della convergenza e la stabilità in questo modo:

$$e_n = |\,\pi - x_n + x_n - \tilde{x}_n\,| \leq \underbrace{|\,\pi - x_n\,|}_{\text{CONVERGENZA}} + \underbrace{|\,x_n - \tilde{x}_n\,|}_{\text{STABILITÀ}}$$

e l'errore $|\pi - x_n|$ è legato alla *convergenza teorica*, avendo: $\lim_{x\to\infty} x_n = l$
mentre l'altro $|x_n - \tilde{x}_n|$ è legato alla *stabilità dell'algoritmo* che implementa il metodo.

Dall'analisi fatta,

$$|\,x_n - \pi\,| = c\,\theta^n \quad \text{con } 0 < \theta < 1$$

mentre

$$|\tilde{x}_n - x_n| \lesssim c' \cdot 4^n\, \varepsilon_M$$

con c e c' costanti. Quindi

$$e_n \lesssim c\,\theta^n + c'\,4^n\,\varepsilon_M$$

da cui ci aspettiamo che finché

$$c'\,4^n\,\varepsilon_M < c\,\theta^n$$

Il termine esponenziale sovrasta e allontana la successione dal valore calcolato; tuttavia accade che l'errore diventi costante quando a_n è molto piccolo, $\tilde{x}_n = 0$ e l'errore è del 100%. Inoltre:

Il cuore del problema è infatti il calcolo di $1-\sqrt{1-\alpha_n}$:

$$\begin{aligned} 1-\sqrt{1-\alpha_n} &= \frac{(1-\sqrt{1-\alpha_n})(1+\sqrt{1-\alpha_n})}{1+\sqrt{1-\alpha_n}} \\ &= \frac{1-(1-\alpha_n)}{1+\sqrt{1-\alpha_n}} \\ &= \frac{\alpha_n}{1+\sqrt{1-\alpha_n}} \end{aligned}$$

Dunque l'iterazione viene scritta (due formule coincidenti, ma in FP la seconda è stabile, la prima fortemente instabile):

$$\begin{aligned} x_{n+1} &= 2^{n-\frac{1}{2}}\sqrt{1-\sqrt{1-\alpha_n}} \quad \Longleftarrow \quad \text{INSTABILE} \\ &= 2^{n-\frac{1}{2}}\sqrt{\frac{\alpha_n}{1+\sqrt{1-\alpha_n}}} \\ &= \frac{\sqrt{2}\,x_n}{\sqrt{1+\sqrt{1-(4^{1-n})x_n^2}}} \quad \Longleftarrow \quad \text{STABILE} \end{aligned}$$

Nel grafico degli errori presente sopra la successione stabiilizzata ha un errore coincidente con quello della successione, finché in quest'ultima si innesca l'instabilità distruttiva, stabilizzandosi poi sotto l'ordine della precisione di macchina, dato che si comincia a dominare sugli errori di arrotondamento.
Questa cosa detta si vede in termini formali dalla stima sottostante (dove ci aspettiamo che per *n* abbastanza grande, ci aspettiamo che il secondo addendo dell'ultima somma sia dominante nell'errore:

$$|\pi-\tilde{y}_n| \le |\pi-y_n| + |y_n-\tilde{y}_n| \lesssim c\,\theta^n + c''\,\varepsilon_M \qquad \text{con } c'' \approx \pi$$

Lezione 7 – Costo computazionale degli algoritmi numerici

Partiamo da due concetti base del calcolo numerico:

- *convergenza*, dove la maggior parte dei metodi numerici prevede la costruzione di una successione (di somme parziali oppure insieme di oggetti, come vedremo) che converge in senso opportuno ad un oggetto limite, oggetto a cui approssimare. Si tratta di un processo infinito che va fermato e prefissato intorno ad un limite, determinato da una certa tolleranza.
  Lo schema tipico che viene seguito è:
  errore(n)/ $e_n = |x_n - L| <=$ *stima(n)* $<=$ *tolleranza(ε)*
  dove la successione x_n tende ad L per n -> ∞

- *stabilità*, in tutti gli algoritmi numerici, anche quelli che fanno a priori un numero finito di passi (come alcuni algoritmi ad es. il metodo di eliminazione di Gauss), vengono introdotti errori durante il processo di calcolo, come errori di arrotondamento/di misura dei dati o anche dovuti all'uso di un algoritmo secondario che fornisce all'algoritmo primario dei risultati approssimati da elaborare. Cerchiamo quindi algoritmi che siano sia *convergenti* che *stabili*.

- *efficienza*, in cui cerchiamo gli algoritmi con basso costo computazionale (qui parlando di costo a *parità di errore*, perché si sa che il risultato non è mai esatto, poiché approssimato ad una certa tolleranza). Ad esempio l'algoritmo di Archimede stabilizzato è molto efficiente, riducendo la soglia esponenziale che ci sarebbe normalmente.

Per calcolare la complessità computazionale, consideriamo quindi due parametri:

- *numero/# di flops* (floating point operations, operazioni in virgola mobile)
- *tempo di calcolo* in merito alle singole operazioni eseguite. I processori puntano ad una potenza di calcolo che si aggira sui Petaflops, la tecnologia agli Exaflops. Questo è il parametro più importante e dipende dal tipo di macchina e di processore/memoria che si sta usando.

Il tempo di calcolo è influenzato anche dalla velocità dei flussi di dati tra le varie parti della macchina (*machine dependent*). Il numero di flops invece è *machine independent*, dando quindi una misura parzialmente incompleta ma universale. L'effetto dei flussi di dati viene mostrato con un esempio semplificato. Ad esempio l'accesso veloce alla

memoria centrale che prelevano dati più velocemente di quanti ce ne siano. Si cerca di minimizzare i flussi di dati e assumiamo un modello molto semplice:

PROCESSORE ←→ MEMORIA CENTRALE (RAM) ←→ HARD DISK

e supponendo di dover fare un prodotto di due matrici, con il vincolo che nella memoria centrale si può memorizzare solo 1 matrice e qualche vettore di dimensione n, ma non due matrici. Definiamo come "float" un reale macchina e, prendendo come esempio una RAM da 8 GB, possiamo memorizzare circa 1 miliardo di floats, avendo 8 bytes. Se avessimo un numero troppo grande (ad es. in questo caso 30000), nella RAM, non ci stanno entrambe le matrici e una di queste deve essere memorizzata nell'hard disk, così come la matrice prodotto.

Ricordiamo che: $$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

cioè prodotto riga/colonna in una matrice, ha un costo cubico, avendo infatti un costo quadratico di prodotti riga/colonna, n prodotti ed $n - 1$ somme algebriche. Qui avremo asintoticamente $2n^{(3)}$ come costo del calcolo misurato in floats.
Possiamo costruire la matrice C per righe e, man mano che le costruiamo, le memorizziamo nell'hard disk. Per ogni riga dobbiamo spostare dall'hard disk alla RAM tutte le colonne, viceversa invece con la riga risultato.
Naturalmente, non è il modo migliore di procedere, osservando in generale che il prodotto matrice-vettore è combinazione lineare delle colonne e della matrice che ha per coefficienti gli elementi del vettore, scritta così in notazione vettoriale:

$$A \cdot \begin{bmatrix} u_1 \\ u_2 \\ \dots \\ u_n \end{bmatrix} = \sum_{j=1}^{n} u_j \cdot \overset{\text{colonna } j}{\downarrow} C_j \quad (A)$$

Costruendo invece la matrice per colonne, ogni colonna di B viene spostata una volta sola, rispetto a prima dove si doveva forzatamente spostare tutto. Il flusso dei dati si riduce a 2n floats, miglioramento sostanziale anche sul tempo di calcolo.
Non ci occuperemo di algoritmi che elaborano grandi masse di dati, ma faremo esempi di confronti di algoritmi che risolvono lo stesso problema con casi computazionali diversi.
Prendiamo vari esempi:

Esempio 1: calcolo del valore di un polinomio
Per esempio prendiamo:

$$p(x) = a_0 + a_1 x + \ldots + a_n x^n$$

in cui si vuole calcolare il valore di p in un punto e l'idea è di usare un ciclo *for* dipendente dal grado *n* del polinomio, prendendo i monomi e sommandoli linearmente così:

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

avendo costo $c(n) = 3n$ flops. Possiamo procedere in altri modi però, ad esempio con un polinomio di grado 3:

$$p(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 = ((a_3 x + a_2) \cdot x + a_1) \cdot x + a_0$$

Qui le potenze non appaiono esplicitamente, ma sono implicite nella rappresentazione. Quindi in generale, dando la sottostante rappresentazione dove non si calcolano le potenze:

Scritto da Gabriel

$$p(x) = a_0 + a_1 x + \ldots + a_n x^n$$
$$= (\ldots((a_n x + a_{n-1})x + a_{n-2})x + \ldots)x) + a_0$$

avremo come costo: $c_n^{(2)} = 2n \cdot flops$

Non dovremo calcolare le potenze di x, con un costo dato da:
Lo *speedup*/guadagno dell'algoritmo 2 è (abbattendo il numero di moltiplicazioni, basato sullo schema di Horner):

$$speed - up = \frac{c_n^{(1)}}{c_n^{(2)}} = \frac{3}{2}$$

Esempio 2: calcolo di una potenza ad esponente intero

Il problema è il calcolo di a^n, dove sappiamo che la potenza è definita come la moltiplicazione *n* volte di "a" e il costo computazionale è n – 1.
È possibile però, diversificare, considerando che se n è potenza di 2, $n = 2^n$, si può calcolare a^n facendo solo m moltiplicazioni.
Per capirlo prendiamo $n = 16 = 2^4$

$$4 \text{ moltiplic.} \begin{cases} a^2 = a \cdot a \\ a^4 = a^2 \cdot a^2 \\ a^8 = a^4 \cdot a^4 \\ a^{16} = a^8 \cdot a^8 \end{cases}$$

dove $a^{\wedge}(2m)$ si calcola con $m = \log_2(n)$ moltiplicazioni.
Se x non fosse una potenza di 2, la rappresentazione sarebbe:

$$n = \sum_{j=0}^{m} c_j 2^j$$

dove $c_j \in \{0,1\}$ sono le cifre binarie e $m = [log_2(n)]$ (dove $[z]$ indica la parte intera, cioè il più piccolo intero $\leq z \in \mathbb{R}$). Ad esempio

$$7 = 1 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2 = (111)_2$$

Usando poi la proprietà delle potenze:

$$a^n = a^{\sum_{j=0}^{m} c_j 2^j} = \prod_{j=0}^{m} a^{c_j 2^j}$$

con ad esempio:

$$a^7 = a^{1\cdot 2^0 + 1\cdot 2^1 + 1\cdot 2^2} = a \cdot a^2 \cdot a^4$$
$$a^{12} = a^{0\cdot 2^0 + 0\cdot 2^1 + 1\cdot 2^2 + 1\cdot 2^3} = a^0 \cdot a^0 \cdot a^4 \cdot a^8 = a^4 \cdot a^8$$

Quindi $m = [log2(n)]$ moltiplicazioni per calcolare $a^{(2)}, a^{(4)}, \ldots, a^{(2m)}$ e nella produttoria c'è un numero di moltiplicazioni uguale al numero di cifre 1 nella codifica binaria – 1.

Siccome $\#\{1\}$ è al massimo $m+1$ e questo accade per $n = 2^k - 1$ (ad es. $15 = 2^4 - 1 = (1111)_2$) in cui la codifica binaria di n è una sequenza di 1 si ha che

$$max\, c_n^{(2)} = m + m = 2 \cdot [log_2(n)]$$

Quindi lo speed up minimo sarà:

$$min\ Speed-Up = \frac{c_n^{(1)}}{max\ c_n^{(2)}} = \frac{n-1}{2[log_2(n)]}$$

al crescere di n si ha che $min\ Speed-Up \sim \frac{n}{2log_2 n}$ (dove $a_n \sim b_n$ indica che $\lim_{n->\infty} \frac{a_n}{b_n} = 1$) mentre

$$max\ Speed-Up \sim \frac{n}{log_2(n)} = \frac{2^m}{m}$$

che si ottiene quando n è una potenza di 2 come visto all'inizio.

Lo speedup è comunque notevole, essendo proporzionale a $n/log(2)[n]$.

Per fissare le idee, $a^{(100)}$ richiederebbe 99 moltiplicazioni con l'algoritmo 1 e solo 8 moltiplicazioni con l'algoritmo 2. Lo speedup (dato proprio da 99/8) è circa 12.4.

Si potrebbe invece pensare che per calcolare a^n sia di considerarlo come $a^n = e^{n\,log(a)}$, usando algoritmi veloci.

Parliamo ora di funzioni, con *l'approssimazione della funzione esponenziale*, utilizzando la formula di Taylor centrata in 0, come ad esempio: $e^x = t_{m-1}(x) + R_m(x)$

dove

$$t_{m-1}(x) = \sum_{j=0}^{m-1} \frac{x^j}{j!}$$

è il polinomio di Taylor di grado $m-1$ e $R_m(x)$ il resto che possiamo scrivere in forma di Lagrange

$$R_m(x) = (e^{\xi})\frac{x^m}{m!}$$

con $\xi \in (0,x)$ supponendo $x > 0$ (se $x < 0$, $e^x = e^{-|x|} = \frac{1}{e^{|x|}}$).

In generale centrando la formula in x_0 per f derivabile m volte

$$t_{m-1}(x) = \sum_{j=0}^{m-1} \frac{f^{(j)}(x_0)}{j!}(x-x_0)^j$$

$$R_m(x) = \frac{f^{(m)}(\xi)}{m!}(x-x_0)^m$$

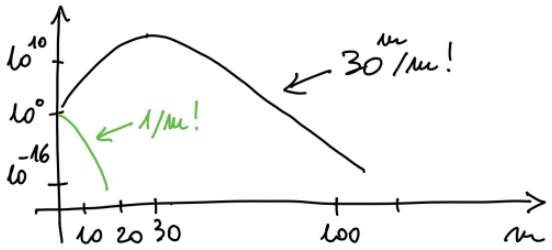
dove $\xi \in int(x_0,x) = (min\{x_0,x\}, max\{x_0,x\})$

L'errore relativo quindi commesso approssimando è:

$$\frac{|\,e^x - t_{m-1}(x)\,|}{e^x} = \frac{R_m(x)}{e^x} = \frac{e^{\xi}}{e^x}\cdot\frac{x^m}{m!} < \frac{x^m}{m!}$$

Il fattoriale ha una crescita estremamente rapida, mentre:

Invece per $x > 1$ la stima dell'errore relativo $\frac{x^m}{m!}$ non è decrescente (in m per x fissato) come per $x \leq 1$, una ha un massimo in corrispondenza di $m = [x]$ e poi decade rapidamente come si vede in questo grafico in scala log (con $x = 30$)



Conseguentemente il calcolo di $e^{(x)}$ alla precisione di macchina è molto efficiente per x <= 1 ma richiede un polinomio di Taylor di grado > x per x > 1.

Scritto da Gabriel

Tuttavia sfruttando le proprietà della funzione esponenziale si può adottare il trucco: $e^x = (e^{\frac{x}{n}})^n$
Il calcolo quindi è stabile perché tutte le operazioni (addizioni/moltiplicazioni) sono stabili e, avendo t(18) che permette di avere un calcolo con errore relativo < 1, la precisione macchina è già sufficiente per ottenere una buona precisione, sfruttando questa scalabilità della potenza rapida applicata sull'argomento dell'esponenziale. Grazie a questa "furbata" dell'approssimazione, si ha una stabilità della cosa e anche una buona efficienza, con basso costo computazionale.

Capitolo 2: Soluzione numerica di equazioni non lineari

Lezione 8: Introduzione alla soluzione numerica di equazioni non lineari, metodo di bisezione

Ci occuperemo del calcolo approssimato della soluzione numerica di equazioni non lineari, trattando sia zeri di funzione (*f(x) = 0*) ed equazioni di punto fisso (*x = φ(x)*)
In entrambi i casi daremo condizioni sufficienti per esistenza ed unicità della soluzione in un certo intervallo.
Ricordiamo a tale scopo, un fondamentale teorema:

TEOREMA (degli zeri di funzioni continue)

> Sia $f(x) \in C[a,b]$ (cioè f continua nell'intervallo chiuso e limitato $[a,b]$ e $f(a)f(b) < 0$ cioè f cambia segno agli estremi) allora
>
> $$\exists \xi \in (a,b) : f(\xi) = 0$$

Togliendo una delle ipotesi, la condizione restante non basta a garantire l'esistenza degli zeri, dato che può cambiare segno ed avere un salto.
Diamo due classiche condizioni sufficienti per garantire l'unicità dello zero:

- f strettamente monotona (strett. crescente/decrescente in [a,b]
- f strettamente convessa/concava ($f(a)f(b) < 0$ e C ∈ (a,b))

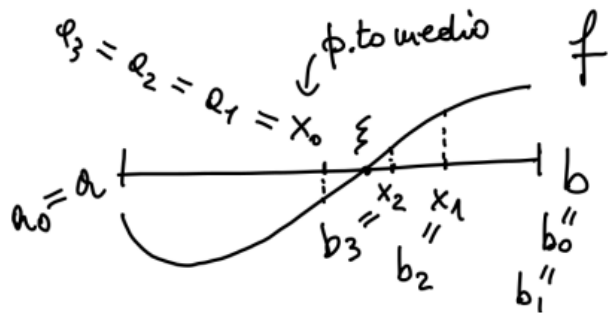
La monotonia è quindi legata al segno della derivata prima, con f che ha:

- derivata prima maggiore di 0 → f strettamente crescente
- derivata prima minore di 0 → f strettamente decrescente

La convessità/concavità, invece, sono legate al segno della derivata seconda, con f che ha:

- derivata seconda maggiore di 0 → f strettamente convessa
- derivata seconda minore di 0 → f strettamente concava

Dati i giusti richiami teorici, il *<u>metodo di bisezione</u>* consiste nell'applicazione del teorema degli zeri di funzioni continue, si assume che f ∈ C[a, b], $f(a)f(b) < 0$. Sappiamo intuitivamente che "uno zero sta qui dentro", aggiungendo inoltre che grazie anche alle successive derivate, per una serie di teoremi, dimostro l'unicità effettiva della soluzione, iterando nell'intervallo [a,b]:



L'idea quindi è di calcolarsi il punto medio e se $f(x_0)=0$ siamo su uno zero. Altrimenti avendo $f(a_0)$ e $f(b_0)$ hanno segno definito, $f(x_0)$ sarà discorde con uno solo dei due e quindi sicuramente ci sarà uno zero in (a_0, x_0) se $f(a_0)f(x_0) < 0$

altrimenti ci sarà uno zero in (x_0, b_0) visto che $f(x_0)f(b_0) < 0$ (sempre per il teorema degli zeri). Questo processo infinito, in generale, permette di costruire 3 successioni a_n, b_n, x_n tali che:

- $\exists\ \xi : f(\xi) = 0,\ \xi \in (a_n, b_n)$
- $|\,\xi - a_n\,|,\ |\,\xi - b_n\,| \leq b_n - a_n = \frac{b-a}{2^n}$
- $|\,\xi - x_n\,| < \frac{b_n - a_n}{2} = \frac{b-a}{2^{n+1}}$

La distanza dal punto medio, si vede facendo un disegno, non può superare la metà della lunghezza dell'intervallo disegnato $[a_n, b_n]$. Iterando miglioriamo l'errore, intuendo la correttezza e l'idea di calcolo grazie al teorema dei due carabinieri, sapendo inoltre che le 3 successioni convergono ad uno zero ξ.

Tecnicamente quindi:

Infatti

$$0 \leq |\,\xi - a_n\,|,\ |\,\xi - b_n\,| < \frac{b-a}{2^n} \underset{n\to\infty}{\longrightarrow} 0$$

e per il teorema dei 2 carabinieri

$$|\,\xi - a_n\,|,\ |\,\xi - b_n\,| \longrightarrow 0,\ n \to \infty$$

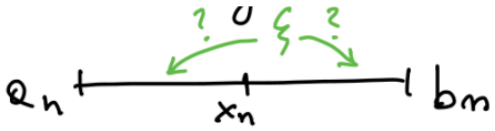
analogamente

$$|\,\xi - x_n\,| \longrightarrow 0,\ n \to \infty$$

visto che

$$0 \leq |\,\xi - x_n\,| < \frac{b-a}{2^{n+1}}$$

Quest'ultima diseguaglianza è immediatamente comprensibile da questo disegno



Rappresenta quindi l'idea intuitiva che ho descritto poco fa.
Volendo garantire una certa tolleranza ε nel calcolo approssimato del vero ξ, risolviamo:

$$e_n = |\,\xi - x_n\,| < \frac{b-a}{2^{n+1}} \leq \varepsilon$$

in modo che $x_n \in (\xi - \varepsilon, \xi + \varepsilon)$, ovvero

$$2^{n+1} \geq \frac{b-a}{\varepsilon}$$

cioè

$$\begin{aligned} n + 1 &\geq \log_2 \frac{b-a}{\varepsilon} \\ &= log_2\,(b-a) + log_2\left(\frac{1}{\varepsilon}\right) \end{aligned}$$

Questa disuguaglianza decide a priori a quale iterazione fermarsi garantendo la tolleranza prendendo:

$$n(\varepsilon) = [\log_2\left(\frac{1}{\varepsilon}\right) + log_2(b-a)] \quad \leftarrow \text{parte intera}$$

Una stima data dal tipo $e(n)\ <=\ stima(n)$ è usualmente chiamata *stima a priori*. Spesso questo tipo di stima è definito sovrastima, non vicina all'errore effettivo ma ne dà solo un confine superiore.
Per avere una stima dell'errore più aderente, facciamo queste osservazioni:

Scritto da Gabriel

$$f(x_n) \to f(\xi) = 0, \quad n \to \infty$$

Quella che stiamo usando qui in realtà è una caratterizzazione della continuità di una funzione in analisi matematica: f è continua in l se e solo se

$$\forall \{x_n\} : \lim_{n\to\infty} x_n = l \text{ si ha } \lim_{n\to\infty} f(x_n) = f(l)$$

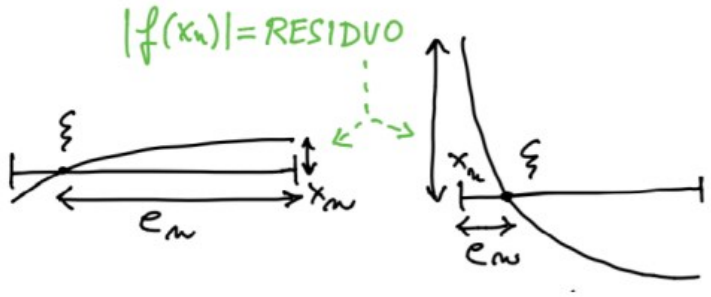
dicendo che "f è continua se e solo se il limite si può trasportare "dentro" la funzione".
Nel nostro caso $f(\xi) = 0$ quindi $f(x_n) \to 0$, $n \to \infty$ e anche $|f(x_n)| \to 0$, $n \to \infty$.
La quantità $|f(x_n)|$ si chiama "RESIDUO" perchè dice quanto "resta" ad f per annullarsi.
Viene allora spontanea questa domanda: siccome $f(x_n) \to 0$, $n \to \infty$, possiamo arrestare il processo di calcolo quando il residuo $|f(x_n)|$ è piccolo? In altre parole

$$|f(x_n)| \leq \varepsilon \overset{?}{\Longrightarrow} e_n \leq \varepsilon$$

La risposta è NO, in realtà

$$|f(x_n)| \leq \varepsilon \not\Rightarrow e_n \leq \varepsilon$$

Il succo di questa cosa sopra è il fatto di capire che la grandezza del residuo non è in sé un buon indicatore dell'errore, ma va opportunamente "pesata". Per capirlo, vediamo i grafici seguenti:



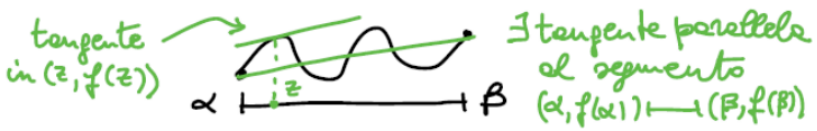
avendo nel primo caso un errore grande (con un residuo piccolo, quindi *sottostima* dell'errore), nel secondo caso l'errore è piccolo (con residuo grande, quindi *sovrastima* dell'errore).
Importante dire che una *sottostima* dell'errore è molto pericolosa perché potremmo fermare le iterazioni quando la successione non è ancora vicina alla tolleranza, avendo poi un *errore più grande della tolleranza.*
Una *sovrastima* invece genera un *incremento del costo computazionale*.
L'idea quindi fa notare che il residuo sottostima la funzione quando è "piatta", mentre la sovrastima quando la funzione è "ripida", variazione veloce intorno allo zero.
Ricordiamo il *teorema del valor medio/di Lagrange*, utile per introdurre il concetto di intorno pesato, quindi approssimabile:

TEOREMA (del valor medio)

Sia $f \in C[\alpha, \beta]$ derivabile in (α, β) allora

$$\exists\, z \in (\alpha, \beta) \text{ tale che } \frac{f(\beta)-f(\alpha)}{\beta-\alpha} = f'(z)$$

Interpretazione geometrica



Tornando all'analisi del residuo nel metodo di bisezione, mettiamoci nelle seguenti ipotesi: $f \in C^1[a,b]$ (cioè f è derivabile con derivata prima continua in $[a,b]$), $\{x_n\} \subset [c,d] \subseteq [a,b]$ (almeno per $n \geq n_0$ cioè per n abbastanza grande) con

$$x_n \to \xi,\ n \to \infty,\ f(\xi) = 0 \text{ e } f'(x) \neq 0 \quad \forall x \in [c,d]$$

allora vale la rappresentazione:

$$e_n = |\, x_n - \xi \,| = \frac{|\, f(x_n) \,|}{|\, f'(z_n) \,|}, \quad n \geq n_0$$

Osserviamo prima di dimostrare questa stima che:

- la rappresentazione dell'errore mostra che *l'errore è un residuo pesato*
- l'ipotesi che la derivata sia diversa da zero dice che lo zero è semplice, ovvero diversa da 0 quando calcolata in ξ. Diciamo quindi che i limiti delle successioni contenute, per il teorema della permanenza del segno:

  $$\exists\, \delta > 0 : f'(x) \neq 0 \quad \forall x \in [\xi - \delta, \xi + \delta] = [c,d]$$

  e siccome $x_n \to \xi,\ n \to \infty \quad \exists\, n_0$ tale che $|x_n - \xi| \leq \delta \quad \forall n \geq n_0$
  Si noti che $f'(z_n) \neq 0,\ n \geq n_0$ perchè $z_n \in int(x_n, \xi) \subset [c,d]$.
  È importante osservare che la rappresentazione dell'errore come residuo pesato non vale solo per il metodo di bisezione, ma per ogni metodo convergente a uno zero semplice se $f \in C^1$ (applicheremo infatti questo risultato più avanti al metodo di Newton)

  Significa umanamente parlando che la successione sta dentro all'intervallo e che ci sta anche il limite della successione stessa, perché intervallo chiuso.
- dalla rappresentazione possiamo ricavare *stime a posteriori* dell'errore (utilizzando il residuo dopo aver prodotto x_n tramite il calcolo).

<u>Dimostrazione</u> della rappresentazione usando il teorema del valor medio, e supponendo che $x_n > \xi$ (l'altro caso è del tutto analogo), con $\alpha = \xi$, $\beta = x_n$:

$$f(x_n) - f(\xi) = f'(z_n)(x_n - \xi),\ z_n \in (\xi, x_n)$$

con $f(\xi) = 0$, cioè

$$|\, f(x_n) \,| = |\, f'(z_n) \,|\, |\, x_n - \xi \,|$$

che si può riscrivere come

$$e_n = |\, x_n - \xi \,| = \frac{|\, f(x_n) \,|}{|\, f'(z_n) \,|}$$

vedendo la successione come residuo ed eseguendo il calcolo.

Il teorema del valor medio non dice chi sia z_n ma solo che esiste almeno un z_n nell'intervallo. Ricaviamo quindi stime "pratiche" dell'errore, usando il residuo opportunamente pesato.

i) Se è noto che $|\, f'(x) \,| \geq k > 0\ \forall x \in [a,b]$ (ma basta $\forall x \in [c,d]$), allora

$$e_n = \frac{|\, f(x_n) \,|}{|\, f'(z_n) \,|} \underset{\substack{\uparrow \\ \text{stima} \\ \text{rigorosa}}}{\leq} \frac{|f(x_n)|}{k}$$

definita come "stima rigorosa" perché mi fermo sotto la soglia di tolleranza.

ii) Se f' è nota o calcolabile, siccome $z_n \to \xi, n \to \infty$ per il teorema dei 2 carabinieri visto che z_n sta fra ξ e x_n, per la continuità di f' si ha che

$$|f'(x_n)|, |f'(z_n)| \underset{n\to\infty}{\longrightarrow} |f'(\xi)| \neq 0$$

Quindi, almeno per n abbastanza grande, $|f'(x_n)|$ e $|f'(z_n)|$ saranno entrambi dell'ordine di grandezza di $|f'(\xi)|$ (notiamo che nel residuo pesato quello che interessa è essenzialmente l'ordine di grandezza del peso $|f'(z_n)|$: per fissare le idee, sto dividendo per 100 o per $\frac{1}{100}$? Cioè, sto "aggiustando" una sovrastima o una sottostima?).
Abbiamo quindi una "STIMA EMPIRICA":

$$e_n = |x_n - \xi| \approx \frac{|f(x_n)|}{|f'(x_n)|}$$

valida almeno per $n \geq \bar{n}$, dove $\bar{n}$ corrisponde ad un controllo empirico che l'ordine di grandezza di $|f'(x_n)|$ si stia "stabilizzando", cioè valga:

$$\left| \frac{|f'(x_n)|}{|f'(x_{n-1})|} - 1 \right| \leq \delta$$

iii) Se invece *f'* non è vuota esplicitamente, va in qualche modo approssimata. Non sempre abbiamo una formula analitica di calcolo, perché *f* può provenire da altri calcoli, senza avere espressioni esplicite.
Se però sapessimo che *f* è derivabile, approssimiamo *f'* ad un rapporto incrementale:

$$f'(z_n) \approx \vartheta_n = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}} \overset{\text{VALOR MEDIO}}{=} f'(\mu_n)$$

dove $int(x_n, x_{n-1}) \ni \mu_n \to \xi, n \to \infty$ e quindi

$$|\vartheta_n| \approx |f'(\xi)| \approx |f'(z_n)|$$

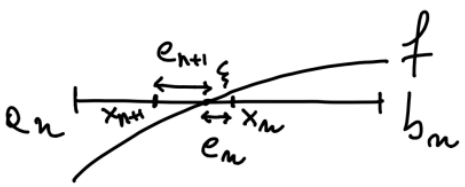
almeno per n abbastanza grande: di nuovo, possiamo usare un criterio empirico per "accettare" il valore di $|\vartheta_n|$ come peso, tipo $\left| \frac{|\vartheta_n|}{|\vartheta_{n-1}|} - 1 \right| \leq \delta$

Detto k_n il peso calcolato con uno degli approcci da 1 a 3 appena mostrati, siamo in grado di scrivere un *test di arresto* per il metodo di bisezione che combina stima *a priori* con la stima *a posteriori*.

$$min\left\{ \frac{b-a}{2^{n+1}}, \frac{|f(x_n)|}{k_n} \right\} \leq \varepsilon$$

Facciamo alcune considerazioni di carattere computazionale:

- il metodo di bisezione funziona con richieste analitiche e computazionali minime. La versione base con la stima a priori chiede solo che siano soddisfatte le ipotesi del teorema degli zeri (continuità e cambio segno agli estremi) e unicamente la possibilità di calcolare correttamente il segno di $f(x_n)$ (per cui è sufficiente un errore relativo < 100%). Infatti in generale se:
  $$\tilde{\alpha} \approx \alpha \neq 0 \quad \text{e} \quad \frac{|\alpha - \tilde{\alpha}|}{|\alpha|} < 1 \quad \Longrightarrow \quad sign(\tilde{\alpha}) = sign(\alpha)$$
- mentre la stima a priori è decrescente, l'errore effettivo invece no in generale, come si vede da:

  dove la stima del residuo pesato è tendenzialmente più accurata, dimezzandosi "in media" su una parte di iterazioni.

Vediamo quindi l'esempio del calcolo di √2 alla precisione di macchina.
Consideriamo l'equazione algebrica $f(x) = x^2 - 2 = 0$
Supponendo poi la funzione all'interno dell'intervallo C, derivabile infinite volte in R con derivate tutte continue, possiamo applicare il metodo di bisezione:

$$f(a) = f(1) = 1 - 2 = -1 < 0$$
$$f(b) = f(2) = 2^2 - 2 = 2 > 0$$

Inoltre $f'(x) = 2x \geq 2 \quad \forall x \in [1,2]$ quindi $\sqrt{2} \in (1,2)$ ed è l'unico zero in tale intervallo (in effetti sappiamo che è l'unico zero in $\mathbb{R}^+$) possiamo applicare il metodo di bisezione che comincia in questo modo:

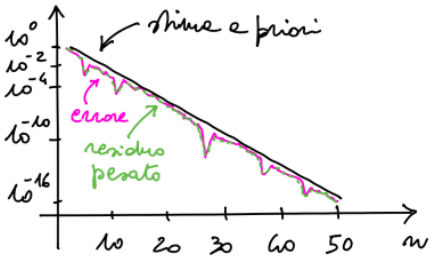
$$x_0 = \frac{1+2}{2} = 1.5,$$
$$x_1 = \frac{1+1.5}{2} = 1.25,$$
$$x_2 = \frac{1.25+1.5}{2} = \frac{2.75}{2} = 1.375,$$
$$x_3 = \frac{1.375+1.5}{2} = 1.4375,$$

Nel grafico sottostante (in scala log) riportiamo l'errore effettivo, la stima a priori

$$\frac{(b_n - a_n)}{2} = \frac{1}{2^{n+1}}$$

e la stima a posteriori

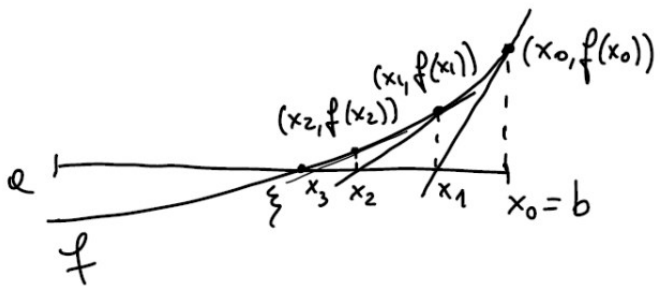
$$\frac{|f(x_n)|}{k} = \frac{|x_n^2 - 2|}{2}$$



vedendo che l'errore "in media" segue l'andamento della stima a priori, la sovrastima a volte di vari ordini di grandezza (picchi di errore verso il basso). D'altra parte la stima del residuo pesato è praticamente sovrapposta all'errore effettivo.

Lezione 9: Metodo di Newton (tangenti), convergenza e velocità di convergenza: parte 1

Come visto il metodo di bisezione è abbastanza lento e, come visto, la stima a priori cade in media ad ogni iterazione di un fattore 1/2; lo stesso fattore si ha pure nel caso di errore relativo.
Affinché l'errore relativo scenda di 1/10 abbiamo bisogno almeno di 3-4 iterazioni; con 50 iterazioni si arriva alla precisione di macchina..
Useremo un metodo molto più efficace, cioè il *metodo di Newton/metodo delle tangenti*, usando le tecniche del calcolo differenziale, pagando un prezzo dal punto di vista di richieste analitiche. L'idea del metodo è di linearizzare iterativamente l'equazione f(x)=0, sostituendo f ad ogni iterazione con la retta tangente del grafico, purché f sia derivabile, come visibile da:



Per trovare l'espressione analitica, calcoliamo lo zero della retta tangente nel punto x_0, $f(x_0)$:

Scritto da Gabriel

$$\begin{cases} y = 0 & \text{asse } x \\ y = f(x_0) + f'(x_0)(x - x_0) & \text{retta tangente} \end{cases}$$

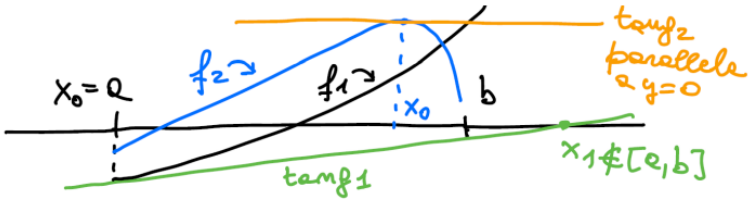
dove stiamo usando l'interpretazione geometrica della derivata nel punto x_0 come coefficiente angolare della retta tangente nel corrispondente punto del grafico di f. Otteniamo l'equazione

$$0 = f(x_0) + f'(x_0)(x - x_0)$$

la cui soluzione è

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}, \quad f'(x_0) \neq 0$$

È essenziale anche la posizione del valore iniziale in quanto, se scritto male, potrebbe far uscire dall'intervallo di definizione:



In generale si cerca di ottenere l'intersezione della tangente:

$$\begin{cases} y = 0 \\ y = f(x_n) + f'(x_n)(x - x_n) \end{cases}$$

ottenendo la formula iterativa

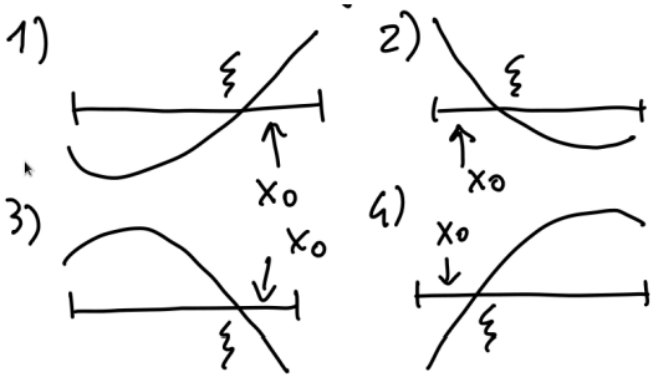
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0, 1, 2, \ldots$$

dando noi un set di condizioni sufficienti, garantendo quindi la convergenza, con ipotesi di tipo geometrico, garantendo una convergenza globale (quindi importante che x_0 sia nella zona giusta dell'intervallo).
Diamo poi il teorema di *convergenza del metodo di Newton con f'' segno costante:*

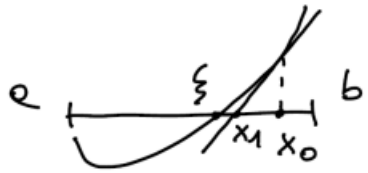
Sia $f \in C^2[a,b]$ (derivabile 2 volte con derivate continue in $[a,b]$), $f(a)f(b) < 0$, $f''(x) > 0 \quad \forall x \in [a,b]$ (oppure $f''(x) < 0 \quad \forall x \in [a,b]$), x_0 tali che $f(x_0)f''(x_0) > 0$ allora

il metodo di Newton è ben definito (cioè $f'(x_n) \neq 0 \quad \forall n$)
e converge all'unico zero ξ di f in (a,b)

Dimostrazione:
Abbiamo 4 casi possibili in base al segno di f'', dando per (1) e (2) una f strettamente convessa, in (3) e (4) f concava, x_0 scelto tra (ε, b] in (1) e (3), x_0 scelto tra [a, ε) in (2), (4).



Non è escluso che la derivata cambi segno, ma l'importante è che sia *la derivata seconda a non cambiare di segno.*
Trattando il caso, con gli altri casi la dimostrazione è totalmente analoga.



con $f(a) < 0$, $f(b) > 0$, $f''(x) > 0$, $\forall x \in [a, b]$, $x_0 \in (\xi, b]$
La dimostrazione si fa per induzione, mostrando che se $x_n \in (\xi, b]$ anche $x_{n+1} \in (\xi, b]$ e inoltre $x_{n+1} < x_n$
Infatti

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Ma se $x_n \in (\xi, b]$ allora $f'(x_n) > 0$ e $f(x_n) > 0$, quindi x_{n+1} si ottiene da x_n sottraendo una quantità > 0, cioè $x_{n+1} < x_n$.
D'altra parte f è strettamente convessa, il che è equivalente a dire che la tangente sta "sotto al grafico" $\forall x \in [a, b]$.
Ma allora la tangente in un punto $\in (\xi, b]$ interseca l'asse x a destra di ξ, cioè se $x_n \in (\xi, b]$ anche $x_{n+1} \in (\xi, b]$.
In definitiva, abbiamo provato che la successione $\{x_n\}$ è decrescente e che $x_n > \xi$ $\forall n$
Dall'analisi matematica è noto che una successione monotona e limitata ha limite e che il limite è $sup\{x_n\}$ se è crescente e $inf\{x_n\}$ se è decrescente (che corrisponde al nostro caso).
Quindi

$$\exists \lim_{n\to\infty} x_n = inf\{x_n\} = \eta \quad \text{con} \quad \eta \geq \xi$$

La lettera η corrisponde ad "eta", mentre la lettera ξ corrisponde a "csi"

$$\begin{aligned}\eta &= \lim x_{n+1} \\ &= \lim \left(x_n - \frac{f(x_n)}{f'(x_n)} \right) \\ &= \lim x_n - \lim \frac{f(x_n)}{f'(x_n)} \\ &= \lim x_n - \frac{\lim f(x_n)}{\lim f'(x_n)} \\ &= \lim x_n - \frac{f(\lim x_n)}{f'(\lim x_n)} \\ &= \eta - \frac{f(\eta)}{f'(\eta)}\end{aligned}$$

dove abbiamo usato le proprietà dei limiti e la continuità di f ed f' (portando il limite "dentro le funzioni"). Quindi

$$\eta = \eta - \frac{f(\eta)}{f'(\eta)} \quad \text{con } f'(\eta) \neq 0 \Longrightarrow \frac{f(\eta)}{f'(\eta)} = 0 \Longrightarrow f(\eta) = 0$$

Di fatto η convergerà a ξ in quanto lo 0 è unico, avendo dunque ξ che equivale a *inf* e *sup* per il calcolo del limite della successione.

Ma allora $\eta = \xi$, perché nelle ipotesi fatte (teorema degli zeri e f'' di segno costante) lo zero è unico, quindi il metodo di Newton è ben definito e $\{x_n\}$ converge a ξ.
Infine, osserviamo che anche nel caso (3), con $x_0 \in [\xi, b)$ si ottiene

$$\xi = \inf\{x_n\} = \lim x_n$$

mentre nei casi (2) e (4) con $x_0 \in [a, \xi)$ si ottiene

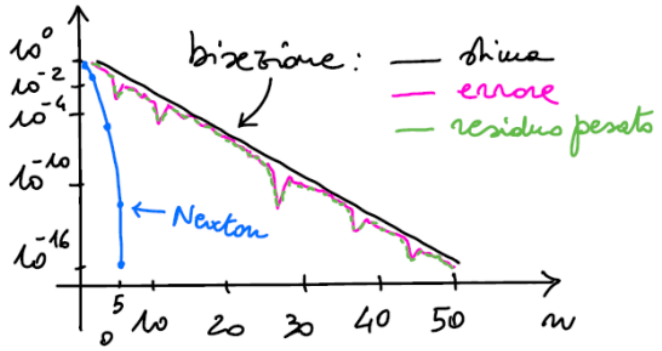
$$\xi = \sup\{x_n\} = lim x_n$$

Oltre al discorso della convergenza anche con condizioni meno forti (chiamata "*locale*"), l'altro punto di forza è la *velocità di convergenza*. Calcoliamo quindi radice di 2 con la bisezione e con Newton.
Qui sono soddisfatte le ipotesi del teorema degli zeri, avendo inoltre derivata prima finita e derivata seconda positiva. Abbiamo quindi una sequenza di iterazioni sapendo che in doppia precisione:

$$fl(\sqrt{2}) = 1.414213562373095$$

con il numero di cifre decimali corrette che aumenta di 1 ogni 3/4 iterazioni, avendo poi con 50 iterazioni almeno 16 cifre corrette.
Tutte le cifre corrette sono ricavate dalle tangenti.

$$\begin{aligned}x_0 &= \boxed{1},5; \\ x_1 &= \boxed{1},25; \\ x_2 &= \boxed{1},375; \\ x_3 &= \boxed{1,4}375; \\ x_4 &= \boxed{1,4}0625; \\ x_5 &= \boxed{1,4}21875; \\ x_6 &= \boxed{1,41}40625; \\ &\dots \\ x_{10} &= \boxed{1,414}55078125; \\ &\dots \\ x_{50} &= fl(\sqrt{2})\end{aligned}$$



Di fatto il numero di cifre decimali corrette <u>raddoppia</u> ad ogni iterazione, mostrando che Newton converge più che esponenzialmente. Dal punto di vista dei grafici di errore, abbiamo un grafico *superesponenziale*, con una curva che va rapidamente alla precisione di macchina, rispetto invece ad una convergenza esponenziale:

Vediamo quindi come decade la relazione di errore. Diamo inoltre il *<u>teorema della velocitò di convergenza del metodo di Newton:</u>*

Sia $f \in C^2[a,b]$ e si assuma di essere in ipotesi che garantiscano la convergenza del metodo di Newton e $\xi \in [a,b] : f(\xi) = 0$; sia inoltre $\{x_n\} \subset [c,d] \subseteq [a,b]$ con $f'(x) \neq 0 \ \forall x \in [c,d]$ allora

$$e_{n+1} \leq c\, e_n^2, \quad n \geq 0, \quad c = \frac{1}{2} \cdot \frac{M_2}{m_1}$$

$$\text{con } M_2 = \max_{x\in[c,d]} |\, f''(x)\,|, \quad m_1 = \min_{x\in[c,d]} |\, f'(x)\,| > 0$$

Prima di dimostrare diciamo che:

i) l'ipotesi $\{x_n\} \subset [c,d]$ con $f'(x) \neq 0$ in $[c,d]$, come abbiamo già visto nell'analisi del residuo pesato col metodo di bisezione, ci assicura che lo zero ξ è semplice, cioè $f'(\xi) \neq 0$

ii) tale ipotesi è soddisfatta ad esempio nelle condizioni del teorema di convergenza dimostrato prima con $[c,d] = [\xi, b]$ nei casi 1) e 3), $[c,d] = [a,\xi]$ nei casi 2) e 4).

<u>Dimostrazione</u>

Usando poi Taylor centrata in x_0 con resto del secondo ordine in forma di Lagrange:

$$f(\xi) = f(x_n) + f'(x_n)(\xi - x_n) + \frac{f''(z_n)}{2}(\xi - x_n)^2$$

dove $z_n \in int(x_n, \xi) \subset [c,d]$ e $f(\xi) = 0$, da cui

$$-\frac{f(x_n)}{f'(x_n)} = \xi - x_n + \frac{f''(z_n)}{2f'(x_n)}(\xi - x_n)^2$$

<u>Importante</u>: Vianello dice di mettere i moduli nell'esecuzione delle stime dato che è una cosa spesso sbagliata agli esami.

Ma dalla definizione del metodo

$$-\frac{f(x_n)}{f'(x_n)} = x_{n+1} - x_n$$

che inserita nella formula di Taylor porta a

$$x_{n+1} - x_n = \xi - x_n + \frac{f''(z_n)}{2f'(x_n)}(\xi - x_n)^2$$

ovvero mettendo i moduli

$$e_{n+1} = |\, x_{n+1} - \xi\,| = c_n e_n^2 \quad \text{con} \quad c_n = \frac{1}{2}\frac{|\, f''(z_n)\,|}{|\, f'(x_n)\,|}$$

La successione poi è limitata, si vede, poi applicando Weierstrass (quest'ultimo dice che se la funzione è continua esiste almeno un punto di massimo/minimo assoluto nell'intervallo [a,b]):

Scritto da Gabriel

$$|f''(z_n)| \leq \max_{x\in[c,d]} |f''(x)| = M_2$$

applicando il teorema di Weierstrass sull'esistenza di massimo e minimo assoluti a $|f''(x)| \in C[c,d]$.
D'altra parte, applicando lo stesso teorema a $|f'(x)| \in C[c,d]$ abbiamo che

$$m_1 = \min_{x\in[c,d]} |f'(x)| > 0$$

(perché $\exists\, \overline{x} : m_1 = |f'(\overline{x})|$ e $f'(\overline{x}) \neq 0$)
Otteniamo quindi $|f'(x_n)| \geq m_1 > 0$ e infine $c_n \leq \frac{1}{2}\frac{M_2}{m_1} = c$.

■

Operiamo poi il confronto con la bisezione, avendo una relazione di tipo *quadratico, con un errore* al passo n+1 *maggiorato* da una quantità *proporzionale al quadrato dell'errore al passo n*.

La relazione è: $\boxed{e_{n+1} \leq ce_n^2}$

osservando che:

$$ce_{n+1} \leq c \cdot c \cdot e_n^2 = (ce_n)^2$$

successivamente:

Applicando la disuguaglianza $ce_{n+1} \leq (ce_n)^2$ *per* $n \geq \overline{n}$:

$$\begin{aligned}
&ce_{\overline{n}+1} \leq (ce_{\overline{n}})^2 \leq \theta^2 \\
&ce_{\overline{n}+2} \leq (ce_{\overline{n}+1})^2 \leq \left(\theta^2\right)^2 = \theta^4 \\
&ce_{\overline{n}+3} \leq (ce_{\overline{n}+2})^2 \leq \left(\theta^4\right)^2 = \theta^8 \\
&\quad \dots \\
&ce_{\overline{n}+k} \leq (ce_{\overline{n}+k-1})^2 \leq \left(\theta^{2^{k-1}}\right)^2 = \theta^{2^k}
\end{aligned}$$

Confrontiamo quindi con il metodo di bisezione e, prendendo θ=1/2.

Otteniamo, dopo k iterazioni di Newton a partire da $\overline{n}$

$$e_{\overline{n}+k}^{Newt} \leq \frac{1}{c} \cdot \left(\frac{1}{2}\right)^{2^k}$$

mentre con k iterazioni del metodo di bisezione

$$e_k^{bisez} \lesssim \left(\frac{1}{2}\right)^k e_0^{bisez}$$

Quindi il theta è stato scelto per fare un confronto con la bisezione, dimostrando che la differenza è molto grande data dalla crescita esponenziale dell'esponente presente, sapendo che si innesca una riduzione rapidissima dell'errore con quadrati successivi, detta convergenza quadratica.

Lezione 10: Metodo di Newton: convergenza locale, test d'arresto, esempi e metodi di linearizzazione (parte 2)

Partendo dal risultato dell'altra volta del metodo di Newton, fa capire che il metodo converge più che esponenzialmente.

Partiamo dalla relazione chiave ottenuta nella lezione precedente per l'errore del metodo di Newton:

$$e_{n+1} = c_n \cdot e_n^2 \leq c e_n^2$$

$$c_n = \frac{1}{2} \cdot \frac{|f''(z_n)|}{|f'(x_n)|}, \; c = \frac{1}{2} \cdot \frac{M_2}{m_1}$$

con $M_2 = \max|f''(x)|$ e $m_1 = \min|f'(x)| > 0$ dove $x \in [c,d]$ assumendo che il metodo sia convergente, $f \in C^2[a,b]$ e che $\{x_n\} \subset [c,d] \subseteq [a,b]$ con $f'(x) \neq 0 \; \forall x \in [c,d]$
Da questa abbiamo ottenuto, fissato $\theta \in (0,1)$ e preso $\bar{n}$ tale che

$$ce_n \leq \theta < 1 \quad \forall n \geq \bar{n} \quad \Longrightarrow \quad e_{\bar{n}+k} \leq \frac{1}{c}\theta^2, \quad k \geq 0$$

Cosa succede invece quando x_0 vale $ce_0 < 1$?
In questo caso avremmo la disuguaglianza

$$ce_n \leq (ce_0)^{2^n}$$

con $n \geq 0$. Infatti

$$ce_1 \leq (ce_0)^2$$
$$ce_2 \leq (ce_1)^2 \leq (ce_0)^4$$
$$\vdots$$
$$ce_n \leq (ce_{n-1})^2 \leq (ce_0)^{2^n}$$

Questo ci fa intuire che se $ce_0 < 1$ cioè

$$e_0 = |x_0 - \xi| < \frac{1}{c}$$

cioè se prendiamo x_0 in un intorno opportuno di ξ zero di f avremo la convergenza con le sole ipotesi che $f \in C^2$ e che $f'(x) \neq 0$ in quell'intorno, perché

$$ce_0 < 1 \quad \Rightarrow \quad (ce_0)^{2^n} \to 0, n \to \infty \quad \Rightarrow \quad e_n \to 0, n \to \infty$$

Di fatto non entriamo nei dettagli della dimostrazione della cosa qui sopra, infatti si dimostra che $1/c$ sia decrescente e ci sarà convergenza con la precedente.

2.3.1 TEOREMA (convergenza locale del metodo di Newton)

Sia ξ zero di f ed $\exists \delta > 0 : f \in C^2(I_\delta)$ e $f'(x) \neq 0 \quad \forall x \in I_\delta$, dove $I_\delta = [\xi - \delta, \xi + \delta]$;
inoltre sia $x_0 \in (\xi - \gamma, \xi + \gamma)$ con $\gamma = \min\left\{\delta, \frac{1}{c}\right\}$ dove

$$c = \frac{1}{2} \cdot \frac{\max|f''(x)|}{\min|f'(x)|}, \quad x \in I_\delta$$

allora

$$\forall n \geq 0 \quad x_n \in I_\gamma \quad \text{e} \quad e_n \leq \frac{1}{c}(ce_0)^{2^n} \underset{n\to\infty}{\longrightarrow} 0$$

Il metodo quindi è veloce e basta che la funzione sia di classe 2 e abbia f' ≠ 0 in un intorno di ε, sapendo che è sempre soddisfatta la seconda ipotesi del set di condizioni se lo 0 è semplice.

Diamo quindi la definizione precisa di ordine di convergenza:

Dato un metodo che produce una successione $\{x_n\}_{n\geq 0}$ tale che $\lim_{n\to\infty} x_n = l$ con l limite finito si dice che:

1. il metodo ha ordine di convergenza almeno $p \geq 1$ se $\exists c > 0$ con $(c \in (0,1)$ se $p = 1)$ tale che
$$e_{n+1} \leq c e_n^p \quad \forall n$$
2. il metodo ha ordine di convergenza esattamente $p \geq 1$ se $\exists L > 0$ (con $L \in (0,1)$ se $p = 1$) tale che:
$$\lim_{n\to\infty} \frac{e_{n+1}}{e_n^p} = L$$
(dove L è spesso chiamata costante asintotica del metodo).
La convergenza è detta lineare se $p = 1$, superlineare se $p > 1$ e in particolare quadratica se $p = 2$, cubica se $p = 3, \ldots$

Nel caso p=1 abbiamo una convergenza almeno lineare e quindi resta minore di 1. Analogamente, sapendo che la derivata seconda ha ordine avendo limite:

Analogamente, se $\lim_{n\to\infty} \frac{e_{n+1}}{e_n} = L$ con $L \in (0,1)$, preso $\overline{\varepsilon} \in (0, 1-L)$ $\exists \overline{n}$ tale che

$$0 \leq \frac{e_{n+1}}{e_n} \leq L + \overline{\varepsilon} = c < 1 \quad \forall n \geq \overline{n}$$

cioè $e_{n+1} \leq c e_n$ $\forall n \geq \overline{n}$, da cui ragionando come sopra $e_n \leq c^{n-\overline{n}} e_{\overline{n}} \to 0$, $n \to \infty$.

Se ci sta convergenza, necessariamente L<=1.

Infatti, se $\lim_{n\to\infty} \frac{e_{n+1}}{e_n} = L > 1$, preso $\overline{\varepsilon} \in (0, L-1)$ $\exists \overline{n}$ tale che $1 < L - \overline{\varepsilon} \leq \frac{e_{n+1}}{e_n}$ $\forall n \geq \overline{n}$, quindi $e_{n+1} \geq (L - \overline{\varepsilon}) \cdot e_n$ da cui

$$e_{\overline{n}+1} \geq (L - \overline{\varepsilon}) \cdot e_{\overline{n}}, \ldots, e_n \geq (L - \overline{\varepsilon})^{n-\overline{n}} e_{\overline{n}} \to \infty, n \to \infty$$

e l'errore divergerebbe, cioè non ci sarebbe convergenza.

Di fatto il metodo di Newton, per definizione di ordine di convergenza, permette di dire che l'ordine è almeno p=2 per zeri semplici, perché sappiamo che $e_{n+1} < c e_n^2$, infatti:

$$\frac{e_{n+1}}{e_n^2} = c_n = \frac{1}{2} \cdot \frac{|f''(z_n)|}{|f'(x_n)|} \to \frac{1}{2} \cdot \frac{|f''(\xi)|}{|f'(\xi)|}$$

Quindi sappiamo anche che la bisezione in media si comporta come un metodo di ordine $p = 1$ e costante asintotica $L = 1/2$. Per il metodo di Newton invece, quando lo zero non è semplice, l'ordine di convergenza di Newton scende a $p = 1$ con costante asintotica $1 - 1/m$, poi usando Taylor:

Ad esempio, se $m = 2$, cioè se $f'(\xi) = 0$ e $f''(\xi) \neq 0$ si ha

$$\frac{e_{n+1}}{e_n} = \frac{e_{n+1}}{e_n^2} \cdot e_n = c_n \cdot e_n = \frac{1}{2} \cdot \frac{|f''(z_n)|}{|f'(x_n)|} \cdot e_n$$

Usando la formula di Taylor

$$f'(x_n) = \underbrace{f'(\xi)}_{=0} + f''(\xi) \cdot (x_n - \xi) + o(e_n)$$

(dove $\alpha_n = o(\beta_n)$ significa che $\lim_{n\to\infty} \frac{\alpha_n}{\beta_n} = 0$), da cui

$$\frac{|f'(x_n)|}{e_n} \to |f''(\xi)|, \quad n \to \infty$$

e quindi

$$\frac{e_{n+1}}{e_n} = e_n \cdot c_n \to \frac{1}{2}, \quad n \to \infty$$

Significa quindi che l'errore decade di un fattore $1/2$ in maniera abbastanza lineare.
Spieghiamo quindi il calcolo del metodo di Newton applicato a √2
Quando si parla di cifre corrette, si parla dell'errore relativo come: $r_n = \frac{e_n}{|\xi|}, \quad \xi \neq 0$

Ora, da $e_{n+1} \leq ce_n^2$ otteniamo

$$r_{n+1} = \frac{e_{n+1}}{|\xi|} \leq c\frac{e_n^2}{|\xi|} = c|\xi| r_n^2$$

Dunque per c|ξ| <= 1 ha un errore relativo al passo n+1 maggiorato dal quadrato dell'errore relativo al passo n, quindi che raddoppia. Le iterazioni di Newton portano ad avere tutte le cifre corrette eseguendo le operazioni macchina insieme:

Per fissare le idee, se al passo k $r_k < 10^{-1}$, avremo $r_{k+1} < 10^{-2}$, $r_{k+2} < 10^{-4}$, $r_{k+3} < 10^{-8}$ e con $r_{k+4} < 10^{-16}$ siamo già sotto la precisione macchina.
Questo è proprio quanto succede con $f(x) = x^2 - 2$ in $[\sqrt{2}, 2]$, dove

$$c = \frac{1}{2}\frac{2}{\min\limits_{x\in[\sqrt{2},2]} 2x} = \frac{1}{2} \cdot \frac{2}{2\sqrt{2}} = \frac{1}{2\sqrt{2}}$$

$$c|\xi| = \frac{\sqrt{2}}{2\sqrt{2}} = \frac{1}{2} < 1$$

e

$$r_1 = \frac{1.5 - \sqrt{2}}{\sqrt{2}} \approx 0.06 < 10^{-1}$$

infatti $r_5 < 10^{-16} \Rightarrow x_5 = fl(\sqrt{2})$.

Occorre saper stimare l'ordine di grandezza di f ed f' almeno per n abbastanza grande, quindi col metodo di Newton diventa essenziale poter calcolare con accuratezza f(xn) e anche f'(xn), per eseguire correttamente la stima.
Comunque il metodo di Newton non ha una stima a priori facile per arrestare le iterazioni e per usarla bene dovremo stimare|f'|ma anche il max di|f''|.
Grazie a Newton possiamo sfruttare una semplice stima a posteriori, cosiddetto STEP $|x_{n+1} - x_n|$
Il metodo di Newton quindi ha una buona stima per costruzione, nonostante lo step vada a 0 per n → ∞, avendo sempre per Newton:

$$STEP(n) = |x_{n+1} - x_n| = \frac{|f(x_n)|}{|f'(x_n)|}$$

visto che

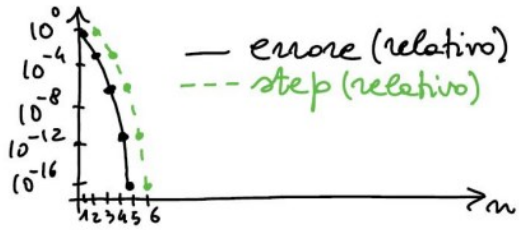
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Quindi per Newton lo step è per costruzione un residuo pesato, calcolato ad ogni iterazione. Ciò è dato dal teorema di convergenza globale (anche quello locale), dato che l'errore è decrescente (en+1 < en) stabilizzandosi per:

$$\frac{|f'(x_{n-1})|}{|f'(x_n)|} \approx 1$$

Questo approccio può essere utile come metodo ibrido ma affidabile, per fare alcune iterazioni partendo con un metodo veloce e adottando un controllo di convergenza, iterando il procedimento finché non entriamo in condizioni di convergenza rapida perché parte dell'intorno.
Siamo quindi interessati all'errore relativo, a meno che ξ = 0.
Per avere un test di arresto che funziona si usa di solito un test tipo: $stima(n) \leq \varepsilon_a + |x_n| \cdot \varepsilon_r$

La stima dello step quindi si comporta molto bene, come ad esempio nel calcolo di √2:



La curva dello step è parallela alla curva di errore, ma è slittata in avanti di 1, dovuto al calcolo del residuo pesato.

$$x_{n+1} - x_n = -\frac{f(x_n)}{f'(x_n)}$$

Siccome l'errore decresce velocemente, la stima dello step diventa ancora più affidabile

$$r_{n+1} = \frac{e_{n+1}}{|\xi|} \ll r_n = \frac{e_n}{|\xi|} \approx \frac{|x_{n+1} - x_n|}{|\xi|}$$

Seguono due esempi di applicabilità del metodo di Newton.

<u>Esempio 1: Metodo di Erone per le radici quadrate</u>

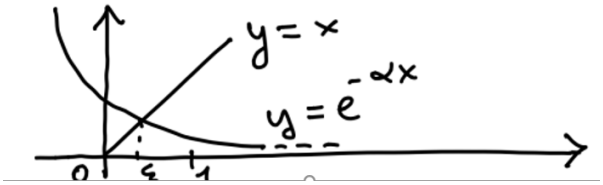
Abbiamo visto come usare il metodo di Newton per calcolare $\sqrt{2}$ come soluzione dell'equazione algebrica (zero di un polinomio) $f(x) = x^2 - 2 = 0$.
L'approccio è generalizzabile al calcolo di $\sqrt{a}$, $a > 0$, risolvendo l'equazione $x^2 - a = 0$.
Osserviamo che $f(0) = -a < 0$ e $f(b) > 0$ per $b^2 > a$.
Visto che $f'(x) = 2x$ e inoltre $f''(x) = 2 > 0$ siamo nelle ipotesi del teorema di convergenza globale scegliendo x_0: $x_0^2 > a$.
Qual è la forma delle iterazioni di Newton in questo caso?

$$\begin{aligned} x_{n+1} &= x_n - \frac{f(x_n)}{f'(x_n)} \\ &= x_n - \frac{x_n^2 - a}{2x_n} \\ &= \frac{2x_n^2 - x_n^2 + a}{2x_n} \\ &= \frac{x_n^2 + a}{2x_n} \\ &= \frac{x_n}{2} + \frac{a}{2x_n}, \quad n \geq 0 \end{aligned}$$

La sostanza è: si innesca una convergenza più che esponenziale e la cosa era già storicamente nota.

Esempio 2: Applicazione del metodo di Newton a un'equazione trascendente

Newton è applicabile quindi alle equazioni del tipo f(x)=0, osservando che anche le equazioni algebriche richiedono metodi numerici: è noto che gli zeri di polinomi di grado >= 5 non sono calcolabili tramite radicali e anche le stesse equazioni di secondo grado richiedono il calcolo di √Δ fatto con un metodo approssimato. Consideriamo l'equazione trascendente f(x) = x – e-αx = 0
Osserviamo che f(0) < 0 ed f(1) > 0, quindi esisterà un ξ tale per cui f(ξ)=0.
Avremo inoltre che f' > 0 mentre f'' < 0 (rispettivamente strettamente crescente e strettamente concava).



Inoltre

$$f'(x) \geq f'(1) = 1 + \alpha e^{-\alpha}$$

e

$$|f''(x)| = \alpha^2 e^{-\alpha x} \leq \alpha^2, \quad x \in [0,1]$$

vale quindi

$$c = \frac{1}{2}\frac{M_2}{m_1} \leq \frac{1}{2}\frac{\alpha^2}{1+\alpha e^{-\alpha}}$$

per $\alpha \leq 1$ abbiamo che

$$c\,|\xi| < \frac{1}{2}$$

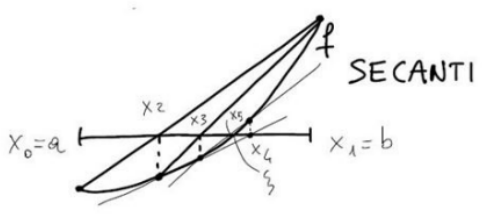
da cui otteniamo

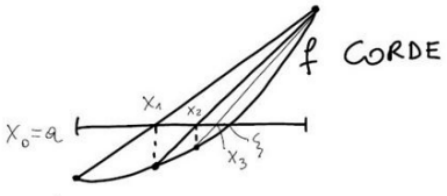
$$r_{n+1} = \frac{e_{n+1}}{|\xi|} < \frac{1}{2} r_n^2$$

e anche in questo caso ci sarà un raddoppio (almeno) delle cifre corrette ad ogni iterazione (per $\alpha = 1$, $x_5 = fl(\xi) = 0.5671432904097838$).

Lezione 11: Iterazioni di punto fisso, teorema delle contrazioni, convergenza locale, ordine di convergenza, Newton come iterazione di punto fisso

Partiamo enunciando il metodo delle corde (segmento che unisce due punti distinti di una curva) e delle secanti (cioè l'intersezione della retta tangente con l'asse delle ascisse), metodo che considera la derivata n-esima della funzione calcolata e utile per la linearizzazione nel calcolo di soluzioni numeriche, con calcolo della soluzione convergente. Andiamo quindi a scrivere semplicemente il coefficiente angolare, partendo appunto da:





Entrambi i metodi corrispondo a sostituire l'equazione $f(x) = 0$ con un'equazione lineare

$$f(x_n) + q_n(x - x_n) = 0$$

dove nel metodo delle corde q_n è il coefficiente angolare della corda (segmento) per $(x_n, f(x_n))$ e $(b, f(b))$

$$q_n = \frac{f(b) - f(x_n)}{b - x_n}$$

mentre nel metodo delle secanti q_n è il coefficiente angolare della retta per $(x_{n-1}, f(x_{n-1}))$ e $(x_n, f(x_n))$

$$q_n = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

(osserviamo che nel metodo di Newton $q_n = f'(x_n)$).

Entrambi i metodi hanno ipotesi di convergenza locale/globale, ma il metodo delle corde ha convergenza lineare, mentre quello delle secanti ha convergenza superlineare, avendo rispetto a Newton un rapporto incrementale (senza dover conoscere la derivata):

$$x_{n+1} = x_n - \frac{f(x_n)}{q_n}$$

in cui se per il teorema dei carabinieri converge, allora la secante si comporta come la tangente, dato dal valor medio e con estremo fisso:

con $n \geq 0$ (corde) e $n \geq 1$ (secanti), ma mentre nelle corde un estremo è fisso, nelle secanti per $f \in C^1$

$$q_n = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}} \overset{\text{valor medio}}{=} f'(u_n)$$

dimostrando poi che l'ordine di convergenza arriva alla sezione aurea, con convergenza rapida nel caso utile (avendo p > 1):

$\exists c' > 0$ tale che

$$c'e_n \leq (c'e_0)^{p^n}$$

Studiamo quindi equazione nella forma:

$$x = \phi(x), \ x \in I \subseteq \mathbb{R}$$

dove $\phi \in C(I)$ e I è un intervallo chiuso (non necessariamente limitato) di $\mathbb{R}$ e la loro soluzione numerica tramite semplici iterazioni del tipo

$$x_{n+1} = \phi(x_n), \ n \geq 0, \ x_0 \in I$$

comunemente dette "iterazioni di punto fisso".

Enunciamo quindi il "teorema delle contrazioni", descrivendo come contrazione "una funzione da uno spazio metrico in sé stesso tale che la distanza tra l'immagine di due elementi qualsiasi dello spazio sia inferiore alla distanza degli elementi stessi":

2.4.1 TEOREMA (esistenza e unicità del punto fisso e convergenza delle iterazioni di punto fisso per una contrazione)

Sia $\phi : I \subseteq \mathbb{R} \to \mathbb{R}$ una funzione derivabile nell'intervallo chiuso $I \subseteq \mathbb{R}$, tale che:

1. $\phi(I) \subseteq I$ cioè l'immagine di I tramite ϕ, $\phi(I) = \{y : y = \phi(x),\ x \in I\}$, è contenuta in I
2. $\exists \theta \in (0,1) : |\phi'(x)| \leq \theta \quad \forall x \in I$

allora

$$\exists! \, \xi \in I : \xi = \phi(\xi) \text{ (punto fisso) } \forall x_0 \in I,\ \xi = \lim_{n\to\infty} x_n$$

$$\text{dove } x_{n+1} = \phi(x_n),\ n \geq 0$$

dando come osservazioni:

1. l'intervallo I è assunto chiuso, ma può non essere limitato, cioè $I = [a,b]$ con $-\infty < a < b < +\infty$ ma anche $I = [a, +\infty)$ oppure $I = (-\infty, b]$ o addirittura $I = \mathbb{R}$

2. ϕ è una contrazione (di I in sé stesso), cioè contrae le distanze di un fattore $\theta < 1$. Infatti per il teorema del valor medio $\forall x, y \in I$

$$\phi(x) - \phi(y) = \phi'(z)(x - y), \quad z \in int(x, y)$$

da cui

$$|\phi(x) - \phi(y)| = |\phi'(z)||x - y| \leq \theta|x - y| < |x - y|$$

3. chiaramente la disuguaglianza appena provata, assunta direttamente come ipotesi, implica che ϕ è continua in I, infatti $\forall x, \overline{x} \in I$

$$0 \leq |\phi(x) - \phi(\overline{x})| \leq \theta|x - \overline{x}|$$

e quindi per il teorema dei 2 carabinieri

$$|\phi(x) - \phi(\overline{x})| \to 0,\ x \to \overline{x}$$

che è equivalente a dire che

$$\phi(x) \to \phi(\bar{x}),\ x \to \bar{x}$$

Dimostrazione

Essendo ϕ continua, dimostriamo l'esistenza di un punto fisso nell'intervallo [a,b] limitato, grazie al cambiamento di segno che garantisce che ci sia almeno un punto in esso contenuto:

Siccome ϕ è continua, tale è

$$f(x) = x - \phi(x)$$

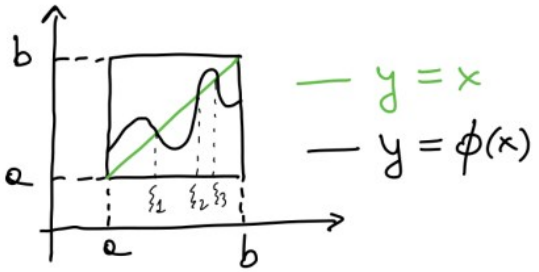
Se $a = \phi(a)$ oppure $b = \phi(b)$ allora a oppure b sono punto fisso.
Se invece $a \neq \phi(a)$ e $b \neq \phi(b)$ siccome $a \leq \phi(x) \leq b \; \forall x \in [a, b]$ si ha $a - \phi(a) < 0$ e $b - \phi(b) > 0$
cioè f è continua e cambia segno agli estremi

$$\Longrightarrow \exists \xi \in (a, b) : f(\xi) = 0$$

cioè

$$\exists \xi \in (a, b) : \xi = \phi(\xi)$$

La continuità non basta però a garantire l'unicità del punto fisso, come si vede da questo disegno



ma avendo ϕ contrazione, l'unicità è assicurata.
Infatti se $\exists \xi_1, \xi_2 \in I$ con $\xi_1 \neq \xi_2$ tali che $\xi_1 = \phi(\xi_1)$ e $\xi_2 = \phi(\xi_2)$ allora

$$|\xi_1 - \xi_2| = |\phi(\xi_1) - \phi(\xi_2)| \leq \theta |\xi_1 - \xi_2|$$

cioè $\theta \geq 1$ contro l'ipotesi che $\theta < 1$.
Resta da dimostrare che $\forall x_0 \in I$, definendo $x_{n+1} = \phi(x_n)$, $n \geq 0$ si ha $\lim_{n \to \infty} x_n = \xi$.
Ora

$$e_{n+1} = |x_{n+1} - \xi| = |\phi(x_n) - \phi(\xi)| \leq \theta |x_n - \xi| = \theta e_n$$

da cui

$$\begin{aligned} & e_1 \leq \theta e_0, \\ & e_2 \leq \theta e_1 \leq \theta^2 e_0, \\ & \quad \vdots \\ & e_n \leq \theta^n e_0 \to 0, \; n \to \infty \text{ perchè } \theta \in (0, 1) \end{aligned}$$

Diamo due osservazioni:
nel teorema delle contrazioni, si ha una successione convergente a ξ ∈ I (con I chiuso) e si ha che con la dimostrazione scritta sopra si ottiene la stima a priori dell'errore:

$$e_n \leq \theta^n e_0$$

avendo le successioni che convergono tutte allo stesso limite, che è l'unico punto fisso di ϕ in I, con convergenza almeno lineare.
si può anche avere una stima a posteriori, spesso più precisa della precedente, scrivendo (qui come al solito assumendo f sia derivabile):

$$x_{n+1} - \xi = x_{n+1} - x_n + x_n - \xi$$

ma

$$x_{n+1} - \xi = \phi(x_n) - \phi(\xi) \underset{\text{VALOR MEDIO}}{=} \phi'(z_n)(x_n - \xi), \quad z_n \in int(x_n, \xi)$$

da cui

$$\phi'(z_n)(x_n - \xi) = x_{n+1} - x_n + x_n - \xi$$

ovvero

$$(1 - \phi'(z_n))(x_n - \xi) = x_n - x_{n+1}$$

e passando ai moduli

$$\frac{|x_{n+1} - x_n|}{|1 - \phi'(z_n)|} = e_n \leq \frac{|x_{n+1} - x_n|}{1 - \theta} \quad \left(1^\circ \text{ stima a posteriori}\right)$$

perchè

$$|\phi'(z_n)| \leq \theta < 1 \Longrightarrow |1 - \phi'(z_n)| \geq |1 - |\phi'(z_n)|| \geq 1 - \theta > 0$$

e

$$\frac{1}{|1 - \phi'(z_n)|} \leq \frac{1}{1 - \theta}$$

Diciamo quindi che l'errore è stimato dallo step $|X_{n+1} - X_n|$ a meno del fattore $\frac{1}{1-\theta}$. Con θ piccolo, lo step preso da solo dà una buona stima dell'errore, mentre se θ è vicino ad 1 lo step va corretto per evitare sottostime. Facciamo poi un esempio, considerando un'equazione a punto fisso:

$$x = \phi(x) = e^{-\alpha x}, \quad \alpha > 0$$

se $\alpha < 1$,

$$|\phi'(x)| = |-\alpha e^{-\alpha x}| \leq \alpha < 1$$

e quindi ϕ contrae le distanze, cioè l'ipotesi (2) del teorema delle contrazioni è soddisfatta con $\theta = \alpha$; d'altra parte, $0 < \phi(x) \leq 1 \, \forall x \in [0, +\infty)$, quindi anche (1) è soddisfatta con $I = [0, +\infty)$. In realtà, siccome $\phi'(x) = -\alpha e^{-\alpha x} < 0$, ϕ è strettamente decrescente (e positiva), quindi visto che $\phi(0) = 1$ e $0 < \phi(1) = 1 - e^{-\alpha} < 1$ si ha

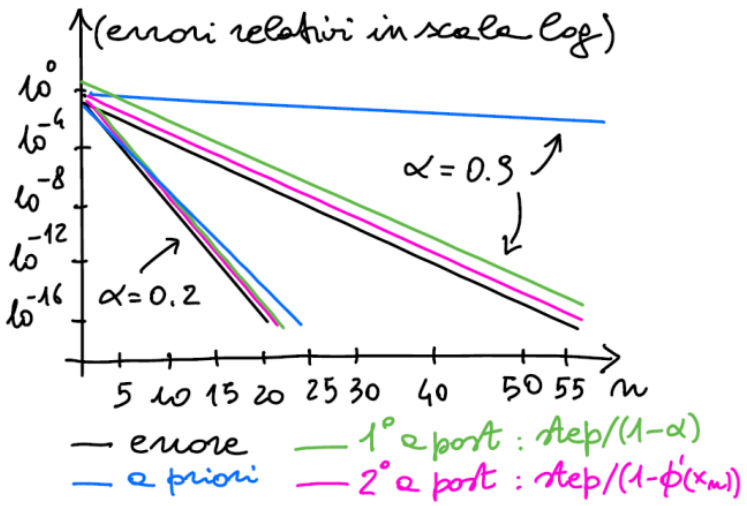
$$0 < \phi(x) \leq 1 \quad \forall x \in [0, 1]$$

cioè ϕ è anche una contrazione di $[a, b] = [0, 1]$ in se stesso.
Allora $\exists$ un unico ξ punto fisso di ϕ in $[0, 1]$: prendiamo

$$x_0 = \frac{1}{2} \Longrightarrow e_0 = |x_0 - \xi| \leq \frac{1}{2}$$

e la successione $x_{n+1} = \phi(x_n)$, $n \geq 0$ converge a ξ con la stima a priori dell'errore $e_n \leq \frac{\alpha^n}{2}$.

Nel grafico sotto si ha l'errore effettivo, stima a priori e stima a posteriori dello step pesato/non pesato con:
α = 0.2 → fl(ξ) = 0.8445798674955478
α = 0.9 → fl(ξ) = 0.5887032951482605
avendo la stima a priori che è una brutta sovrastima. Con 0.2 si ha convergenza più rapida, essendo α più piccolo, per quanto la stima in blu non sia molto lontana dalle altre. Il modulo di f'(ξ) comanda la convergenza, come si vede sotto:



Si vede chiaramente che la convergenza è lineare, che la stima a priori è una sovrastima, molto distante dall'errore per $\alpha = 0.9$: infatti

$$\frac{e_{n+1}}{e_n} = |\phi'(z_n)| \to |\phi'(\xi)|, \quad n \to \infty \quad \text{cioè} \quad |\phi'(\xi)| = \alpha e^{-\alpha\xi} = L$$

è la costante asintotica, il parametro che effettivamente regola la velocità di convergenza (α è solo una stima) perchè per n abbastanza grande

$$e_{n+1} \approx L e_n$$

Qui per $\alpha = 0.9$ si ha $L \approx 0.53$ che è ben minore di α, mentre per $\alpha = 0.2$ si ha $L \approx 0.17$ che è poco minore di α.
In effetti la 2° stima a posteriori,

$$e_n \approx \frac{|x_{n+1} - x_n|}{(1 - \phi'(x_n))}$$

è una stima aderente dell'errore ma è shiftata in avanti di 1, fenomeno che abbiamo già visto nella stima con lo step nel metodo di Newton, perchè per stimare e_n bisogna essere al passo $n+1$ (step $= |x_{n+1} - x_n|$)

Anche per le iterazioni di punto fisso vale un risultato di convergenza locale (con dim. facoltativa che non inserisco), in questo caso con x_0 abbastanza vicino a ξ:

2.4.4 TEOREMA (convergenza LOCALE delle iterazioni di punto fisso)

Sia ξ punto fisso di $\phi \in C^1(I_\delta(\xi))$ dove $I_\delta(\xi) = [\xi - \delta, \xi + \delta]$, $\delta > 0$ e sia $|\phi'(\xi)| < 1$ allora

$$\Rightarrow \exists \delta' \leq \delta : x_{n+1} = \phi(x_n),\ n \geq 0,\ \text{converge a } \xi, \quad \forall\, x_0 \in I_{\delta'}(\xi)$$

Dando poi un teorema fondamentale:

2.4.5 TEOREMA (ordine di convergenza delle iterazioni di punto fisso)

Sia ξ punto fisso di $\phi \in C^p(I)$, $p \geq 1$ con I intervallo di $\mathbb{R}$ e supponiamo di essere in ipotesi che garantiscano la convergenza a ξ di $x_{n+1} = \phi(x_n)$, $n \geq 0$, con $x_0 \in I$ (ad esempio le ipotesi del teorema delle contrazioni)
Allora:

1. $\{x_n\}$ ha ordine esattamente $p = 1 \iff 0 < |\phi'(\xi)| < 1$
2. $\{x_n\}$ ha ordine esattamente $p > 1 \iff \phi^{(j)}(\xi) = 0,\ 1 \leq j \leq p-1$ *e* $\phi^{(p)}(\xi) \neq 0$

Dimostrazione

1. si dimostra subito visto che

$$e_{n+1} = |\phi'(z_n)|e_n, \ z_n \in int(\xi, x_n)$$

per il teorema del valor medio, quindi

$$\lim_{n\to\infty} \frac{e_{n+1}}{e_n} = |\phi'(\lim z_n)| = |\phi'(\xi)|$$

2. per 2) utilizziamo la formula di Taylor di grado $p-1$ centrata in ξ con resto p-esimo in forma di Lagrange

$$x_{n+1} = \phi(x_n) = \underset{\xi}{\phi(\xi)} + \phi'(\xi)(x_n - \xi) + \frac{\phi''(\xi)}{2}(x_n-\xi)^2 + \ldots + \frac{\phi^{(p-1)}(\xi)}{(p-1)!}(x_n-\xi)^{p-1} + \frac{\phi^{(p)}(u_n)}{p!}(x_n-\xi)^p$$

con $u_n \in int(\xi, x_n)$

- **Dimostriamo prima " $\Leftarrow$ " (condizione sufficiente)**
  se $\phi^{(j)}(\xi) = 0, \ 1 \le j \le p-1$ e $\phi^{(p)}(\xi) \neq 0$, da Taylor

$$x_{n+1} - \xi = \frac{\phi^{(p)}(u_n)}{p!}(x_n - \xi)^p$$

e passando ai moduli

$$\frac{e_{n+1}}{e_n^p} = \frac{|\phi^{(p)}(u_n)|}{p!} \underset{n\to\infty}{\longrightarrow} \frac{|\phi^{(p)}(\xi)|}{p!} \neq 0$$

perché $u_n \to \xi$, $n \to \infty$ e $\phi^{(p)}$ è continua quindi

$$\lim \phi^{(p)}(u_n) = \phi^{(p)}(\lim u_n) = \phi^{(p)}(\xi)$$

ovvero $\{x_n\}$ ha ordine esattamente p.

Per **dimostrare "$\Rightarrow$" (condizione necessaria)**
supponiamo per assurdo che $\{x_n\}$ abbia ordine esattamente p ma che $\exists\ j < p$ tale che $\phi^{(j)}(\xi) \neq 0$, prendiamo $k = \min\{j < p : \phi^{(j)}(\xi) \neq 0\}$ e scriviamo

$$\frac{e_{n+1}}{e_n^p} = \frac{e_{n+1}}{e_n^k} \cdot e_n^{k-p}$$

Ora per ipotesi

$$\frac{e_{n+1}}{e_n^p} \to L \neq 0$$

d'altra parte con lo stesso ragionamento usato per "$\Leftarrow$" tramite la formula di Taylor si avrebbe

$$\frac{e_{n+1}}{e_n^k} \to \frac{|\phi^{(k)}(\xi)|}{k!} = L' \neq 0$$

ma allora

$$\frac{e_{n+1}}{e_n^p} = \frac{e_{n+1}}{e_n^k} \cdot e_n^{k-p}$$

$$\left(\frac{e_{n+1}}{e_n^k} \to L' \text{ ed } e_n^{k-p} \to \infty \text{ perchè } k-p < 0 \text{ ed } e_n \to 0\right)$$

cioè

$$\frac{e_{n+1}}{e_n^p} \to \infty, \quad n \to \infty$$

contraddicendo l'ipotesi che abbia limite finito.

ribadendo che la condizione è necessaria e sufficiente, caratterizzando completamente le iterazioni di punto fisso con ordine p >= 1, avendo in particolare convergenza quadratica con ϕ''(ξ) ≠ 0, cubica con ϕ'''(ξ) ≠ 0, ecc. Vediamo infine che il metodo di Newton è reinterpretabile come iterazione di punto fisso e ammette convergenza locale, con convergenza quadratica per zeri semplici.

Scritto da Gabriel

Infatti l'iterazione di Newton

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n \geq 0$$

è di tipo punto fisso con

$$\phi(x) = x - \frac{f(x)}{f'(x)}$$

e se $f \in C^k(I)$ con $f'(x) \neq 0 \forall x \in I$ intervallo, allora $\phi \in C^{k-1}(I)$.
È evidente che ξ è zero (semplice) di $f \iff \xi$ è punto fisso di ϕ.
Inoltre, il teorema di convergenza locale per Newton (zeri semplici) discende immediatamente dal teorema di convergenza locale per le iterazioni di punto fisso se $f \in C^2(I_\delta(\xi))$, infatti

$$\phi'(x) = \frac{d}{dx}\left(x - \frac{f}{f'}\right) = 1 - \underbrace{\frac{(f')^2 - ff''}{(f')^2}}_{\text{derivata del rapporto}} = \frac{ff''}{(f')^2}$$

quindi $f(\xi) = 0 \Rightarrow \phi'(\xi) = 0$ e $|\phi'(\xi)| = 0 < 1$, allora $\exists \delta' \leq \delta$ tale che l'iterazione di Newton converge come iterazione di punto fisso $\forall x_0 \in I_{\delta'}(\xi)$.

Interpretando Newton come iterazione di punto fisso, la convergenza per zeri semplici è almeno quadratica, con $\phi'(\xi) = 0$ e quadratica con $\phi''(\xi) \neq 0$, infatti:

$$\phi'' = \frac{d}{dx}\left(\frac{ff''}{(f')^2}\right) = \underbrace{\frac{(f'f'' + ff''')(f')^2 - 2f'f''(ff'')}{(f')^4}}_{\text{ancora derivata del rapporto}} \quad \overset{\downarrow \textit{ derivata del prodotto}}{}$$

da cui

$$\phi''(\xi) = \frac{f''(\xi)}{f'(\xi)} \neq 0$$

perché

$$f''(\xi) \neq 0 \ (e\ f'(\xi) \neq 0)$$

Non è difficile vedere (come approfondimento facoltativo) che interpretando Newton come iterazione di punto fisso, nel caso di zero multiplo l'ordine di convergenza diventa $p = 1$ con costante asintotica

$$|\phi'(\xi)| = 1 - \frac{1}{m}$$

dove m è la molteplicità di ξ (il numero di derivate successive che si annullano in ξ) e che se m è nota, l'iterazione

$$x_{n+1} = x_n - m\frac{f(x_n)}{f'(x_n)} = \phi_m(x_n)$$

torna di ordine almeno $p = 2$ perché

$$\phi_m'(\xi) = \lim_{x \to \xi} \phi_m'(x) = 0$$

Capitolo 3: Interpolazione e approssimazione di dati e funzioni

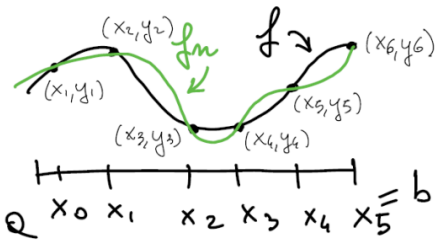
Lezione 12 - Ricostruzione di funzioni da dati discreti, interpolazione polinomiale, esistenza e unicità, il problema della convergenza, esempio di Runge

Ci occupiamo della ricostruzione approssimata di funzioni da dati discreti, partendo da un campionamento finito, concentrandoci sull'interpolazione e sull'approssimazione ai minimi quadrati.
Partendo quindi dall'interpolazione polinomiale:

dati $n+1$ punti $\{(x_i, y_i)\}$ con $y_i = f(x_i)$, $0 \le i \le n$, sul grafico di una funzione $f : [a, b] \to \mathbb{R}$, cerchiamo una funzione $f_n \in \mathcal{F}_n$ (dove $\mathcal{F}_n$ è una opportuna famiglia di funzioni "semplici"), tale che

$$f_n(x_i) = y_i, \ 0 \le i \le n$$

cioè tale che il grafico di f_n passi per il grafico di f in corrispondenza delle ascisse di interpolazione $\{x_i\}$ (che chiameremo nodi, mentre gli $\{y_i\}$ sono i valori della funzione campionata)



Lo scopo dell'interpolazione è la ricostruzione della funzione fuori dal campionamento, avendo quindi f_n scelta in modo tale che dist(f, f_n) → 0, n → ∞
rappresentando con "dist" la distanza tra due funzioni, errore commesso in fase di approssimazione.
Date due funzioni continue *f, g* ∈ C[a, b]:

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

quindi il massimo modulo della differenza dei valori assunti dalle due funzioni (grazie a Weierstrass esistono min e max assoluti) e la distanza è ben definita, con una convergenza definita come *convergenza uniforme*, avendo un errore ben approssimato dappertutto. Potremo quindi parlare di convergenza una volta decisa la famiglia di funzioni e avendo garanzia dell'esistenza di f_n.
Ci concentriamo sui *polinomi*, funzioni facili da gestire e memorizzare, scritte nella forma:

$$p(x) = a_0 + a_1 x + \ldots + a_n x^n$$

Il primo problema che dobbiamo risolvere è di tipo algebrico: dati $n+1$ punti del grafico di f, cioè $n+1$ punti del piano $\{(x_i, y_i)\}_{0 \le i \le n}$ con $y_i = f(x_i)$, esiste un polinomio $f_n \in \mathbb{P}_n$ che interpola f? E se esiste, è unico?
Innanzitutto osserviamo che le incognite in questo problema non sono le "x", visto che i nodi di campionamento $\{x_i\}$ fanno parte dei dati del problema, ma sono gli $n+1$ coefficienti $\{a_j\}$. Quindi cercare un polinomio interpolatore in $\mathbb{P}_n$ è una scelta del tutto sensata, perchè il numero di incognite è uguale al numero di vincoli (i vincoli di interpolazione $f_n(x_i) = y_i$, $0 \le i \le n$).
Scrivendo esplicitamente tali vincoli otterremo un sistema di equazioni con tante equazioni quante sono le incognite. Per capirlo, trattiamo il caso in cui $n = 5$ (come nel disegno fatto sopra), cioè cerchiamo un polinomio di grado 5

$$f_5(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5$$

con i seguenti vincoli di interpolazione dati da un sistema lineare di 6 equazioni nelle varie incognite:

$$\begin{cases} f_5(x_0) = a_0 + a_1 x_0 + \ldots + a_5 x_0^5 = y_0 \\ f_5(x_1) = a_0 + a_1 x_1 + \ldots + a_5 x_1^5 = y_1 \\ f_5(x_2) = a_0 + a_1 x_2 + \ldots + a_5 x_2^5 = y_2 \\ f_5(x_3) = a_0 + a_1 x_3 + \ldots + a_5 x_3^5 = y_3 \\ f_5(x_4) = a_0 + a_1 x_4 + \ldots + a_5 x_4^5 = y_4 \\ f_5(x_5) = a_0 + a_1 x_5 + \ldots + a_5 x_5^5 = y_5 \end{cases}$$

Scritto da Gabriel

Di fatto il sistema è lineare perché un polinomio di grado <= n si scrive come combinazione lineare degli n+1 monomi, prendendo ad esempio il seguente spazio vettoriale con dimensione n+1 dei monomi linearmente indipendenti:

$$\mathbb{P}_n = \left\{ p(x) = \sum_{j=0}^{n} a_j x^j \right\}$$

generalmente poi interpretato sotto forma di sistema lineare di interpolazione partendo da:

$$f_n(x_i) = \sum_{j=0}^{n} a_j x_i^j = y_i \quad 0 \le i \le n$$

dove rappresentiamo poi in forma compatta come $V\underline{a} = \underline{y}$ dove:

$$\underline{a} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix}, \quad \underline{y} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix} \in \mathbb{R}^{n+1}$$

dove questi sopra sono rispettivamente vettori dei coefficienti incogniti e vettore dei valori campionati, mentre $V=x_{ij}$ è definita a livello matriciale come "matrice di Vandermonde", dipendente solo dai nodi di interpolazione (se i nodi sono distinti il sistema ha soluzione unica, quindi esiste il polinomio di grado <= n che interpola gli n+1 dati, definita come prodotto di tutte le possibili differenze):

$$V = \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}$$

Supponendo che esistano due polinomi $p, q \in P_n$ | $p(x_i) = y_i = q(x_i)$ allora il polinomio "$p - q$", dimostrando che il polinomio interpolatore è unico:

$$p, q \in \mathbb{P}_n \Rightarrow \alpha p + \beta q \in \mathbb{P}_n \quad \forall \alpha, \beta \in \mathbb{R})$$

e si ha

$$(p-q)(x_i) = p(x_i) - q(x_i) = 0, \quad 0 \le i \le n$$

cioè $p - q$ avrebbe $n + 1$ zeri distinti.
Ma per il teorema fondamentale dell'algebra già ricordato sopra, $p - q$ può avere al massimo n zeri distinti, a meno che non sia il polinomio nullo, cioè deve essere

$$(p-q)(x) = 0 \quad \forall x \Rightarrow p(x) = q(x) \ \forall x$$

Per dire poi che esiste il polinomio interpolatore, consideriamo che:
Dati gli $n + 1$ nodi distinti $\{x_i\}_{0 \le i \le n}$ consideriamo per ogni nodo fissato (cioè per ogni i fissato) il "polinomio elementare di Lagrange" così definito

$$l_i(x) = \frac{N_i(x)}{N_i(x_i)}$$

dove

$$N_i(x) = (x - x_0)(x - x_1) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_n) = \prod_{j=0,\ j \ne i}^{n} (x - x_j)$$

avendo che $N_i(x)$ è polinomio di grado effettivo "n" mentre $N_i(x_i)$ è numero ≠ 0 e:

Quindi $l_i(x) \in \mathbb{P}_n$ e ha grado effettivo n,

$$l_i(x) = \frac{1}{N_i(x_i)} x^n + \dots$$

Inoltre

$$l_i(x_k) = \delta_{ik} = \begin{cases} 0 & i \neq k \\ 1 & i = k \end{cases}$$

(il cosiddetto "delta di Kronecker"). Infatti

$$l_i(x_i) = \frac{N_i(x_i)}{N_i(x_i)} = 1$$

e per $k \neq i$

$$N_i(x_k) = (x_k - x_0) \dots \overbrace{(x_k - x_k)}^{=0} \dots (x_k - x_n) = 0$$

quindi $N_i(x_k)$ che contiene un fattore nullo che annulla il prodotto, ad esempio:

Ad esempio

$$N_0(x) = (x - x_1) \dots (x - x_n)$$

e

$$N_0(x_1) = 0 = N_0(x_2) = \dots = N_0(x_n)$$

allo stesso modo

$$\begin{aligned} N_1(x) &= (x - x_0)(x - x_2) \dots (x - x_n) \\ N_1(x_0) &= 0 = N_1(x_2) = \dots = N_1(x_n) \\ &\dots \\ N_n(x) &= (x - x_0)(x - x_1) \dots (x - x_{n-1}) \\ N_n(x_0) &= 0 = N_n(x_1) = \dots = N_n(x_{n-1}) \end{aligned}$$

possiamo allora definire

$$f_n(x) = \prod_n(x) = \sum_{i=0}^{n} y_i l_i(x)$$

che è il polinomio interpolatore di Lagrange, definito come:

$$\begin{aligned} \prod_n(x_k) &= \sum_{i=0}^{n} y_i l_i(x_k) \\ &= \sum_{i=0}^{n} y_i \delta_{ik} \\ &= y_k \ \delta_{kk} \quad \longleftarrow \quad \text{perchè } \delta_{ik} = 0, i \neq k \\ &= y_k, \quad 0 \leq k \leq n \end{aligned}$$

che è interpolatore su n+1 nodi distinti.

Prima osservazione importante:

Chi è l'interpolatore?

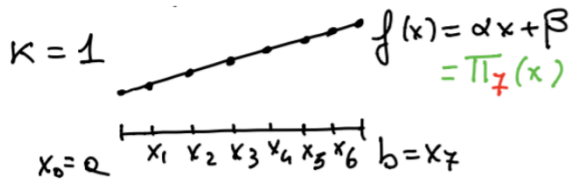
L'interpolatore è il polinomio stesso, cioè se $f(x) = p(x) \in \mathbb{P}_n$ allora $\prod_n(x) = p(x)$, perché il polinomio interpolatore è unico in $\mathbb{P}_n$ e $p(x)$ ovviamente interpola se stesso.

Questo ci fa capire che un polinomio interpolatore su $n+1$ nodi distinti può avere grado $< n$: ad esempio, se $f(x) = a_1 x + a_0$ (cioè f è un polinomio di grado 1), allora $\prod_n(x) = f(x)\ \forall n \geq 1$ e allo stesso modo $f(x) = a_2 x^2 + a_1 x + a_0$

$$\implies \prod_n(x) = f(x) \quad \forall n \geq 2$$

In generale se $f(x) \in \mathbb{P}_k$ allora $\prod_n(x) = f(x)\ \forall n \geq k$

Scritto da Gabriel



$$\prod_7(x) = f(x) = \alpha x + \beta$$

(per quanti modi di campionamento mettiamo, l'interpolatore resterà sempre $\alpha x+\beta$ cioè di grado 1).

Seconda osservazione importante:
Una volta dimostrata l'unicità avremo anche l'esistenza, infatti, si parte dall'interpolazione in P_n su n+1 nodi diversi, equivalente al sistema lineare di dimensione n+1 $\quad V\underline{a} = \underline{y}$

Ora, se c'è unicità $V\underline{a} = \underline{0} \Rightarrow \underline{a} = \underline{0}$ cioè l'applicazione lineare associata a V è iniettiva cioè le colonne di V sono linearmente indipendenti cioè $rango(V) = n + 1$ e quindi V è invertibile (ricordiamo che $V\underline{a} = \sum_{k=1}^{n+1} a_k l_k(V)$ dove $l_k(V)$ è la colonna k-esima di V).
D'altra parte, se c'è esistenza $\forall \underline{y} \in \mathbb{R}^{n+1}$ significa che le $n+1$ colonne di V generano tutti i vettori di $\mathbb{R}^{n+1}$ cioè sono una base (l'applicazione lineare è suriettiva) e quindi $rango(V) = n + 1 \Rightarrow V$ invertibile.
Però dal punto di vista applicativo non ci accontentiamo di questo, in particolare è stato importante scrivere il polinomio interpolatore $\prod_n$ in una forma esplicita, la forma di Lagrange (ma ce ne sono altre), utile, come vedremo, anche per studiare altri aspetti come ad es. la stabilità dell'interpolazione (nel senso della "risposta" dell'interpolazione ad errori sui valori $\{y_i\}$).
Possiamo anche osservare che in base all'Osservazione 1, la forma di Lagrange ci dice che i polinomi elementari di Lagrange

$$\{l_i(x)\}_{0\le i\le n}$$

sono una base di $\mathbb{P}_n$ (diversa dalla base monomiale $\{x^i\}_{0\le i\le n}$).
Infatti, se $p \in \mathbb{P}_n$ per l'unicità

$$\prod_n(x) = p(x) = \sum_{i=0}^{n} p(x_i) l_i(x)$$

cioè ogni polinomio di grado $\le n$ si può scrivere come combinazione lineare degli $n + 1$ polinomi elementari (che dipendono solo dalla scelta dei nodi di interpolazione $\{x_i\}$)

Si pone poi il *problema della convergenza*, in cui l'interpolazione si pone l'idea di usare l'informazione su una funzione ottenuta tramite campionamento discreto ricostruendo la funzione. Partiamo sempre dalla formula di Taylor, che però ha carattere locale, ma noi ovviamente siamo interessati ad un'approssimazione globale, infittendo il campionamento.
Citiamo solo il seguente teorema utile al nostro scopo:

3.1.7 TEOREMA (di densità di Weierstrass)

$\forall f \in C[a, b]$ fissata e $\forall \varepsilon > 0$

$$\exists\, p_\varepsilon \in \mathbb{P} : dist(f, p_\varepsilon) \le \varepsilon$$

Dove:

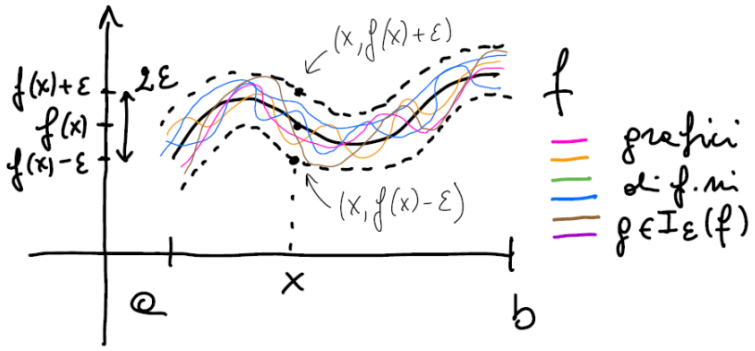
- $\mathbb{P} = \bigcup_{n\ge 0} \mathbb{P}_n$ è l'insieme (che è anche spazio vettoriale) di tutti i polinomi di tutti i gradi
- $dist(f, g) = \max_{x\in[a,b]} |f(x) - g(x)|$ è la distanza tra funzioni (continue) che avevamo definito all'inizio

osservando che avendo una distanza fra funzioni siamo in grado di definire cosa sia un intorno di raggio ε>0 di una f ∈ C[a, b]:

Scritto da Gabriel

$$I_\varepsilon(f) = \{g \in C[a,b] : dist(f,g) \le \varepsilon\}$$

Graficamente, consideriamo un "tubicino" di ampiezza verticale 2ε costruito intorno al grafico di f



Chi è $I_\varepsilon(f)$? Non è il "tubicino" dai contorni tratteggiati "paralleli" al grafico di f (questo permette solo la rappresentazione grafica), invece $I_\varepsilon(f)$ è l'insieme delle infinite funzioni continue il cui grafico "vive" nel tubicino, cioè i cui valori $\forall x \in [a,b]$ stanno tra $f(x)-\varepsilon$ e $f(x)+\varepsilon$, cioè

$$f(x) - \varepsilon \le g(x) \le f(x) + \varepsilon$$

Diciamo inoltre che il teorema di densità di Weierstrass afferma che ogni funzione $f \in C[a,b]$ si può *approssimare arbitrariamente* bene su tutto l'intervallo (perché si parla del *massimo errore*) con un *opportuno polinomio*.

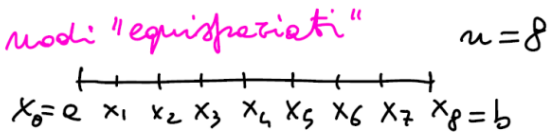
Una conseguenza (che non dimostriamo) del teorema di Weierstrass è che $\forall n \ge 0$ $\exists$ un (unico) polinomio $p_n^* \in \mathbb{P}_n$ tale che

$$\min_{p \in \mathbb{P}_n} dist(f,p) = dist(f,p_n^*) \underset{n\to\infty}{\longrightarrow} 0$$

che si chiama "polinomio di migliore approssimazione uniforme" di f in $\mathbb{P}_n$ (purtroppo questo polinomio ottimale è molto difficile da calcolare); ma visto che è possibile approssimare bene $f \in C[a,b]$ (e d'ora in poi ci muoveremo nell'ambito di funzioni almeno continue) con polinomi, c'è la speranza che infittendo il campionamento, la successione di interpolatori $\{\prod_n\}$ converga nella distanza del max errore a $f \in C[a,b]$? Cioè è vero che $\lim_{n\to\infty} \prod_n = f$ nel senso che $dist(f, \prod_n) \to 0$, $n \to \infty$? (la speranza viene dal fatto che infittendo il campionamento stiamo in effetti aumentando l'informazione sulla funzione f)
Purtroppo la risposta è NO, cioè in generale non è vero che la successione di interpolatori converge alla funzione campionata (come vedremo, la convergenza dipende dalla distribuzione dei nodi $\{x_i\}$).
A questo proposito, consideriamo uno dei metodi più usuali di campionamento, cioè un campionamento a passo costante in $[a,b]$, cioè

$$x_i = a + ih, \quad 0 \le i \le n, \quad h = \frac{b-a}{n}$$

che consiste nel partizionare $[a,b]$ in n sottointervalli di ampiezza uguale e prendere come nodi di campionamento i loro estremi



$$\begin{aligned} x_0 &= a, \\ x_1 &= a + h, \\ x_2 &= a + 2h, \\ &\dots \\ x_n &= a + nh = b \end{aligned}$$

Questo è un metodo di campionamento del tutto naturale nelle scienze sperimentali e nelle applicazioni tecnologiche, si pensi ad esempio al caso in cui la variabile indipendente è il tempo e si campiona con un passo temporale costante (1 min, 1 sec, 1 millisec, ...).
Bene (anzi, male), purtroppo ci sono funzioni anche molto regolari per cui l'interpolazione polinomiale a passo costante non converge.

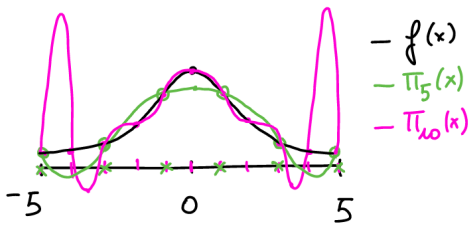
Un classico esempio verificabile sperimentalmente è la *funzione di Runge*, definita come:

Scritto da Gabriel

$$f(x) = \frac{1}{1+x^2},\ x \in [a,b] = [-5,5]$$

$$x_i = a + ih = -5 + i \cdot \frac{10}{n},\ 0 \le i \le n$$

Si osservi che f è pari ed estremamente regolare, $f \in C^\infty(\mathbb{R})$ (per inciso, f è la derivata di $\arctan(x)$); ma cosa succede interpolando f a passo costante?



vedendo che verso gli estremi dell'intervallo, anche infittendo il campionamento, l'approssimazione peggiora. Il fenomeno persiste al crescere di "n" e verso gli estremi si generano oscillazioni sempre più ampie definite da:

$$dist\left(\prod_n^{eq}, \frac{1}{1+x^2}\right) = \max_{x\in[-5,5]}\left|\prod_n^{eq}(x) - \frac{1}{1+x^2}\right| \underset{n\to\infty}{\longrightarrow} \infty$$

Tale fenomeno di divergenza mostra che la scelta di nodi equispaziati non è appropriata per ricostruire la funzione per interpolazione, essendo troppo vicini all'intervallo in rapporto alla lunghezza dell'intervallo stesso.
Ci sono quindi due strade per risolvere il problema di convergenza dell'interpolazione polinomiale:

- usando distribuzioni speciali di nodi di campionamento
- cambiando tipo di interpolazione, passando dall'interpolazione con un polinomio di grado → ∞ all'interpolazione polinomiale a tratti, in cui si usano polinomi di grado fissato di sottointervalli di ampiezza → 0

Lezione 13 - Formula dell'errore di interpolazione, interpolazione di Chebyshev, costante di Lebesgue e stabilità dell'interpolazione

Finora non abbiamo fornito stime dell'errore di interpolazione e della distanza:

$$dist\,(f, \prod_n) = \max_{x\in[a,b]} |\, f(x) - \prod_n(x) \,|$$

dove $f \in C[a,b]$ e $\prod_n$ è il polinomio interpolatore di grado $\le n$ su $n+1$ nodi distinti in $[a,b]$.
A questo proposito, enunciamo e dimostriamo qui sotto un classico risultato di rappresentazione.

3.2.1 TEOREMA (rappresentazione di $f(x) - \prod_n(x)$ per $f \in C^{n+1}[a,b]$)

Siano $f \in C^{n+1}[a,b]$ e $\prod_n \in \mathbb{P}_n$ il polinomio interpolatore su $n+1$ nodi distinti $\{x_i\} \subset [a,b]$, allora

$$E_n(x) = f(x) - \prod_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x)$$

dove

$$\omega_{n+1}(x) = \prod_{i=0}^{n}(x - x_i)$$
$$= (x - x_0)(x - x_1)\dots(x - x_n)$$

e $\xi \in int(x, x_0, \dots, x_n)$ (l'interno del più piccolo intervallo che contiene tutti i nodi e x fissato in $[a,b]$).

Pensiamo di aver ordinato i nodi tale che siano equispaziati (a <= x_0 < x_1 < … < x_n <= b, non è detto che x_0 = a o x_0 = b).
In questo caso la formula è simile a Taylor, ma l'interpolazione è a carattere globale e abbiamo la produttoria con "x" che varia nell'intervallo [a,b] (dim. facoltativa che non inserisco).

Scritto da Gabriel

Ricordiamo il *teorema di Rolle*:

TEOREMA (di Rolle)

Se $\phi \in C[\alpha, \beta]$ è derivabile in (α, β) e $\phi(\alpha) = \phi(\beta)$

$$\Rightarrow \exists z \in (\alpha, \beta) : \phi'(z) = 0$$

notando che la dimostrazione del teorema vale quando la funzione è costante oppure max/min assoluti sono nell'intervallo aperto [a,b] con la derivata che si annulla e, per il teorema applicato a *g* esistono n+1 punti ognuno interno a ciascun intervallo tali che g'(ξ_{ji})=0. Quindi avremo una serie di *n* intervalli con estremi i punti ξ_{ji} in cui g' si annulla.

Applicando ripetutamente il teorema di Rolle a $g'', g''', \dots,$ arriviamo infine a dire che $\exists \xi = \xi_{n+1,1} \in int(x, x_0, \dots, x_n)$ tale che $g^{(n+1)}(\xi) = 0$ ma

$$\begin{aligned} g^{(n+1)}(z) &= E_n^{(n+1)}(z) - \omega_{n+1}^{(n+1)}(z)\frac{E_n(x)}{\omega_{n+1}(x)} \\ &= f^{(n+1)}(z) - (n+1)!\,\frac{E_n(x)}{\omega_{n+1}(x)} \end{aligned}$$

perchè $\prod_n^{(n+1)}(z) = 0 \ \forall z$ visto che $\prod_n \in \mathbb{P}_n$ e $\omega_{n+1}^{(n+1)}(z) = (n+1)!$ visto che $\omega_{n+1}(z) = z^{n+1} + \dots$
(da cui $\omega'_{n+1}(z) = (n+1)z^n + \dots$, $\omega''_{n+1}(z) = (n+1)nz^{n-1} + \dots$, $\dots$).
Quindi $\exists \xi \in int(x, x_0, \dots, x_n)$ tale che

$$f^{(n+1)}(\xi) - (n+1)!\frac{E_n(x)}{\omega_{n+1}(x)} = 0$$

che è la rappresentazione cercata. ■

Grazie alla rappresentazione esplicita di E_n siamo in grado di fare stime di:

$$\max_{x\in[a,b]} |E_n(x)| = dist\left(f, \prod_n\right)$$

Innanzitutto osserviamo che

$$f \in C^{n+1}[a,b] \Longrightarrow f^{(n+1)} \in C[a,b]$$

e quindi per il teorema di Weierstrass sull'esistenza di *max* e *min* assoluti di una funzione continua su un intervallo chiuso e limitato

$$\left|f^{(n+1)}(\xi)\right| \leq \max_{x\in[a,b]} \left|f^{(n+1)}(x)\right| = M_{n+1} \tag{3.1}$$

Inoltre comunque $|x - x_i| \leq b - a$ quindi

$$|\omega_{n+1}(x)| \leq (b-a)^{n+1} \tag{3.2}$$

(che è una stima rozza ma valida per qualsiasi distribuzione dei nodi).

Quindi in generale possiamo scrivere

$$dist\left(f, \prod_n\right) = \frac{f^{n+1}(\xi)}{(n+1)!}\omega_{n+1}(x) \overset{(1)}{\leq} M_{n+1}\frac{\omega_{n+1}(x)}{(n+1)!} \overset{(2)}{\leq} \overbrace{M_{n+1}\frac{(b-a)^{n+1}}{(n+1)!}}^{stima}$$

La stima potrebbe sembrare tendente a 0 per n → ∞, dato che:

$$\frac{(b-a)^{n+1}}{(n+1)!} \to 0, \quad n \to \infty$$

perchè è il termine generale della serie di e^{b-a} che converge.
Ma sappiamo dall'esempio di Runge che ci sono casi in cui

$$dist\left(f, \prod_n\right) \to \infty, \quad n \to \infty$$

Può quindi succedere che il fattore M_{n+1} sia non limitato e cresca più rapidamente del termine generale della serie, avendo stima divergente. Nel caso però di nodi equispaziati, al posto dell'esempio di Runge (in cui si ha l'errore max divergente e di fatto stima divergente), si ricava:

$$dist\left(f, \prod_n\right) \leq M_{n+1} \frac{h^{n+1}}{4(n+1)}$$

dove $h = \frac{(b-a)}{n}$, ma di nuovo quello che conta è la velocità di crescita di M_{n+1} (che nel caso dell'esempio di Runge cresce più rapidamente addirittura di $(n+1) \cdot h^{-(n+1)}$).

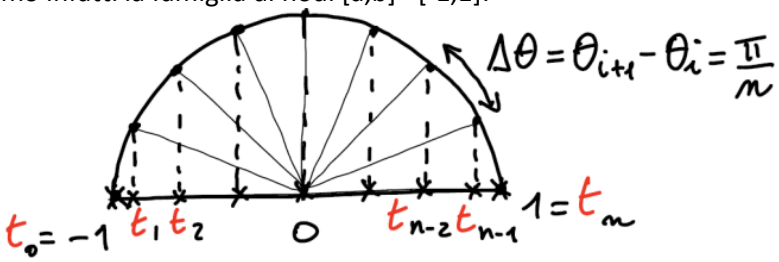
La stima fa quindi capire che ci sono funzioni per cui l'interpolazione è sempre convergente e se una funzione C[a,b] ha tutte le derivate *equilimitate* (limitate in modulo dalla stessa costante), allora:

$$dist\left(f, \prod_n\right) \leq M \frac{(b-a)^{n+1}}{(n+1)!} \underset{n\to\infty}{\longrightarrow} 0$$

Esempi sono:

- $f(x) = e^x$ per cui $\left|f^{(n+1)}(x)\right| = e^x \leq e^b \quad \forall x \in [a,b]$
- $f(x) = \sin(x)$ e $f(x) = \cos(x)$ per cui $\left|f^{(n+1)}(x)\right|$ è $|\sin(x)|$ oppure $|\cos(x)|$ e quindi $M_{n+1} \leq 1$

idea che comunque, pur in maniera "fortunata" non risolve il problema della convergenza.
Volendo usare un unico polinomio interpolatore, anche con nodi sparsi, si hanno situazioni di non convergenza.
Vediamo infatti la famiglia di nodi [a,b] =[-1,1]:



cosiddetti punti equispaziati sulla semicirconferenza di centro (0,0) e raggio 1 corrispondenti agli angoli:

$$\theta_i = i \cdot \frac{\pi}{n}, \quad 0 \leq i \leq n,$$

con $\theta_0=0$, $\theta_1= \pi/n$, $\theta_2= 2\pi/n$ … $\theta_n=n\pi/n$ proiettati sull'intervallo [-1, 1].

e trattansi di coseni, in quanto il segno viene cambiato per avere $-1 = t_0 < t_1 < \ldots t_n = 1$, addensandosi al crescere di n verso 0 per interpolazione. Sono così definiti:

$$t_i^{Cheb} = -\cos(\theta_i) = -\cos\left(\frac{i\pi}{n}\right), \quad 0 \leq i \leq n$$

Si può dimostrare (ma la dimostrazione è molto difficile e fa parte di un'intera teoria dell'approssimazione con polinomi) che l'interpolatore su questi nodi, che chiameremo $\prod_n^{Cheb}$, converge uniformemente se $f \in C^k[-1,1]$ per $k > 0$, cioè

$$dist(f, \prod_n^{Cheb}) \to 0 \quad n \to \infty$$

Per un intervallo generale $[a,b]$ il risultato resta valido se si prendono i nodi ottenuti dalla trasformazione affine

$$\alpha(t) = \frac{b-a}{2} \cdot t + \frac{b+a}{2}, \quad t \in [-1,1]$$

che manda $[-1,1] \to [a,b]$, cioè i nodi $x_i^{Cheb} = \alpha(t_i^{Cheb})$ (geometricamente, corrispondono alla stessa costruzione di prima fatta con la semicirconferenza centrata nel punto medio dell'intervallo, $\frac{b+a}{2}$, e di raggio la semilunghezza dell'intervallo, $\frac{b-a}{2}$).
In effetti si riesce anche a dare l'ordine di infinitesimo in n per k fissato

Diamo quindi il seguente teorema:

3.2.4 TEOREMA (convergenza uniforme dell'interpolazione di Chebyshev)

Sia $f \in C^k[a,b]$, $k > 0$, allora

$$\exists\, c_k > 0 : dist\left(f, \prod_n^{Cheb}\right) \leq c_k \frac{\log(n)}{n^k}$$

Si osservi che

$$\frac{\log(n)}{n^k} \to 0, \quad n \to \infty, \quad \forall\, k > 0$$

Con i nodi di Chebyshev si risolve il problema della convergenza uniforme per l'interpolazione polinomiale. Non risulta essere l'unica famiglia di nodi di questo tipo, ad esempio anche:

Un'altra famiglia, ad esempio, è

$$x_i^{Cheb2} = \alpha(t_i^{Cheb2})$$

con

$$t_i^{Cheb2} = -\cos\left(\frac{\pi}{2} \cdot \frac{2i+1}{n+1}\right), \quad 0 \leq i \leq n$$

Resta quindi da discutere il problema della ***stabilità***, dato che i valori campionati non sono mai noti in modo esatto, in quanto sempre affetti da errore del tipo:

$$\max_{0 \leq i \leq n} |\, \tilde{y}_i - y_i \,| \leq \varepsilon$$

Quindi il polinomio interpolatore sarà $\tilde{\prod}_n(x)$ invece di $\prod_n(x)$.

Come risponde quindi l'interpolatore agli errori sui dati? Dobbiamo fare una stima della distanza, capendo come dipende da "ε" e da "n". Usiamo quindi la forma di Lagrange, con interpolatore esatto:

$$\prod_n(x) = \sum_{i=0}^{n} y_i\, l_i(x), \quad y_i = f(x_i)$$

mentre l'interpolatore "perturbato" (quello costruito coi valori affetti da errore, gli unici che abbiamo veramente a disposizione) è

$$\tilde{\prod}_n(x) = \sum_{i=0}^{n} \tilde{y}_i\, l_i(x)$$

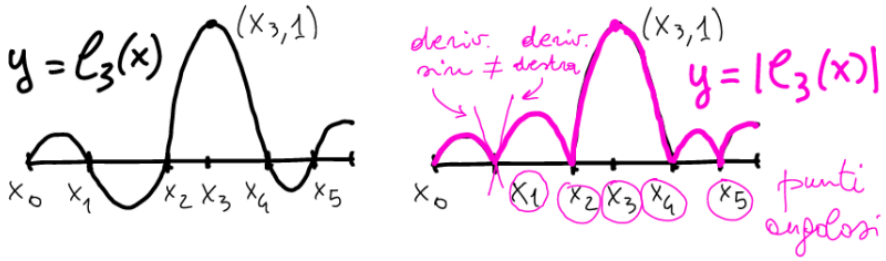
Allora

$$\begin{aligned}
\left| \prod_n(x) - \tilde{\prod}_n(x) \right| &= \left| \sum_{i=0}^{n} y_i\, l_i(x) - \sum_{i=0}^{n} \tilde{y}_i\, l_i(x) \right| \\
&= \left| \sum_{i=0}^{n} (y_i - \tilde{y}_i)\, l_i(x) \right| \\
&\leq \sum_{i=0}^{n} \overbrace{|\, y_i - \tilde{y}_i \,|}^{\leq \varepsilon} |\, l_i(x) \,| \\
&\leq \varepsilon \sum_{i=0}^{n} |\, l_i(x) \,|, \quad \forall\, x \in [a,b]
\end{aligned}$$

La funzione

$$\lambda_n(x) = \sum_{i=0}^{n} |\, l_i(x) \,|$$

si chiama funzione di Lebesgue: non è un polinomio, ma è la somma di moduli di polinomi (i polinomi elementari di Lagrange) ed è solo continua in $[a,b]$, perchè gli zeri di $l_i(x)$ in (a,b), cioè i nodi $\{x_j \in (a,b) : j \neq i\}$, sono punti "angolosi" (cioè punti di non derivabilità di $|\, l_i(x) \,|$, per cui $|l_i| \in C[a,b]$ ma $|l_i| \notin C^1[a,b]$



Prendendo poi il massimo, capiamo che l'errore dipende da ε, infatti:
(ricordando che Λ rappresenta la lambda maiuscola)

Prendendo il max in $[a,b]$ da ambo i lati della disuguaglianza

$$\left|\prod_n(x) - \tilde{\prod}_n(x)\right| \leq \varepsilon \sum_{i=0}^{n} |l_i(x)| = \varepsilon \lambda_n(x)$$

Si ottiene

$$dist\left(\prod_n, \tilde{\prod}_n\right) \leq \varepsilon \Lambda_n$$

dove

$$\Lambda_n = \max_{x\in[a,b]} \lambda_n(x)$$

viene detta COSTANTE DI LEBESGUE dei nodi di interpolazione.
Si noti infatti che tale quantità (costante in x) dipende solo dai nodi $\{x_i\}$ e agisce come un coefficiente di amplificazione del massimo errore ε sui dati.
Si può dimostrare (ma è difficile) che esistono delle costanti $\alpha_1, \alpha_2, \alpha_3 > 0$ tali che

1. $\Lambda_n \geq \alpha_1 log(n)$ per qualsiasi distribuzione di nodi
2. $\Lambda_n^{eq} \sim \alpha_2 \frac{2^n}{nlog(n)}$, $n \to \infty$, $\alpha_2 = \frac{2}{e} \approx 0.74$
3. $\Lambda_n^{Cheb} \leq \alpha_3 log(n)$, $\alpha_3 \approx \frac{2}{\pi} \approx 0.64$

quindi la costante di Lebesgue cresce almeno logaritmicamente, ma nel caso di nodi equispaziati ha crescita di fatto esponenziale, con distribuzioni dei nodi tipo Chebyshev con crescita logaritmica e quasi ottimale.
La stima mostra che l'interpolazione su nodi equispaziati risulta *instabile*, mentre l'int. di Chebyshev risulta *stabile*.
Ad esempio per $n = 30$ si ha:

- $\Lambda_{30}^{eq} \approx 8 \cdot 10^6$
- $\Lambda_{30}^{Cheb} \lesssim 2.2$

e per $n = 50$:

- $\Lambda_{50}^{eq} \approx 4 \cdot 10^{12}$
- $\Lambda_{50}^{Cheb} \lesssim 2.49$

Ciò significa che anche per le classi di funzioni su cui convergerebbe (ad es. funzioni equilimitate), l'interpolazione polimoniale su nodi equispaziati è inutilizzabile appena *n* cresce.

Invece l'interpolazione di Chebyshev è un'ottima scelta, visto che partendo dalla disuguaglianza

$$\left|f(x)-\tilde{\Pi}_n(x)\right| \leq \left|f(x)-\Pi_n(x)\right| + \left|\Pi_n(x)-\tilde{\Pi}(x)\right|$$

e prendendo il max in $[a,b]$ da ambo i lati, otteniamo

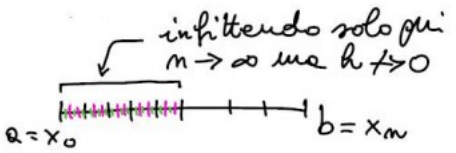
$$dist\left(f,\tilde{\Pi}_n\right) \leq \underbrace{dist\left(f,\Pi_n\right)}_{\text{convergenza}} + \underbrace{dist\left(\Pi_n,\tilde{\Pi}_n\right)}_{\text{stabilità}} \underbrace{\lesssim}_{\text{per i nodi di Cheb}} c_k\frac{\log(n)}{n^k} + 0.64\cdot\log(n)\cdot\varepsilon$$

se $f \in C^k[a,b], k>0$ e $\max|y_i-\tilde{y}_i| \leq \varepsilon$

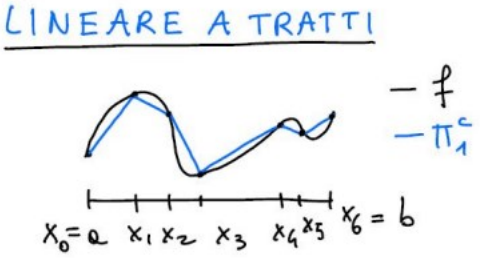
Nell'interpolazione di Chebyshev si ha uno svantaggio, in quanto è necessario interpolare su *quei* nodi specificamente per poter avere convergenza e stabilità.

Lezione 14 - Interpolazione polinomiale a tratti, convergenza e stabilità, interpolazione spline

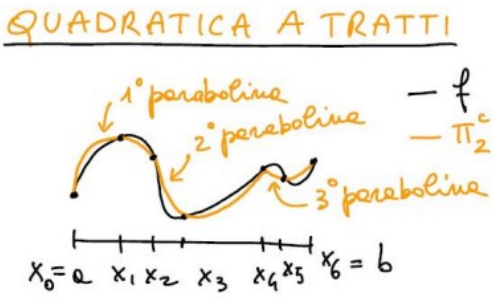
L'idea è di costruire funzioni polinomiali interpolanti *a tratti*, "incollando" per continuità interpolatori di grado *fissato* su una partizione dell'intervallo [a,b] in intervalli consecutivi.
Assumiamo che a = $x_0 < x_1 < x_2 < \dots < x_n$ = b
Il parametro che infittisce il campionamento è *h* = max Δx_i con $\Delta x_i = x_{i+1}-x_i$, 0 <= i <= n – 1.
Aumentando il numero di nodi non si aumenta l'informazione campionata, ma vengono solo infittiti in una parte dell'intervallo:



Per capire la tecnica, iniziamo con un disegno dell'interpolazione



e qui sotto dell'interpolazione



Nel primo caso l'interpolante si ottiene tracciando i tratti di retta congiungenti i 2 punti del grafico a nodi corrispondenti a 2 nodi consecutivi e nel secondo caso si tracciano le parabole corrispondenti a pacchetti di 3 nodi consecutivi, come ad esempio:

$$x_0, x_1, x_2 \to 1^\circ \text{ pacchetto}$$
$$x_2, x_3, x_4 \to 2^\circ \text{ pacchetto}$$
$$x_4, x_5, x_6 \to 3^\circ \text{ pacchetto}$$

In generale per costruire una funzione interpolante di grado *s* a tratti si prendono pacchetti consecutivi con *s+1* nodi distinti, che individuano un unico polinomio interpolatore "locale" di grado *s*, con un nodo di raccordo tra 2 pacchetti consecutivi.
Prendiamo quindi questo insieme di vincoli, avendo *n* multiplo intero di *s*, quindi $n = k * s$:

pacchetti	tratti	interpolatori locali
$x_0, x_1, \dots, x_s$	$[x_0, x_s]$	$\prod_{s,1}$
$x_s, x_{s+1}, \dots, x_{2s}$	$[x_s, x_{2s}]$	$\prod_{s,2}$
$x_{2s}, x_{2s+1}, \dots, x_{3s}$	$[x_{2s}, x_{3s}]$	$\prod_{s,3}$
$\dots$	$\dots$	$\dots$
$x_{(k+1)\cdot s}, \dots, x_{ks}$	$[x_{(k-1)\cdot s}, x_{ks}]$	$\prod_{s,k}$

La funzione interpolante, definita come Π^s_c , è ottenuta incollando i nodi di raccordo dei vari polinomi interpolatori locali di grado s_i che sono unici, dicendo che anche la funzione polinomiale di grado "s" a tratti è *unica* essendo unici gli interpolatori locali e:

avremo che $x_j \in [x_{(i-1)s}, x_{is}]$ per un certo i e quindi

$$\forall j \quad \prod_s^c(x_j) = \prod_{s,i}(x_j) = y_j = f(x_j)$$

Osserviamo fra l'altro che per l'unicità se $f \in \mathbb{P}_m,\ m \le s \ \Rightarrow \prod_s^c = f$ (cioè se f è un polinomio di grado $\le s$, allora l'interpolante a tratti di grado s è quello stesso polinomio).
Rispetto all'interpolazione con un unico polinomio $\prod_n$ il cui grado n viene mandato ad ∞, l'aspetto chiave nella costruzione di $\prod_s^c$ è che s è fissato (e per infittire il campionamento si prende $h = \max\ \Delta x_i \to 0$, in modo da infittire in tutto l'intervallo $[a, b]$).
Il caso dei nodi equispaziati rientra nella costruzione con

$$x_i = a + ih,\ 0 \le i \le n,\ h = \frac{b-a}{n}$$

Quindi l'interpolazione polinomiale garantisce la convergenza uniforme, quindi la distanza tra f e gli interpolatori locali tendente a 0, per *h* tendente anch'esso a 0.
Enunciamo poi:

3.3.2 TEOREMA (convergenza uniforme dell'interpolazione polinomiale a tratti)

Siano $f \in C^{s+1}[a, b],\ s \ge 0$ e $\{x_i\} \subset [a, b]$ $n+1$ nodi distinti, con n multiplo di s.
Allora

$$\exists k_s > 0 : dist(f, \prod_s^c) \le k_s h^{s+1},\ h = \max\ \Delta x_i$$

Dimostrazione (eseguita per s=1, come interpolazione lineare a tratti):

Prendiamo il max. errore che dovrebbe andare a 0:

$$dist(f, \prod_1^c) = \max_{x \in [a,b]} |f(x) - \prod_1^c(x)|$$
$$= \max_{1 \le i \le n} \max_{x \in [x_{i-1}, x_i]} |f(x) - \prod_1^c(x)|$$

Scritto da Gabriel

definito come:

$$= \max_{1\le i\le n} \max_{x\in[x_{i-1},x_i]} |f(x) - \prod_{1,i}(x)|$$

Ora ricordando la stima dell'errore di interpolazione polinomiale a grado s ricavata dalla scorsa lezione

$$\max_{x\in[\alpha,\beta]} |f(x) - \prod_s(x)| \le \max_{x\in[\alpha,\beta]} |f^{(s+1)}(x)| \cdot \frac{h^{s+1}}{4(s+1)}$$

valida per $f \in C^{s+1}[\alpha,\beta]$ e $h = \frac{\beta-\alpha}{s}$, e applicandola per $s = 1$ e $[\alpha,\beta] = [x_{i-1}, x_i]$, otteniamo

$$\max_{x\in[x_{i-1},x_i]} |f(x) - \prod_{1,i}(x)| \le \max_{x\in[x_{i-1},x_i]} |f''(x)| \cdot \frac{\Delta^2 x_i}{8} \le M_{2,i}\frac{h^2}{8}, \quad M_{2i} = \max_{x\in[x_{i-1},x_i]} |f''(x)|$$

da cui

$$\begin{aligned} dist(f, \prod_1^c) &= \max_{1\le i\le n} \max_{x\in[x_{i-1},x_i]} |f(x) - \prod_{1,i}(x)| \\ &\le \frac{h^2}{8} \max_{1\le i\le n} M_{2,i} \\ &= \frac{M_2}{8}h^2 \end{aligned}$$

con $M_2 = \max_{x\in[a,b]}|f''(x)|$ (di nuovo perchè il max di $|f''(x)|$ su $[a,b]$ è il massimo dei max di $|f''(x)|$ sui singoli intervallini).

Parliamo poi della <u>stabilità dell'interpolazione polinomiale a tratti</u>, dove sappiamo che risulta essere instabile ma, per la costante di Lebesgue, quando questa risulta piccola l'interpolazione a tratti risulterà sostanzialmente *stabile*, per il coefficiente di amplificazione che dipende da "s" ma <u>non</u> da "n".

Infatti, detta $\prod_1^c$ l'interpolante a tratti costruita sui valori approssimati $\{\tilde{y}_i\}$, con $\max_j|y_j - \tilde{y}_j| \le \varepsilon$:

$$\begin{aligned} dist(\prod_s^c, \widetilde{\prod_s^c}) &= \max_{1\le i\le k} \max_{x\in[x_{(i-1)s},x_{is}]} \left|\prod_s^c(x) - \widetilde{\prod_s^c}(x)\right| \\ &= \max_{1\le i\le k} \max_{x\in[x_{(i-1)s},x_{is}]} \left|\prod_{s,i}(x) - \widetilde{\prod}_{s,i}(x)\right| \\ &\le \Lambda_s^{eq} \cdot \varepsilon \quad (\text{con } n = ks) \end{aligned}$$

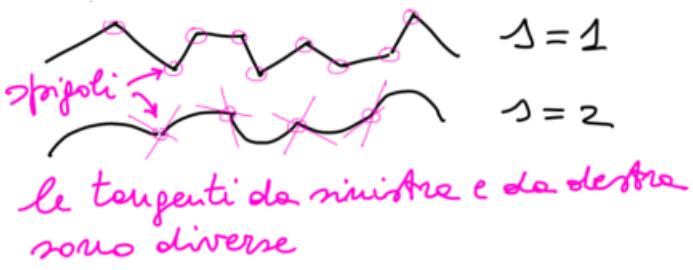
perchè

$$\max_{x\in[x_{(i-1)s},x_{is}]} \left|\prod_{s,i}(x) - \widetilde{\prod}_{s,i}(x)\right| \le \Lambda_s \cdot \varepsilon$$

dove Λ_s^{eq} è la costante di Lebesgue per grado s, cioè:

$$\Lambda_s^{eq} \approx \frac{2}{e} \cdot \frac{2^s}{s\log(s)}, \text{ con } s > 1 \text{ fissato}$$

L'interpolazione polinomiale a tratti risulta essere molto utile nella ricostruzione di funzioni di dati discreti, avendo garanzia di convergenza partendo da un campionamento, nell'approssimazione di integrali, della funzione integranda. In generale le derivate risultano essere diverse e quindi poco adatte alle applicazioni grafiche, dato che Π^s_c è solo continua, vedendo infatti le int. lineare/quadratica a tratti:

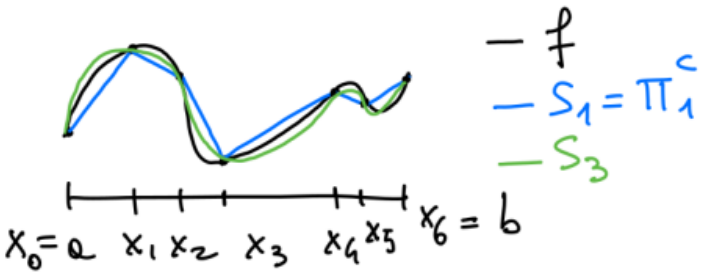


Nella grafica vogliamo interpolazioni possibilmente senza spigoli e, da questo, si diffonde l'interpolazione *spline*, descrivendo precisamente il metodo.

Data $f \in C[a,b]$ e $n+1$ punti $\{(x_i, y_i)\}_{0\le i\le n}$, $y_i = f(x_i)$, con $a = x_0 < x_1 < x_2 < \cdots < x_n = b$, una funzione polinomiale a tratti interpolante si chiama interpolante spline di grado k, indicata con $S_k(x)$, se valgono le seguenti condizioni:

1. $S_k(x_i) = y_i$, $0 \le i \le n$ (vincoli di interpolazione)
2. $S_k|_{I_i} \in \mathbb{P}_k$, $I_i = [x_i, x_{i+1}]$ e $0 \le i \le n-1$ (S_k ristretta all'i-esimo intervallino tra i nodi consecutivi è un polinomio di grado $\le k$)
3. $S_k \in C^{k-1}[a,b]$ (vincolo di regolarità)

Le *spline* in generale non sono polinomi ma *funzioni polinomiali a tratti*, con la differenza rispetto all'interpolazione standard che i nodi non sono raggruppati a pacchetti, dato che la spline è localmente un polinomio fra due nodi consecutivi, "incollando" tra di loro i vari polinomi locali con raccordi "lisci", quindi con le derivate nei punti di raccordo vincolate ad essere continue fino ad un certo ordine. Nel caso di k=3, definite come *spline cubiche*:



Innanzitutto osserviamo che in $1_i = [x_i, x_{i+1}]$ $S_3(x)$ è un polinomio cubico, cioè ha la forma

$$S_3|_{I_i} = p_{3,i} = a_{0,i} + a_{1,i}x + a_{2,i}x^2 + a_{3,i}x^3$$

Ci sono quindi 4 coefficienti incogniti per ciascuno degli n intervallini, quindi le incognite sono in tutto $4\,n$.
E i vincoli?

$$S_3(x_0) = p_{3,1}(x_0) = y_0$$
$$S_3(x_n) = p_{3,n}(x_n) = y_n$$
$$p_{3,i}(x_i) = y_i = p_{3,i+1}(x_i),\ 1 \le i \le n-1$$

questi vincoli assicurano interpolazione e continuità, restano i vincoli di regolarità

$$\left.\begin{aligned} p'_{3,i}(x_i) &= p'_{3,i+1}(x_i) \\ p''_{3,i}(x_i) &= p''_{3,i+1}(x_i) \end{aligned}\right\} 1 \le i \le n-1 \ \longleftarrow\ 2(n-1) \text{ vincoli}$$

In generale quindi avremo come vincoli, che diventano poi equazioni:

$$2 + 2(n-1) + 2(n-1) = 4n - 2$$

Le incognite sono i coefficienti degli "n" polinomi cubici, che entrano linearmente nelle corrispondenti equazioni, dovuto ai polinomi e loro derivate definite come combinazioni lineari dei monomi x^j e loro derivate, anch'esse spazi di funzioni derivabili. I vincoli diventano dunque *sistema lineare* di $4n - 2$ equazioni in $4n$ incognite.
Il sistema quindi non è determinato e, per trovare la soluzione, si impongono 2 vincoli aggiuntivi, facendo in modo il sistema diventi quadrato ($4n \times 4n$).

Ci sono vari set di coppie di condizioni aggiuntive che si può dimostrare che garantiscono una matrice $4n \times 4n$ non singolare (cioè invertibile; non faremo queste dimostrazioni).
Ad esempio le condizioni

$$S_3''(x_0) = p_{3,1}''(x_0) = 0 = p_{3,n}''(x_n) = S_3''(x_n)$$

(derivate seconde nulle agli estremi) oppure le condizioni

$$p_{3,1}'''(x_1) = p_{3,2}'''(x_1),\ p_{3,n-1}'''(x_{n-1}) = p_{3,n}'''(x_{n-1})$$

(S_3''' continua nel secondo e nel penultimo nodo).

L'ultima coppia di vincoli costruisce le *spline naturali*, implementate matematicamente dando come input una coppia di vettori (in Matlab funzione *spline*). Noi vogliamo quindi capire la convergenza di questa costruzione algebrica. Enunciamo:

3.3.5 TEOREMA (convergenza uniforme dell'interpolazione spline cubica)

Siano $f \in C^4[a,b]$ e $\{x_i\} \subset [a,b]$ $n+1$ nodi equispaziati $(h = \frac{b-a}{n})$

allora

$$\exists k_{3,j} > 0 :\ dist_{0 \le j \le 3}(f^{(j)}, S_3^{(j)}) \le k_{3,j} \cdot h^{4-j}$$

Ma c'è di più: $S_3 \in C^2[a,b]$ per costruzione, e risulta che

$$dist(f', S_3') \le k_{3,1} \cdot h^3$$
$$dist(f'', S_3'') \le k_{3,2} \cdot h^2$$

e addirittura

$$dist(f''', S_3''') \le k_{3,3} \cdot h$$

cioè non solo S_3 converge uniformemente ad f, ma anche le sue derivate fino alla terza convergono alla corrispondente derivata di f (si noti che S_3 è cubica a tratti, S_3' è quadratica a tratti, S_3'' è lineare a tratti e S_3''' è costante a tratti).
Per capirlo basta pensare che localmente si tratta di derivare un polinomio di grado 3, $p_{3,i}(x)$ $1 \le i \le n$.

Localmente si deriva un polinomio di grado 3 e mentre S_3 approssima *f* interpolandola, le sue derivate approssimano le f^j, con *j* compreso tra 1 e 3, ma non le interpolano (costruita per essere "liscia", non per interpolare). In merito alla stabilità abbiamo che:

$$\max |y_i - \tilde{y}_i| \le \varepsilon$$

si ha

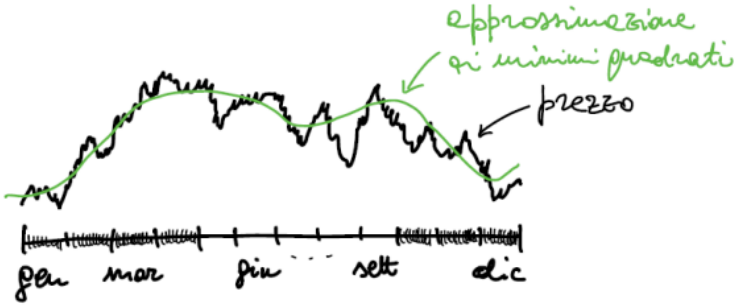
$$dist(\tilde{S}_3, S_3) \le c \cdot \varepsilon$$

per una opportuna costante $c > 0$.

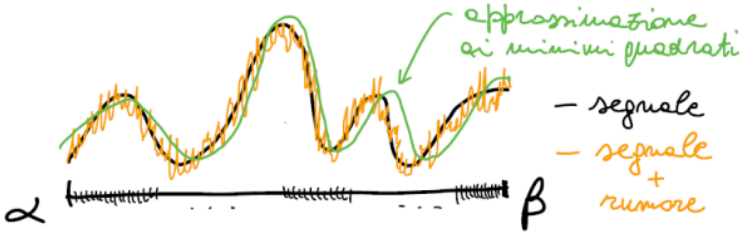
Le derivate di S_3 non sono stabili, data l'instabilità che si eredita dalle derivazioni e quindi l'interpolazione a tratti standard/spline risolve il problema della convergenza su distribuzioni arbitrarie, purché h tenda a 0 e resti limitato il rapporto tra la differenza dei nodi non equispaziati.
In generale quindi una volta scelto il grado k-esimo dell'interpolazione a tratti, l'errore resta generalmente di ordine *n+1*.

Lezione 15 – Approssimazione polinomiale ai minimi quadrati

Vediamo l'idea di rappresentare in modo discreto una funzione con la tecnica dei *minimi quadrati*, che ha forti connessioni con la statistica nel campo dei metodi di regressione. Ecco l'esempio grafico non regolare:



tale poi da avere normalmente una richiesta di *regolarizzare* il grafico cercando una media/trend dei dati:



dove in realtà il segnale è sempre disturbato da rumore con un passo molto piccolo e quindi, normalmente, ci potrebbe essere una richiesta di *filtrare* il rumore. Supponiamo il rumore sia funzione derivabile, quindi la immaginiamo come somma della derivata del segnale con la derivata del rumore, quindi:
$\tilde{s}' = s' + r'$ dove $|r'| \gg |s'|$,

Vogliamo quindi filtrare il rumore per far emergere il segnale sottostante e così regolarizzare il segnale misurato, con la tecnica dello *smoothing* (regolarizzazione).
Immaginiamo quindi di avere un gran numero di dati, formato da coppie di dati $\{(x_i, y_i)\}$, $y_i = f(x_i)$, $1<=i<=N$:

Fissato un grado m (tipicamente con un $m \ll N$) l'approssimazione polinomiale ai minimi quadrati consiste nel cercare un polinomio $L_m \in \mathbb{P}_m$ (cioè di grado $\leq m$) tale che la somma degli scarti quadratici sia minima

$$\sum_{i=1}^{N}(y_i - L_m(x_i))^2 = \min_{p\in\mathbb{P}_m}\sum_{i=1}^{N}\underbrace{(y_i - p(x_i))^2}_{i-esimo\ scarto\ quadratico}$$

Ora, indicheremo con $\phi(a)$ la funzione di $m+1$ variabili

$$\phi(a) = \sum_{i=1}^{N}\left(y_i - \sum_{j=0}^{m} a_j x_i^j\right)^2$$

cioè la somma degli scarti quadratici del polinomio con coefficiente $\{a_j\}$ calcolato negli $\{x_i\}$ rispetto ai valori misurati $\{y_i\}$.

Questa funzione $\phi(a)$ è in realtà essa stessa un polinomio quadratico (di grado 2) nelle variabili $\{a_j\}$. Infatti sviluppando i quadrati si ha che

$$\left(y_i - \sum_{j=0}^{m} a_j x_i^j\right)^2 = y_i^2 - 2y_i\sum_{j=0}^{m} a_j x_i^j + \left(\sum_{j=0}^{m} a_j x_i^j\right)^2$$

e quindi è chiaro che in ciascuno scarto quadratico le variabili $\{a_j\}$ compaiano come a_j, a_j^2 e $a_j a_k$.
Facciamo l'esempio di $m = 1$ (l'approssimazione lineare ai minimi quadrati o rette dei minimi quadrati): in questo caso $p \in \mathbb{P}_1$ ha la forma

$$p(x) = a_0 + a_1 x$$

e ϕ diventa

$$\phi(a) = \phi(a_0, a_1) = \sum_{i=1}^{N}\left(y_i - (a_0 + a_1 x_i)\right)^2 = \sum_{i=1}^{N}\left(y_i^2 - 2y_i(a_0 + a_1 x_i) + (a_0 + a_1 x_i)^2\right)$$

ma

$$(a_0 + a_1 x_i)^2 = a_0^2 + 2a_0 a_1 x_i + a_1^2 x_i^2$$

da cui

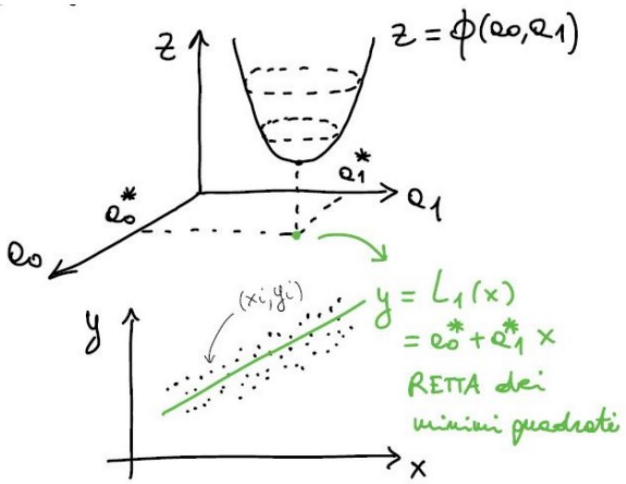
$$\begin{aligned}&(y_i - (a_0 + a_1 x_i))^2 = \\ &= y_i^2 - 2y_i(a_0 + a_1 x_i) + a_0^2 + 2a_0 a_1 x_i + a_1^2 x_i^2 \\ &= y_i^2 - 2y_i a_0 - 2x_i y_i a_1 + a_0^2 + 2a_0 a_1 x_i + a_1^2 x_i^2\end{aligned}$$

che è un polinomio di grado 2 in a_0 e a_1 con coefficienti che dipendono da x_i e y_i.
Quindi alla fine

$$\phi(a) = \sum y_i^2 - \left(2\sum y_i\right)a_0 - 2\left(\sum x_i y_i\right)a_1 + Na_0^2 + 2\left(\sum x_i\right)a_0 a_1 + \left(\sum x_i^2\right)a_1^2$$

è un polinomio di grado 2 in a_0 e a_1.

L'interpretazione geometrica è quindi cercare il punto del piano (a_0 , a_1), corrispondente al vertice di un paraboloide convesso.



Definiamo quindi una matrice di Vandermonde rettangolare:

$$V = (v_{ij}) = (x_i^j)$$

con $1 \le i \le N$ e $0 \le j \le m$ cioè

$$V = \begin{pmatrix} 1 & x_1 & x_1^2 & \dots & x_1^m \\ 1 & x_2 & x_2^2 & \dots & x_2^m \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_N & x_N^2 & \dots & x_N^m \end{pmatrix} \in \mathbb{R}^{N\times(m+1)}$$

si ha che

$$\begin{aligned}\phi(a) &= \sum_{i=1}^{N}(y_i - (Va)_i)^2 \\ &= (y - Va, y - Va)\end{aligned}$$

dove $(Va)_i$ indica l'elemento i-esimo del prodotto matrice-vettore

$$Va = V \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{pmatrix} = \left\{\sum_j v_{ij} \cdot a_j\right\} = \left\{\sum_j x_i^j a_j\right\}$$

e (u, v) indica il prodotto scalare di due vettori di $\mathbb{R}^N$, cioè

$$(u, v) = \sum_{i=1}^{N} u_i \cdot v_i$$

$$(u, u) = \sum_{i=1}^{N} u_i^2$$

intendendo quest'ultimo come quadrato della lunghezza del vettore *u* e si pensa a minimizzare la lunghezza del vettore *y – Va*.

Scritto da Gabriel

3.4.2 TEOREMA (sistema delle equazioni "normali" per i minimi quadrati)

Dati N punti $\{(x_i, y_i)\}$, $y_i = f(x_i)$, $1 \le i \le N$ e $m < N$, il vettore $a \in \mathbb{R}^{m+1}$ minimizza $\phi(a) = \sum_{i=1}^{N}(y_i - \sum_{j=0}^{m} a_j \cdot x_i^j)^2 \iff$ risolve il sistema $V^tVa = V^ty$.
Dove V^t è la trasposta di V.

intendendo il sistema come *sistema delle equazioni normali*, con dimensione (m+1) x (m+1), infatti:

$$V \in \mathbb{R}^{N\times(m+1)}, \quad V^t \in \mathbb{R}^{(m+1)\times N}, \quad y \in \mathbb{R}^N$$

e quindi:

$$V^tV \in \mathbb{R}^{(m+1)\times(m+1)} \text{ e } V^ty \in \mathbb{R}^{m+1}$$

(qualunque sia il numero di dati: quando si cerca la retta dei minimi quadrati, $m = 1$, possono esserci 100, 1000, 100000 dati ma il sistema è sempre 2×2 perchè si cercano 2 coefficienti).

Dimostrazione

Dire che $a \in \mathbb{R}^{m+1}$ è di minimo (assoluto) per $\phi(a)$ equivale a dire che:

$$\phi(a+b) \ge \phi(a) \quad \forall b \in \mathbb{R}^{m+1}$$

ma

$$\begin{aligned}\phi(a+b) &= (y - V(a+b),\, y - V(a+b)) \\ &= (y - Va - Vb,\, y - Va - Vb) \\ &= (y - Va,\, y - Va) + (y - Va,\, -Vb) + (-Vb,\, y - Va) + (-Vb,\, -Vb) \\ &= \phi(a) + 2(Va - y,\, Vb) + (Vb,\, Vb) \\ &= \phi(a) + 2(V^t(Va - y),\, b) + (Vb,\, Vb)\end{aligned}$$

usando le seguenti proprietà del prodotto scalare (indicato come $(u,v)_n$,con u^tv che indica i vettori colonna):

1. $(u, v)_n = (v, u)_n \quad u, v, w \in \mathbb{R}^n$
2. $(\alpha u, v)_n = \alpha(u, v)_n \quad \alpha \in \mathbb{R}$
3. $(u + v, w)_n = (u, w)_n + (v, w)_n$
4. $(u, Az)_n = (A^tu, z)_k \quad u \in \mathbb{R}^n,\ z \in \mathbb{R}^k,\ A \in \mathbb{R}^{n\times k}$

Vediamo prima la dimostrazione dell'implicazione "←", assumendo che $V^tVa=V^ty$ con:

$$V^tVa - V^ty = V^t(Va - y) = 0 \quad \text{e} \quad (V^t(Va - y),\, b) = \underset{\substack{\| \\ \text{vettore nullo in } \mathbb{R}^{m+1}}}{(0, b)} = 0$$

da cui:

$$\phi(a+b) = \phi(a) + \underset{\substack{\| \\ \sum_{i=1}^{N}(Vb)_i^2 \ge 0}}{(Vb,\, Vb)} \ge \phi(a) \quad b \in \mathbb{R}^{m+1}$$

e poi la dimostrazione dell'implicazione "→", assumendo che:

$$\phi(a+b) \ge \phi(a) \quad \forall b \in \mathbb{R}^{m+1}$$

Allora:

$$\phi(a+b) = \phi(a) + 2(V^t(Va - y),\, b) + (Vb, Vb) \ge \phi(a) \quad \forall b$$

Cioè:

$$2(V^t(Va - y),\, b) + (Vb, Vb) \ge 0 \quad \forall b$$

Prendiamo $b = \varepsilon v$, con v versore (cioè vettore di lunghezza 1, $(v, v) = 1$). Si ha:

$$\begin{aligned}&2(V^t(Va - y),\, \varepsilon v) + (V(\varepsilon v),\, V(\varepsilon v)) = \\ &= 2\varepsilon(V^t(Va - y),\, v) + \varepsilon^2(Vv, Vv) \ge 0 \quad \forall \varepsilon \ge 0 \text{ e } \forall v\end{aligned}$$

Dividendo per $\varepsilon > 0$:

$$2(V^t(Va - y),\, v) + \varepsilon(Vv, Vv) \ge 0 \quad \forall \varepsilon \text{ e } \forall v$$

Scritto da Gabriel

Per $\varepsilon \to 0$ la disuguaglianza viene mantenuta, ottenendo:

$$(V^t(Va-y),\, v) \geq 0 \quad \forall v$$

Ma se vale $\forall$ versore, possiamo prendere $-v$ al posto di v e otteniamo:

$$(V^t(Va-y),\, -v) = -(V^t(Va-y),\, v) \geq 0 \quad \forall v$$

Cioè $(V^t(Va-y),\, v) \leq 0,\ \forall v$. Siccome $(V^t(Va-y),\, v)$ risulta sia ≥ 0 che ≤ 0, allora è 0:

$$(V^t(Va-y),\, v) = 0 \quad \forall v$$

Ma chi è l'unico vettore ortogonale a tutti i vettori? É il vettore nullo, cioè

$$V^t(Va-y) = 0 \iff V^tVa = V^ty$$

Ovvero a è soluzione del sistema delle equazioni normali.

Ricordiamo le proprietà della matrice quadrata V^tV.

Innanzitutto osserviamo che è simmetrica (ricordando che $(AB)^t = B^tA^t\ \forall A, B$ matrici compatibili per il prodotto), infatti $(V^tV)^t = V^t(V^t)^t = V^tV$. Inoltre V^tV è semidefinita positiva, cioè $(V^tVv, v) \geq 0\ \forall v \in \mathbb{R}^{m+1}$.
Infatti

$$(V^tVv, v)_{m+1} = (Vv, (V^t)^tv)_N = (Vv, Vv)_N \geq 0 \quad \forall v$$

È noto dall'algebra lineare che allora V^tV ha tutti gli autovalori ≥ 0: ci interessa sapere quando sono tutti > 0, cioè quando è definita positiva (nel qual caso è non singolare, quindi invertibile e il sistema delle equazioni normali ha soluzione unica). Ricordiamo che definita positiva significa:

$$\begin{aligned} (V^tVv, v) &\geq 0 \qquad \forall v \\ (V^tVv, v) &= 0 \qquad \text{solo se } v = 0 \end{aligned}$$

Abbiamo appena visto che
$(V^tVv, v) = (Vv, Vv)$.
Ora $(Vv, Vv) = 0 \iff Vv = 0$ quindi $v = 0$ se V ha rango massimo ($rango(V) = m+1$), cioè se le colonne di V sono linearmente indipendenti; ricordiamo che Vv si può interpretare come combinazione lineare delle colonne di V con coefficenti gli elementi di v:

$$v = (\alpha_1, ..., \alpha_{m+1})^t \Rightarrow Vv = \sum_{j=1}^{m+1} \alpha_j C_j(V)$$

Per garantire che rango(V) = m+1, basta avere m+1 punti distinti tra i nodi di campionamento.

Supponiamo infatti che siano i primi $m+1$, cioè $x_1, ..., x_{m+1}$ con $x_i \neq x_j,\ i \neq j,\ 1 \leq i, j \leq m+1$
(altrimenti basta riordinarli)

$$V = \begin{pmatrix} 1 & x_1 & x_1^2 & \dots & x_1^m \\ 1 & x_2 & x_2^2 & \dots & x_2^m \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{m+1} & x_{m+1}^2 & \dots & x_{m+1}^m \\ 1 & x_{m+2} & x_{m+2}^2 & \dots & x_{m+2}^m \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_N & x_N^2 & \dots & x_N^m \end{pmatrix} \text{ Vandermonde di interpolazione} = U \in \mathbb{R}^{(m+1)\times(m+1)}$$

La sottomatrice $V \in \mathbb{R}^{(m+1)(m+1)}$ nel riquadro è una matrice di Vandermonde

dunque è non singolare e il rango della sottomatrice sarà *m+1*, quindi le *m+1* colonne sono linearmente indipendenti. Se fossero linearmente dipendenti, la proprietà sarebbe valida per le sottocolonne formate dai primi *m+1* elementi, infatti se:

$$\exists \alpha \neq 0 : \sum_{j=1}^{m+1} \alpha_j \mathcal{C}_j(V) = 0$$

allora

$$\sum_{j=1}^{m+1} \alpha_j \mathcal{C}_j(U) = 0$$

perchè

$$\mathcal{C}_j(V) = \begin{pmatrix} x_1^{j-1} \\ \vdots \\ x_{m+1}^{j-1} \\ x_{m+2}^{j-1} \\ \vdots \\ x_N^{j-1} \end{pmatrix} \quad \text{dove} \quad \begin{pmatrix} x_1^{j-1} \\ \vdots \\ x_{m+1}^{j-1} \end{pmatrix} = \mathcal{C}_j(U)$$

Scritto da Gabriel

Dunque possiamo calcolare esplicitamente gli elementi della matrice V^tV, dipendenti solo dagli $\{x_i\}$ e dal vettore termine noto V^ty, dipendente anche dagli $\{y_i\}$.

$$V^tV = \begin{pmatrix} 1 & 1 & \dots & \dots & 1 \\ x_1 & x_2 & \dots & \dots & x_N \\ \vdots & \vdots & & & \vdots \\ x_1^m & x_2^m & \dots & \dots & x_N^m \end{pmatrix} \cdot \begin{pmatrix} 1 & x_1 & \dots & x_1^m \\ 1 & x_2 & \dots & x_2^m \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ 1 & x_N & \dots & x_N^m \end{pmatrix}$$

$$= \begin{pmatrix} N & \sum x_i & \dots & \sum x_i^m \\ \sum x_i & \sum x_i^2 & \dots & \sum x_i^{m+1} \\ \vdots & \vdots & & \vdots \\ \sum x_i^m & \sum x_i^{m+1} & \dots & \sum x_i^{2m} \end{pmatrix} \in \mathbb{R}^{(m+1)\times(m+1)}$$

Dove $V^t \in \mathbb{R}^{(m+1)\times N}$ e $V \in \mathbb{R}^{N\times(m+1)}$

$$V^ty = \begin{pmatrix} 1 & 1 & \dots & \dots & 1 \\ x_1 & x_2 & \dots & \dots & x_N \\ \vdots & \vdots & & & \vdots \\ x_1^m & x_2^m & \dots & \dots & x_N^m \end{pmatrix} \cdot \begin{pmatrix} y_1 \\ \vdots \\ \vdots \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} \sum y_i \\ \sum x_i \cdot y_i \\ \vdots \\ \sum x_i^m \cdot y_i \end{pmatrix}$$

Dove $y \in \mathbb{R}^N$ e $V^t \cdot y \in \mathbb{R}^{m+1}$

Vediamo quindi il sistema delle equazioni normali:

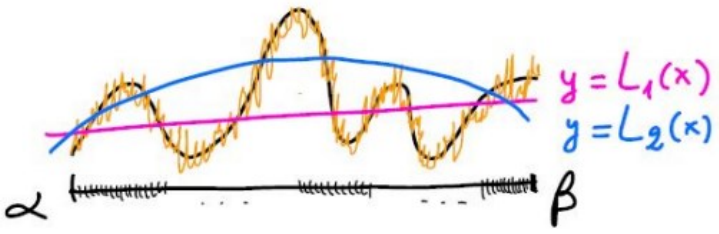
- con m=1 (retta dei minimi quadrati)

$$\begin{pmatrix} N & \sum x_i \\ \sum x_i & \sum x_i^2 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} \sum y_i \\ \sum x_i \cdot y_i \end{pmatrix}$$

- con m=2 (parabola dei minimi quadrati)

$$\begin{pmatrix} N & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} \sum y_i \\ \sum x_i \cdot y_i \\ \sum x_i^2 \cdot y_i \end{pmatrix}$$

Per la scelta del grado "m" del polinomio L_m dei minimi quadrati, come detto, si prende m << N, con scelta suggerita qualitativamente, dal grafico e dalla grandezza di min $\phi(a)$ al variare di m o sull'errore di approssimazione dei dati campionati. Tornando all'esempio iniziale del segnale con rumore, le approssimazioni con m=1 ed m=2 sono inadeguate alla ricostruzione del segnale, aspettandosi sia efficace solo un grado m > 6.



Capitolo 4: Integrazione numerica e derivazione numerica

Lezione 16 - Integrazione numerica (formula quadratura): stabilità dell'operatore di integrazione, formule di quadratura algebriche e composte, convergenza e stabilità, formule a pesi positivi

Ci occupiamo ora di studiare le funzioni partendo da un campionamento finito parlando di derivazione e di integrazione. Discutiamo quindi la *stabilità dell'integrazione numerica*.
Osserviamo infatti che l'operatore di integrazione:

$$I : f \longmapsto I(f) = \int_a^b f(x)dx \quad \in \mathbb{R}$$

risulta essere stabile, dato che controllando gli errori sulla funzione si controllano anche sull'integrale.

Infatti se $\tilde{f} \approx f$ con

$$dist(f, \tilde{f}) = \max_{x \in [a,b]} |f(x) - \tilde{f}(x)| \leq \varepsilon$$

allora

$$\begin{aligned}
\left|I(f) - I(\tilde{f})\right| &= \left|\int_a^b f(x)dx - \int_a^b \tilde{f}(x)dx\right| \\
&= \left|\int_a^b (f(x) - \tilde{f}(x))dx\right| = \left|I(f - \tilde{f})\right| \\
&\leq \int_a^b \left|f(x) - \tilde{f}(x)\right|dx = I\left(\left|f - \tilde{f}\right|\right) \\
&\leq \int_a^b dist(f, \tilde{f})dx \\
&= dist(f, \tilde{f}) \int_a^b 1dx \\
&= dist(f, \tilde{f})(b - a) \\
&\leq \varepsilon(b - a)
\end{aligned}$$

avendo usato la linearità dell'operatore di integrazione:

$$I(\alpha f + \beta g) = \alpha I(f) + \beta I(g) \quad \forall f, g \in C[a, b], \quad \alpha, \beta \in \mathbb{R}$$

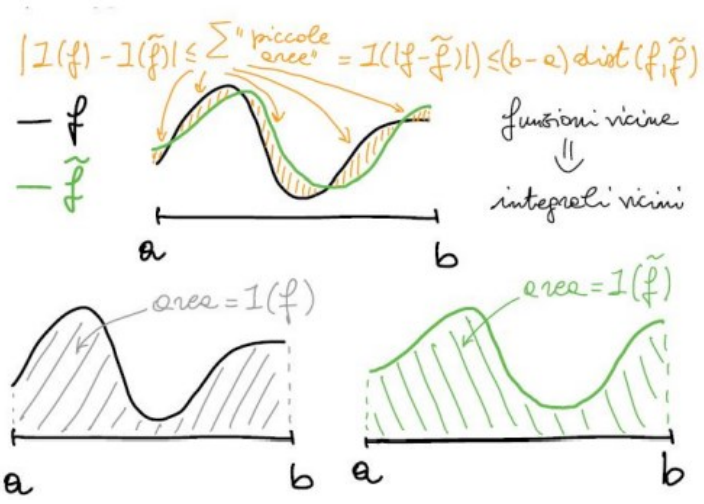
e la disuguaglianza fondamentale:

$$|I(f)| \leq I(|f|)$$

ragionando quindi che il *problema è stabile*, infatti errori piccoli sulla funzione portano ad errori piccoli nell'integrale, diversamente dalla derivazione, dove anche valori vicini danno risultati ben distanti.

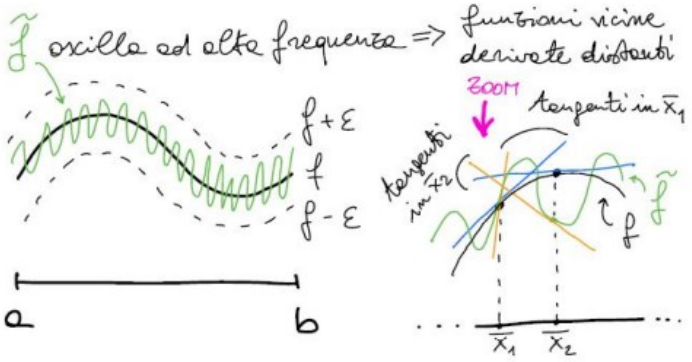
Graficamente possiamo notare che *l'integrale è stabile* mentre *la derivata è instabile*.

STABILITÀ DELL'INTEGRALE



Scritto da Gabriel

INSTABILITÀ DELLA DERIVATA



Possiamo introdurre il tema delle formule di quadratura, per il calcolo approssimato degli integrali, dato che "f" non è nota se non tramite un campionamento discreto oppure non si sa se si possa calcolare un integrale analiticamente usando il teorema fondamentale del calcolo, che citiamo:

$$\int_a^b f(x)dx = F(b) - F(a)$$

dove F è una primitiva di f, cioè $F'(x) = f(x)$; come noto, due primitive differiscono di una costante

$$\{F : F' = f\} = \{F = G + c,\ G' = f,\ c \in \mathbb{R}\}$$

dove G è una primitiva fissata, ad esempio la funzione integrale

$$G(x) = \int_a^x f(t)dt,\ \ x \in [a, b]$$

Non tutte le funzioni hanno primitive esprimibili tramite funzioni elementari (esempio, la funzione $f(x) = e^{-x^2}$, la "error function"). Un'idea semplice per il calcolo approssimato di integrali è di sostituire alla funzione integranda "f" in C[a,b] una funzione interpolante f_n su n+1 nodi distinti $\{x_i\}$ in [a,b].

$$f_n(x_i) = y_i = f(x_i), \quad 0 \le i \le n$$

Utilizzando i due tipi di interpolazione visti in precedenza, si ottengono 2 famiglie di formule definite come *formule di quadratura*, o *<u>formule algebriche/interpolatorie</u>* con

$$f_n(x) = \prod_n(x)$$

cioè il polinomio interpolatore di grado $\le n$, mentre per

$$f_n(x) = \prod_s^c(x)$$

e con la funzione polinomiale composta a tratti di grado locale "s" abbiamo le *<u>formule composte</u>*.
Vediamo quindi che entrambi i tipi di formule:

$$I_n(f) = I(f_n) = \int_a^b f_n(x)dx$$

hanno la forma di somma pesata dei valori campionati

$$I_n(f) = \sum_{i=0}^{n} w_i f(x_i)$$

Per quanto riguarda le *formule algebriche sotto forma di somma pesata*, ricordando la forma di Lagrange dell'interpolatore per le formule $I(f_n) = I(\Pi_n)$:

$$\begin{aligned} I_n(f) = I(\prod_n) &= \int_a^b \prod_n(x)dx \\ &= \int_a^b \left(\sum_{i=0}^{n} y_i l_i(x)\right) dx \\ &= \sum_{i=0}^{n} \int_a^b y_i l_i(x)dx \\ &= \sum_{i=0}^{n} w_i y_i \quad \text{con} \quad \overbrace{w_i}^{PESI} = \int_a^b l_i(x)dx, \quad 0 \le i \le n \end{aligned}$$

con i pesi che sono gli integrali dei polinomi elementari di Lagrange, dipendenti solo dai nodi.

Andiamo poi con le *formule composte sotto forma di somma pesata*, con i nodi n = k*s a pacchetti di s+1 con nodo di raccordo:

$$\begin{aligned} a = &x_0 < x_1 < \ldots < x_s \\ &x_s < \ldots < x_{2s} \\ &\vdots \\ &x_{(k-1)s} < \ldots < x_{ks} = x_n = b \end{aligned}$$

possiamo scrivere

$$\begin{aligned} I_n(f) = I(\prod_s^c) &= \int_a^b \prod_s^c(x)dx \\ &= \sum_{j=1}^{k} \int_{x_{(j-1)s}}^{x_{js}} \prod_s^c(x)dx \\ &= \sum_{j=1}^{k} \int_{x_{(j-1)s}}^{x_{js}} \prod_{s,j}(x)dx \end{aligned}$$

(dove $\prod_{s,j}$ è l'interpolazione locale di grado $\le s$ sul tratto $[x_{(j-1)s}, x_{js}]$ cioè $\prod_{s,j}(x) = \sum_{i=(j-1)s}^{js} y_i l_{i,j}(x)$, i-esimo pol. elementare relativo al pacchetto $x_{(j-1)s,\ldots,x_{js}}$)

$$\begin{aligned} &= \sum_{j=1}^{k} \int_{x_{(j-1)s}}^{x_{js}} \left(\sum_{i=(j-1)s}^{js} y_i l_{i,j}(x)\right) dx \\ &= \sum_{j=1}^{k} \sum_{i=(j-1)s}^{js} y_i \int_{x_{(j-1)s}}^{x_{js}} l_{i,j}(x)dx \\ &= \sum_{j=1}^{k} \sum_{i=(j-1)s}^{js} y_i w_{i,j} \end{aligned}$$

dove

$$w_{i,j} = \int_{x_{(j-1)s}}^{x_{js}} \underbrace{l_{i,j}(x)}_{\substack{\uparrow \\ \textbf{PESI} \\ \text{(dipendono} \\ \text{solo dai nodi)}}} , \quad (j-1)s \le i \le js \quad 1 \le j \le k$$

Ciascun valore compare una volta tranne per i nodi di raccordo, dato che lì compare 2 volte e dunque i 2 pesi corrispondenti vanno sommati.

Riarrangiando la doppia somma si arriva quindi a

$$I_n(f) = I(\prod_s^c) = \sum_{i=0}^{n} w_i \cdot y_i$$

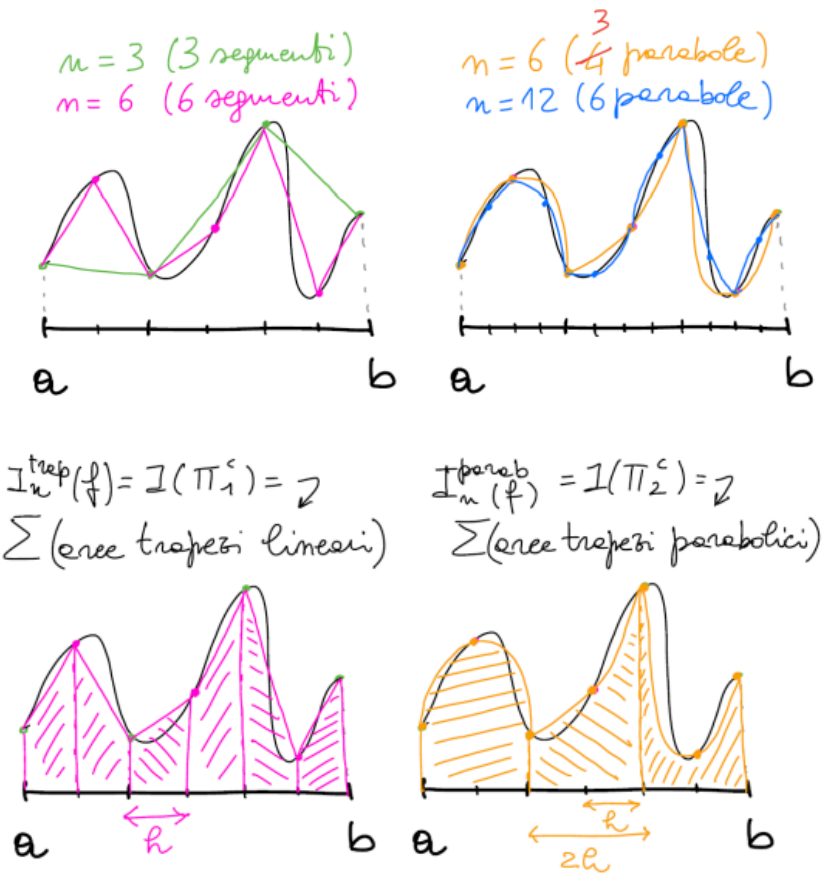
dove $w_i = w_{i,j}$ per $(j-1)\cdot s < i < j \cdot s$

$$w_i = w_{i,j} + w_{i,(j+1)}, \quad i = j \cdot s, \quad 1 \le j \le k-1$$

Facciamo quindi due esempi per il caso dei nodi equispaziati:

1. $s = 1$: formula composta dei TRAPEZI (generata dall'interpolazione lineare a tratti).

2. $s = 2$: formula composta delle PARABOLE (generata dall'interpolazione quadratica a tratti).



Per s=1, l'integrale (area sotto la curva di grafico di "f") viene approssimato dalla somma delle aree dei trapezi lineari corrispondenti all'interpolante lineare a tratti.

Osservando che l'i-esimo trapezio ha altezza $h = (b-a)/n$ e basi $f(x_{i-1})$ e $f(x_i)$, $1 \le i \le n$, si ha area trapezio i-esimo $= \frac{h}{2} \cdot (f(x_{i-1}) + f(x_i))$ e quindi

$$\begin{aligned} I_n^{trap}(f) &= I(\prod_1^c) \\ &= \int_a^b \prod_1^c(x)dx \\ &= \sum(\text{aree trapezi}) \\ &= \overbrace{\frac{h}{2} \cdot (f(x_0) + f(x_1))}^{\text{area trapezio 1}} + \overbrace{\frac{h}{2} \cdot (f(x_1) + f(x_2))}^{\text{area trapezio 2}} + \ldots + \\ &+ \underbrace{\frac{h}{2} \cdot (f(x_{n-2}) + f(x_{n-1}))}_{\text{area trapezio (n-1)-esimo}} + \underbrace{\frac{h}{2} \cdot (f(x_{n-1}) + f(x_n))}_{\text{area trapezio n-esimo}} \\ &= \frac{h}{2}(f(x_0) + f(x_n)) + \sum_{i=1}^{n-1} h \cdot f(x_i) \end{aligned}$$

(perché ogni nodo interno x_i, $1 \le i \le n-1$, compare in 2 trapezi consecutivi, cioè la FORMULA DEI TRAPEZI)

$$I_n(f) = \sum_{i=0}^{n} w_i \cdot f(x_i), \text{ con } w_i = \begin{cases} \frac{h}{2}, & i = 0, n \\ h, & 1 \le i \le n-1 \end{cases}$$

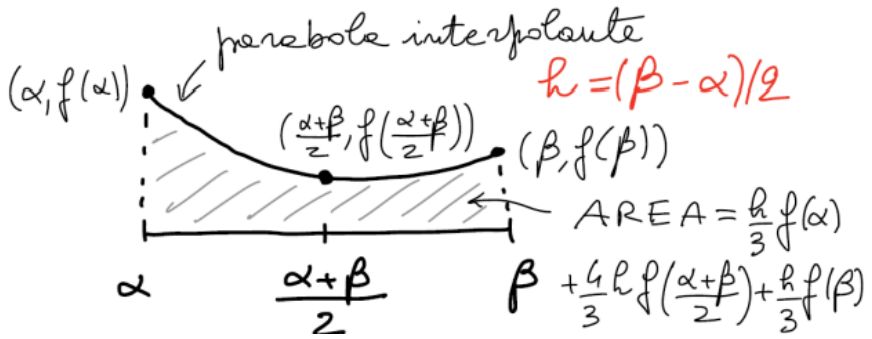
Per s=2 integrando la funzione quadratica a tratti Π_2^c si ottiene la *formula delle parabole/di Simpson*.

$$I_n^{parab}(f) = I\left(\prod_2^c\right) = \sum(\text{aree trapezi parabolici}) = \sum_{i=0}^{n} w_i \cdot f(x_i), \text{ con } w_i =$$

$$\begin{cases} h/3, & i = 0, n \text{ pari} \\ 4h/3, & i \text{ dispari} \\ 2h/3, & i \text{ pari } 2 \le i \le n-2 \end{cases}$$

La formula si ottiene calcolando l'integrale di una interpolante di grado 2 su $[\alpha, \beta]$ costruita con i valori $f(\alpha)$, $f(\beta)$, $f((\alpha+\beta)/2)$

$$\int_\alpha^\beta \prod_2(x)dx = \frac{h}{3} \cdot f(\alpha) + \frac{4}{3} \cdot h \cdot f\left(\frac{\alpha+\beta}{2}\right) + \frac{h}{3} \cdot f(\beta)$$



osservando che per entrambe le formule di approssimazione si hanno *pesi positivi*, dato che sono *somme pesate*. Analizziamo poi la *convergenza delle formule di quadratura*, chiedendosi per quali distribuzioni di nodi ed integrande "f" si ha:

$$\lim_{n \to \infty} I_n(f) = I(f)?$$

dove $I_n(f) = \sum_{i=0}^{n} w_i \cdot y_i$, $y_i = f(x_i)$, è una successione di formule di quadratura.
Visto che abbiamo costruito le formule usando una funzione interpolante

$$f_n(x) : f_n(x_i) = f(x_i), \quad 0 \leq i \leq n$$

cioè $I_n(f) = I(f_n)$
Ci viene in aiuto la stima fondamentale ricavata a inizio lezione (ponendo $\tilde{f} = f_n$)

$$\begin{aligned} |I(f) - I_n(f)| &= |I(f) - I(f_n)| \\ &= |I(f - f_n)| \\ &\leq I(|f - f_n|) \\ &\leq (b-a) \cdot dist(f, f_n) \end{aligned}$$

Per le *formule algebriche*:
In questo caso $f_n = \prod_n$ quindi

$$|I(f) - I_n(f)| \leq (b-a) dist(f, \prod_n)$$

aspettandoci problemi per funzioni su nodi equispaziati in quanto potrebbero essere non regolari (es. Runge). Possono però convergere per particolari funzioni, ad esempio le *formule di Newton-Cotes*, algebriche su nodi equispaziati non convergenti. Usando però le formule interpretando il polinomio su nodi di tipo Chebyshev abbiamo che:

$$dist(f, \prod_n^{Cheb}) \leq c_k \frac{log(n)}{n^k}$$

per $f \in C^k[a,b]$, e quindi tali formule sono sicuramente convergenti per $f \in C^k[a,b], \quad k > 0$.

Per quanto riguarda invece le *formule composte*:
La situazione cambia completamente con le formule composte, ottenute come $I_n(f) = I(\prod_s^c)$, con n multiplo di s.
Infatti per tali formule s è fissato e

$$|I(f) - I_n(f)| \leq (b-a) \cdot dist(f, \prod_s^c) \leq (b-a) k_s \cdot h^{s+1} \qquad \text{se } f \in C^{s+1}[a,b]$$

con $h = max\ \Delta x_i$.
Dunque per qualsiasi distribuzione di nodi per cui h → 0 (anche nodi non equispaziati), l'errore di interpolazione a tratti va anch'esso a 0 e le formule sono sempre convergenti, tuttavia l'errore rimane comunque dipendente dalla linearità dell'interpolatore. Avremo comunque altre formule, tipo quella *dei trapezi* con ordine h^2 oppure la formula *della parabola* con ordine h^4.

Discutiamo la *stabilità delle formule di quadratura*.
Sapendo che i valori campionati non sono mai noti esattamente, immaginiamo di avere a disposizione dei valori perturbati $\tilde{y}_i \approx y_i,$

assumendo di avere una stima: $\max_i |y_i - \tilde{y}_i| \leq \varepsilon$

Dunque ci chiediamo: qual è la risposta della formula di quadratura agli errori sui dati?
Quindi come si stima in funzione di ε:
$|I_n(f) - \tilde{I}_n(f)|$, dove $\tilde{I}_n(f) = \sum_{i=0}^{n} w_i \tilde{y}_i$?

La stima è semplice:

$$\begin{aligned} |I_n(f) - \tilde{I}_n(f)| &= \left| \sum_i w_i y_i - \sum_i w_i \tilde{y}_i \right| \\ &= \left| \sum_i w_i (y_i - \tilde{y}_i) \right| \\ \text{dis. triangolare} \rightarrow &\leq \sum_i |w_i| \underbrace{|y_i - \tilde{y}_i|}_{\leq \varepsilon} \\ &\leq \sum_i |w_i| \cdot \varepsilon \\ &= \varepsilon S_n, \quad S_n = \sum_{i=0}^{n} |w_i| \end{aligned}$$

e il ruolo della costante di Lebesgue nelle formule di quadratura è giocato da: $S_n = \sum_{i=0}^{n} |w_i|$
modulante la risposta agli errori.
Le formule di quadratura sono *stabili* se S_n è *limitata*, cioè:

$$\exists k > 0 : S_n \leq k \quad \forall n$$

avendo che S_n non è la somma parziale di una serie, dato che i nodi cambiano.
Le formule di quadratura algebriche e composte con *pesi positivi* sono *stabili*, infatti:

$$S_n = \sum_{i=0}^{n} |w_i| = \sum_{i=0}^{n} w_i = \sum_{i=0}^{n} 1 \cdot w_i = \int_a^b 1 \cdot dx = b - a$$

cioè $S_n = b - a$, dunque è limitata e anche costante.
Perché $\sum_i w_i = \sum_i 1 \cdot w_i = \int_a^b 1 \cdot dx$?
La prima uguaglianza dice che è come se stessimo applicando la formula alla funzione costante $f = 1$; la seconda viene dal fatto che sia le formule algebriche che le formule composte sono "esatte", cioè fanno errore zero, sulle funzioni costanti (che sono polinomi di grado 0).
Infatti l'interpolazione con un unico polinomio $\prod_n$ fa errore zero se $f \in \mathbb{P}$ e $deg(f) \leq n$ ($deg(f)$ significa $grado(f)$), mentre l'interpolazione a tratti fa errore zero se $f \in \mathbb{P}$ e $deg(f) \leq s$ (in entrambi i casi per l'unicità della funzione interpolante).
Purtroppo non tutte le formule di quadratura hanno pesi > 0.

In particolare, i pesi delle formule di *Newton-Cotes* sono positivi per n <= 7, mentre per n > 7 cominciano ad apparire pesi negativi ed il modulo dei pesi cresce tale che S_n diverga esponenzialmente (rendendo le formule molto instabili). Notiamo però che le formule composte di grado s <= 7 a pesi positivi sono stabili (in particolare quelle per s=1 (trapezi), (parabole), s=3 (cubiche)).

Riassumendo:

$$\begin{aligned} |I(f) - \tilde{I}_n(f)| &= |I(f) - I_n(f) + I_n(f) - \tilde{I}_n(f)| \\ &\leq \underbrace{|I(f) - I_n(f)|}_{\text{CONVERGENZA?}} + \underbrace{|I_n(f) - \tilde{I}_n(f)|}_{\text{STABILITÀ?}} \end{aligned}$$

Ad esempio per le formule di "tipo Chebyshev" con $f \in C^k[a,b]$, $k > 0$

$$|I(f) - \tilde{I}_n^{cheb}(f)| \leq \underbrace{(b-a)C^k \frac{log(n)}{n^k}}_{\rightarrow 0,\ n \rightarrow \infty} + \underbrace{(b-a)\varepsilon}_{\text{STABILI}}$$ s=2

mentre per le formule composte con $s \leq 7$ (trapezi, parabole, cubiche, ...)

$$|I(f) - \tilde{I}_n(f)| \leq \underbrace{(b-a)k_s h^{s+1}}_{\rightarrow 0,\ h \rightarrow 0} + \underbrace{(b-a)\varepsilon}_{\text{STABILI}}$$

Per le *formule algebriche*, con i nodi di tipo Chebyshev hanno *pesi positivi e sono stabili e convergenti*.
La costruzione di quadratura si può estendere ad integrali generalizzati del tipo:

$$I_w(f) = \int_a^b f(x) \cdot w(x)dx$$

con $f \in C[a,b]$ e w "funzione peso" positiva, continua e integrabile in (a,b) (ma non necessariamente limitata, ad esempio $w(x) = (1-x^2)^{-1/2},\ [a,b] = [1,1]$).

ad esempio le citate formule gaussiane, formule algebriche a pesi positivi, costruibili per qualsiasi funzione peso "w" e convergenti per ogni f in C{a,b]. In conclusione, la teoria di convergenza è molto più fine di quella tracciata con la stima fondamentale e le formule a pesi positivi non sono solo stabili ma anche convergenti all'integrale di qualsiasi funzione continua.

Lezione 17 - Derivazione numerica: instabilità dell'operatore derivata, formule di derivazione approssimata, instabilità e minimizzazione dell'errore e cenni all'estrapolazione (inizio lezione 18)

Parliamo della ***derivazione numerica***, intesa come operatore funzionale:

$$D : C^1[a,b] \to C[a,b], \quad f \longmapsto Df = f'$$

mandando una funzione "f" derivabile con derivata continua in [a,b] nella sua derivata f'. Sappiamo inoltre si tratta di operazione lineare, quindi:

$$(\alpha f + \beta g)' = \alpha f' + \beta g'$$

La derivazione risulta essere instabile; notiamo infatti che per quanto riguarda l'operazione di integrazione:

$$f \longmapsto I(f) = \int_a^b f(x)dx$$

che è stabile perché

$$|I(f) - I(\tilde{f})| \leq (b-a)dist(f,\tilde{f})$$

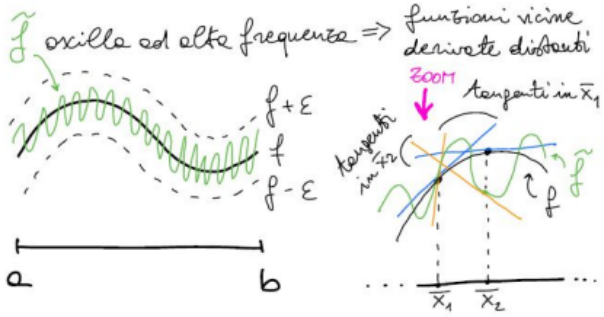
e quindi

$$dist(f,\tilde{f}) \to 0 \Rightarrow |I(f) - I(\tilde{f})| \to 0$$

nel caso della derivazione

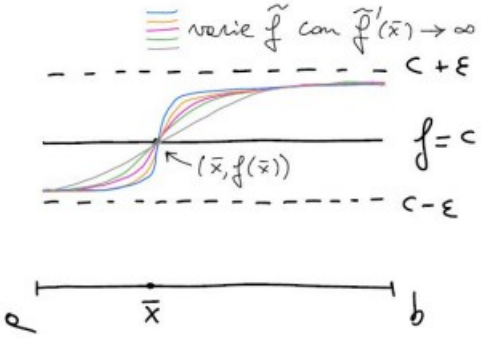
$$dist(f,\tilde{f}) \to 0 \not\Rightarrow dist(f',\tilde{f}') \to 0$$

come si può vedere dal seguente disegno (preso dalla lezione 16)



in cui funzioni arbitrariamente vicine ***possono*** avere derivate arbitrariamente distanti, dato che come si vede la distanza tra le funzioni è oscillante.

Altro disegno utile a vedere l'instabilità è quello a fianco in cui, in intorno di funzione costante, compaiono varie funzioni con derivate via via crescenti. Dunque le derivate hanno contemporaneamente distanza tendente a 0 oppure ad infinito.



un

Ci aspettiamo quindi che anche gli algoritmi ereditino questa instabilità, partendo proprio dalla stessa derivata. Cominciamo quindi a considerare il problema del calcolo di f' in un singolo punto tramite valori campionati in un intorno:

$$I_r = I_r(x) = [x - r, x + r]$$

assumendo $f \in C^2(I_r)$ e usando il rapporto incrementale destro

$$\delta_+(h) = \frac{f(x+h) - f(x)}{h}, \quad 0 < h \leq r$$

Dalla definizione stessa di derivabilità in x abbiamo che

$$\lim_{h \to 0} \delta_+(h) = f'(x)$$

cioè l'algoritmo di calcolo approssimato della derivata corrispondente al calcolo del rapporto incrementale destro è convergente (per questo basterebbe la derivabilità di f in x).
Possiamo però ottenere una stima dell'errore utilizzando la formula di Taylor centrata in x con incremento (passo) h

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2} f''(z)$$

dove $z \in (x, x+h)$. Allora

$$f(x+h) - f(x) = hf'(x) + \frac{h^2}{2} f''(z)$$

cioè

$$\delta_+(h) = \frac{f(x+h) - f(x)}{h} = f'(x) + O(h)$$

nel senso che $\exists c > 0$ tale che

$$|\delta_+(h) - f'(x)| \leq ch, \qquad c = \frac{1}{2} \underset{t \in I_r}{max} |f''(t)| \geq \frac{|f''(z)|}{2}$$

Ricordiamo inoltre i *simboli asintotici* di variabile "u" continua o discreta:

- simbolo "O-grande": $\alpha(u) = O(\beta(u))$
  per $u \to \overline{u}$ significa $\exists c > 0$ (indipendente da u) tale che $|\alpha(u)| \leq c|\beta(u)|$ $\forall u$ in un intorno di $\overline{u}$
- simbolo "o-piccolo": $\alpha(u) = o(\beta(u))$
  per $u \to \overline{u}$ significa $\alpha(u)/\beta(u) \to 0, \ u \to \overline{u}$
- simbolo "$\sim$": $\alpha(u) \sim \beta(u)$
  per $u \to \overline{u}$ significa $\alpha(u)/\beta(u) \to 1, \ u \to \overline{u}$

La convergenza del rapporto incrementale è *lenta*, dato che l'errore è infinitesimo di ordine 1 in *h* e, a causa della presenza di errori sia nelle misure che nel calcolo, avremo solo dei valori approssimati di ciò che vogliamo stimare per l'errore:

$$|f(t) - \tilde{f}(t)| \leq \varepsilon \ \forall t \in I_r$$

Chiamiamo allora $\tilde{\delta}_+(h)$ il rapporto incrementale "perturbato"

$$\tilde{\delta}_+(h) = \frac{\tilde{f}(x+h) - \tilde{f}(x)}{h}$$

che è l'unica quantità che siamo effettivamente in grado di calcolare.
Possiamo scrivere

$$\begin{aligned} |f'(x) - \tilde{\delta}_+(h)| &= |f'(x) - \delta_+(h) + \delta_+(h) - \tilde{\delta}_+(h)| \\ &\leq \underbrace{|f'(x) - \delta_+(h)|}_{\text{convergenza}} + \underbrace{|\delta_+(h) - \tilde{\delta}_+(h)|}_{\text{stabilità}} \quad \longleftarrow \text{diseg. triangolare} \end{aligned}$$

in cui come al solito separiamo lo studio della convergenza dall'analisi di stabilità.

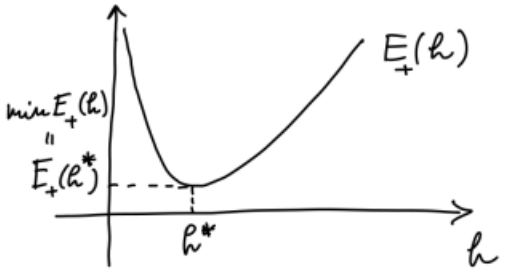
Qui sotto poi lo studio dell'analisi di stabilità:

$$\begin{aligned}
|\delta_+(h) - \tilde{\delta}_+(h)| &= \left| \frac{f(x+h) - f(x)}{h} - \frac{\tilde{f}(x+h) - \tilde{f}(x)}{h} \right| \\
&= \left| \frac{f(x+h) - \tilde{f}(x+h)}{h} + \frac{\tilde{f}(x) - f(x)}{h} \right| \\
&\leq \frac{1}{h} \left| f(x+h) - \tilde{f}(x+h) \right| + \frac{1}{h} \left| \tilde{f}(x) - f(x) \right| \quad \longleftarrow \text{diseg. triangolare} \\
&\leq \frac{1}{h}\varepsilon + \frac{1}{h}\varepsilon = \frac{2\varepsilon}{h}
\end{aligned}$$

da cui

$$\left| f'(x) - \tilde{\delta}_+(h) \right| \leq ch + \frac{2\varepsilon}{h} = E_+(h)$$

Per ε (errore massimo di calcolo/campionamento di f) ci sono 2 esigenze contrastanti, in quanto si deve cercare di prendere h piccolo per rendere piccolo l'errore di convergenza ma, d'altro canto, per ε fissato prendere h → 0 implica che il secondo addendo va ad infinito, dunque *l'errore sulla funzione viene amplificato*. Questa quindi l'instabilità che si eredita dalla derivazione.
Vediamo cosa succede da questo grafico dell'errore $E_+(h)$:



notando che la funzione E_+ è convessa e non si può prendere h grande (per far dominare il termine *ch*) ma neanche piccolo (perché dominerebbe (2ε)/h tendente ad infinito per h → 0).
Si può quindi cercare di prendere il passo h=h* come punto di minimo.
Vediamo poi il calcolo del passo ottimale e dell'errore minimo, andando a cercare i punti stazionari di $E_+(h)$:
Cerchiamo i punti stazionari di $E_+(h)$ (dove $E'_+(h) = 0$)

$$E'_+(h) = \left(ch + \frac{2\varepsilon}{h} \right)' = c - \frac{2\varepsilon}{h^2} = 0$$

$$\Downarrow$$

$$h^2 = \frac{2\varepsilon}{c} \Rightarrow h^* = h^*(\varepsilon) = \sqrt{\frac{2\varepsilon}{c}}$$

Dove con il simbolo $'$ si intende la derivata di $\left(ch + \frac{2\varepsilon}{h}\right)'$ in h.
Inoltre $E''_+(h) = \frac{4\varepsilon}{h^3} > 0$ da cui si vede che $E_+(h)$ è convessa quindi h^* è di minimo:

$$E_+(h^*) = ch^* + \frac{2\varepsilon}{h^*} = c \cdot \sqrt{\frac{2\varepsilon}{c}} + \frac{2\varepsilon}{\sqrt{\frac{2\varepsilon}{c}}} = \sqrt{2c}\sqrt{\varepsilon} + \sqrt{2c}\sqrt{\varepsilon} = 2\sqrt{2c}\sqrt{\varepsilon}$$

Abbiamo quindi che $h^* = O(\sqrt{\varepsilon})$ e $E_+(h^*) = O(\sqrt{\varepsilon})$.
Stiamo quindi limitando la perdita, passando da un errore O(ε) ad un errore stimato O(sqrt(ε)), con perdita di precisione notevole ma non illimitata.
Un modo di aumentare l'ordine di infinitesimo è cambiare la formula di approssimazione della derivata.

Assumiamo ora che $f \in C^3(I_r)$ e scriviamo la formula di Taylor "da destra" e "da sinistra" (centrandola sempre in x, con passo $0 < h \leq r$).

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{3!}f'''(\xi)$$
$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{3!}f'''(\eta)$$

dove $\xi \in (x, x+h)$ e $\eta \in (x-h, x)$ da cui si ottiene, sottraendo membro a membro

$$f(x+h) - f(x-h) = 2hf'(x) + O(h^3)$$
e anche
$$\delta(h) = \frac{f(x+h) - f(x-h)}{2h} = f'(x) + O(h^2)$$

(sottraendo si elidono i termini di grado pari in h), con

$$\begin{aligned} |f'(x) - \delta(h)| &= \frac{1}{12} \cdot |f'''(\xi) + f'''(\eta)| \cdot h^2 \\ &\leq \frac{1}{12}\left(|f'''(\xi)| + |f'''(\eta)|\right) \cdot h^2 \\ &\leq d \cdot h^2 \end{aligned}$$

dove $d = \frac{1}{6}\max_{t \in I_r} |f'''(t)|$.

Si ha quindi che l'errore commesso nell'approssimazione della derivata col rapporto incrementale simmetrico è $O(h^2)$; ora però dobbiamo occuparci della risposta dell'algoritmo agli errori su f, assumendo come prima $\left|\tilde{f}(t) - f(t)\right| \leq \varepsilon$ dove $\tilde{f}(t)$ sono i valori di f affetti da errore.

Dobbiamo quindi stimare $|\delta(h) - \tilde{\delta}(h)|$, con

$$\tilde{\delta}(h) = \frac{\tilde{f}(x+h) - \tilde{f}(x-h)}{2h}$$

(rapporto incrementale simmetrico "perturbato"), vista la stima

$$\begin{aligned} |f'(x) - \tilde{\delta}(h)| &= |f'(x) - \delta(h) + \delta(h) - \tilde{\delta}(h)| \\ &\leq \underbrace{|f'(x) - \delta(h)|}_{\text{convergenza}} + \underbrace{|\delta(h) - \tilde{\delta}(h)|}_{\text{stabilità}} \end{aligned}$$

Ora

$$\begin{aligned} |\delta(h) - \tilde{\delta}(h)| &= \frac{1}{2h}|f(x+h) - f(x-h)| - |\tilde{f}(x+h) - \tilde{f}(x-h)| \\ &= \frac{1}{2h}|(f(x+h) - \tilde{f}(x+h)) + (\tilde{f}(x-h) - f(x-h))| \\ &\leq \frac{1}{2h}(|f(x+h) - \tilde{f}(x+h)| + |\tilde{f}(x-h) - f(x-h)|) \\ &\leq \frac{1}{2h}(\varepsilon + \varepsilon) = \frac{2\varepsilon}{2h} = \frac{\varepsilon}{h} \end{aligned}$$

Otteniamo quindi

$$|f'(x) - \tilde{\delta}(h)| \leq dh^2 + \frac{\varepsilon}{h} = E(h)$$

La stima è simile a prima ma ora l'esponente di *h* nella stima teorica di convergenza è 2 invece di 1 e ci aspettiamo che per rendere dh^2 piccolo basti un passo più grande di quello che serve per *ch*.

Infatti fissato $\sigma > 0$, per avere $dh^2 \leq \sigma$ serve $h = O(\sqrt{\sigma})$ mentre $ch \leq \sigma$ richiede $h = O(\sigma)$, che è una grossa differenza per σ piccolo (ad esempio $\sqrt{\sigma} = 10^{-4}$ per $\sigma = 10^{-8}$).
Questo comporta una minore amplificazione attesa dell'errore ε sui valori di f (tenendo però sempre presente che l'instabilità esiste, perché come prima $\varepsilon/h \to \infty$, $h \to 0$ per ε fissato).
Come nel caso precedente, possiamo cercare di minimizzare

$$\begin{aligned} E(h) &= dh^2 + \frac{\varepsilon}{h} \\ E'(h) &= \left(dh^2 + \frac{\varepsilon}{h}\right)' \\ &= 2dh - \frac{\varepsilon}{h^2} = 0 \Rightarrow h^3 = \frac{\varepsilon}{2d} \\ &\Rightarrow h^* = h^*(\varepsilon) = \left(\frac{\varepsilon}{2d}\right)^{\frac{1}{3}} \end{aligned}$$

D'altra parte

$$\begin{aligned} E(h^*) &= d(h^*)^2 + \frac{\varepsilon}{h^*} \\ &= d\left(\frac{\varepsilon}{2d}\right)^{\frac{2}{3}} + \varepsilon\left(\frac{2d}{\varepsilon}\right)^{\frac{1}{3}} \\ &= 2^{-2/3}\cdot d^{1/3}\cdot\varepsilon^{2/3} + (2d)^{1/3}\cdot\varepsilon^{2/3} \\ &= d^{1/3}\cdot(2^{-2/3}+2^{1/3})\cdot\varepsilon^{2/3} \end{aligned}$$

cioè

$$h^* = O(\varepsilon^{1/3}) \quad \text{e} \quad E(h^*) = O(\varepsilon^{2/3})$$

È chiaro dunque il miglioramento rispetto all'errore minimale del rapporto incrementale standard. L'instabilità ha un effetto ridotto confrontato all'errore ε su f.
Possiamo confrontare le due stime di errore in un esempio con $f'(0) = cos(0) = 1\ e\ f(x) = sin(x)$.
In questo caso avremo $|f''(t)| = |sin(t)| <= 1\ ed\ |f'''(t)| = |cos(t)| <= 1$ e possiamo prendere $c = 1/2\ e\ d = 1/6$, avendo:

1. con $\delta_+(h)$ (detto h_+^* il peso ottimale)

$$h_1^* = 2\sqrt{\varepsilon},\ E_+(h_1^*) = 2\sqrt{\varepsilon}$$

2. con $\delta(h)$

$$h_2^* = 3^{1/3}\cdot\varepsilon^{1/3},\ E(h_2^*) = \ldots \approx 1.04\cdot\varepsilon^{2/3}$$

per $\varepsilon = 10^{-6}$ ad esempio si ottiene

a) $h_1^* = 2\cdot 10^{-3},\ E_+(h_1^*) = 2\cdot 10^{-3}$

b) $h_2^* \approx 1.44\cdot 10^{-2},\ E(h_2^*) \approx 1.04\cdot 10^{-4}$

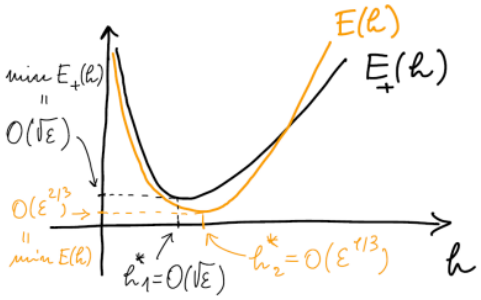
Astraendo rispetto ai numeri appena calcolati, si ha che

$$h_1^* = O(\sqrt{\varepsilon}) < h_2^* = O(\varepsilon^{1/3})$$

mentre

$$E(h_2^*) = O(\varepsilon^{2/3}) < E_+(h_1^*) = O(\sqrt{\varepsilon})$$

almeno per ε abbastanza piccolo, cioè graficamente



Nel calcolo approssimato della derivata, dunque, non conviene campionare con passo troppo piccolo in quanto l'errore di approssimazione potrebbe ampliare in modo inaccettabile l'errore di misura/calcolo.
Per evitare di usare passi troppo piccoli, convengono formule di approssimazione della derivata del tipo:

$$\phi(h) = f'(x) + O(h^p)$$

con un "p" grande. Citiamo la lezione dopo in merito *all'estrapolazione*, cioè il processo di calcolo di informazioni esterne rispetto ad un insieme discreto di dati noti. Dato un piano cartesiano, a ciascun valore di y_w si vuole trovare un valore di x_w maggiore di x_i
Usiamo una delle formule usate nel calcolo approssimato della derivata tramite derivazione numerica:

Chiamando $\phi(h)$ una di queste formule e $\tilde{\phi}(h)$ la formula in cui si usano i dati realmente misurati, cioè valori $\tilde{f}(t)$ con $|\tilde{f}(t) - f(t)| \leq \varepsilon$, abbiamo visto che

$$\left|\tilde{\phi}(h) - f'(x)\right| = O(h^p) + O\left(\frac{\varepsilon}{h}\right)$$

(con $p = 1$ per $\phi(h) = \delta_+(h)$ rapporto incrementale standard e $p = 2$ per $\phi(h) = \delta(h)$ rapporto incrementale simmetrico), dove il termine $O(\varepsilon/h)$ mostra l'instabilità della formula per ε fissato e $h \to 0$.

Scritto da Gabriel

Citiamo la funzione ϕ(h) con il calcolo visto sopra, che ci permette di enunciare *la stima a posteriori* dell'errore e *l'aumento a costo computazionale basso dell'ordine di infinitesimo "p" dell'errore*.

Cominciamo con $\phi(h) = \delta_+(h)$ assumendo che $f \in C^3(I_r)$ dove $I_r = I_r(x) = [x - r, x + r]$ è un intorno chiuso di x. Usando la formula di Taylor di grado 2 in h con resto in forma di Lagrange

$$f(x+h) = f(x) + hf'(x) + h^2\frac{f''(x)}{2} + h^3\frac{f'''(z)}{6}$$

con $z \in (x, x+h)$, da cui

$$\delta_+(h) = \frac{f(x+h) - f(x)}{h} = f'(x) + h\frac{f''(x)}{2} + h^2\frac{f'''(z)}{6} = f'(x) + h\frac{f''(x)}{2} + O(h^2)$$

Si vede quindi che chiedendo maggiore regolarità ad f, l'errore $\delta_+(h) - f'(x)$ ha una struttura asintotica del tipo

$$\delta_+(h) - f'(x) = ch + O(h^2)$$
$$c = \frac{f''(x)}{2}$$

Ovviamente $\delta_+(h) - f'(x) = O(h)$ ma siamo riusciti a dare una struttura più fine al termine $O(h)$ come somma di un termine proporzionale ad h + un infinitesimo di ordine 2 in h.

In merito alla *stima a posteriori dell'errore* cerchiamo di stimare la parte principale dell'errore, quindi il termine *ch*, sapendo che non conosciamo f''(x), quindi neanche il termine *c* e non possiamo avere informazioni quantitative su derivate successive; tuttavia affermiamo che per f ∈ C^3 esiste la struttura asintotica indicata appena sopra.

L'idea è semplice: calcolando la formula con un passo più piccolo (ad esempio $\frac{h}{2}$ che è la scelta usuale) abbiamo

$$\delta_+\left(\frac{h}{2}\right) = f'(x) + \frac{f''(x)}{2}\frac{h}{2} + O(h^2)$$

visto che se $u(h) = O(h^2)$ allora $u(\frac{h}{2})$ è ancora $O(h^2)$ (qui $u(h) = \delta_+(\frac{h}{2}) - f'(x) - \frac{f''(x)}{2}h$).
Infatti $|u(h) \le \gamma h^2|$ con γ indipendente da h, quindi $|u(\frac{h}{2})| \le \gamma(\frac{h}{2})^2 = \frac{\gamma}{4}h^2 = O(h^2)$.
Ma allora sottraendo membro a membro

$$\begin{aligned}\delta_+(h) - \delta_+\left(\frac{h}{2}\right) &= \cancel{f'(x)} + \frac{f''(x)}{2}h + O(h^2) - \left(\cancel{f'(x)} + \frac{f''(x)}{2}\frac{h}{2} + O(h^2)\right)\\ &= \frac{f''(x)}{2}\left(h - \frac{h}{2}\right) + O(h^2) - O(h^2)\\ &= \frac{f''(x)}{2}\frac{h}{2} + O(h^2)\end{aligned}$$

perché somma o differenza di due termini che sono $O(h^2)$ resta un $O(h^2)$.
Infatti se $u(h) = O(h^2)$ e $v(h) = O(h^2)$, cioè $\exists \gamma_1, \gamma_2 > 0$ tali che $|u(h)| \le \gamma_1 h^2$ e $|v(h)| \le \gamma_2 h^2$, allora

$$\begin{aligned}|u(h) \pm v(h)| &\underset{\text{diseg. triangolare}}{\le} |u(h)| + |v(h)|\\ &\le \gamma_1 h^2 + \gamma_2 h^2 = (\gamma_1 + \gamma_2)h^2\end{aligned}$$

Abbiamo ottenuto

$$\delta_+(h) - \delta_+\left(\frac{h}{2}\right) = \frac{f''(x)}{2}\frac{h}{2} + O(h^2)$$

e sappiamo che

$$\delta_+\left(\frac{h}{2}\right) - f'(x) = \frac{f''(x)}{2}\frac{h}{2} + O(h^2)$$

Trascurando i termini $O(h^2)$ (visto che per h piccolo il termine lineare in h sarà dominante nell'errore)

$$\left|\delta_+(h) - \delta_+\left(\frac{h}{2}\right)\right| \approx \frac{|f''(x)|}{2}\frac{h}{2} \approx \left|\delta_+\left(\frac{h}{2}\right) - f'(x)\right|$$

cioè abbiamo ricavato una stima a posteriori (con le quantità calcolate) dell'errore commesso approssimando $f'(x)$ con $\delta_+(\frac{h}{2})$ (che è tendenzialmente più accurato di $\delta_+(h)$)

$$\left|f'(x) - \delta_+\left(\frac{h}{2}\right)\right| \underset{\text{stima a posteriori}}{\approx} \left|\delta_+(h) - \delta_+\left(\frac{h}{2}\right)\right|$$

Per quanto riguarda invece *l'aumento dell'ordine di infinitesimo dell'errore*:

Ripartiamo dalla struttura asintotica:

$$\delta_+(h) = f'(x) + \frac{f''(x)}{2}h + O(h^2)$$
$$\delta_+\left(\frac{h}{2}\right) = f'(x) + \frac{f''(x)}{2}\frac{h}{2} + O(h^2)$$

Adesso sfruttiamo di nuovo la struttura asintotica, stavolta per eliminare la parte principale dell'errore e arrivare ad una formula con errore di ordine superiore come infinitesimo in h

$$\delta_+(h) = f'(x) + \frac{f''(x)}{2}h + O(h^2)$$
$$2\delta_+\left(\frac{h}{2}\right) = 2f'(x) + \frac{f''(x)}{2}2\frac{h}{2} + O(h^2)$$

quindi

$$\begin{aligned} 2\delta_+\left(\frac{h}{2}\right) - \delta_+(h) &= 2f'(x) + \cancel{\frac{f''(x)}{2}h} + O(h^2) - \left(f'(x) + \cancel{\frac{f''(x)}{2}h} + O(h^2)\right) \\ &= 2f'(x) - f'(x) + O(h^2) - O(h^2) \\ &= f'(x) + O(h^2) \end{aligned}$$

(si noti che abbiamo usato il fatto che $2O(h^2) = O(h^2) : |u(h)| \leq \gamma h^2 \Rightarrow |k \cdot u(h)| \leq k \cdot \gamma h^2$ se k è una costante).
In definitiva:

$$\phi_1(h) = 2\delta_+\left(\frac{h}{2}\right) - \delta_+(h) = f'(x) + O(h^2)$$

cioè con una semplice speciale combinazione lineare delle formule con passo h e $\frac{h}{2}$ abbiamo ricavato una nuova formula con errore $O(h^2)$ invece di $O(h)$.

Grazie a questa struttura asintotica si ha la tecnica della *estrapolazione*, poco generalizzabile a tutte le formule in cui l'errore è dotato di struttura asintotica.

Come primo esempio, pensiamo di calcolare la derivata di $f(x) = e^x$ in $x = 0$, $f'(0) = e^0 = 1$ utilizzando $\delta_+(h)$, $\delta_+\left(\frac{h}{2}\right)$ e $\phi_1(h)$ con $h = \frac{1}{10}$

$$\delta_+(h) = \frac{e^h - e^0}{h} = \frac{e^{1/10} - 1}{1/10} = 1.0517\ldots7$$
$$\delta_+\left(\frac{h}{2}\right) = \frac{e^{1/20} - 1}{1/20} = 1.0254\ldots2$$
$$\phi_1(h) = 2 \cdot \delta_+(h/2) - \delta_+(h) = 0.99913\ldots5$$

Guardando gli errori (arrotondati alla seconda cifra)

$$|\delta_+(h) - f'(0)| \approx 0.5 \cdot 10^{-1}$$
$$|\delta_+(h/2) - f'(0)| \approx 0.25 \cdot 10^{-1}$$
$$|\phi_1(h) - f'(0)| \approx 0.87 \cdot 10^{-3}$$

Si noti il miglioramento ottenuto con $\phi_1(h)$, che ha un errore paragonabile a quello ottenibile con $\delta(h) = f'(0) + O(h^2)$

$$|\delta(h) - f'(0)| = \left|\frac{e^h - e^{-h}}{2h} - 1\right| \approx 1.7 \cdot 10^{-3}$$

Si noti anche che la stima a posteriori dell'errore commesso con $\delta_+(h/2)$

$$\left|\delta_+(h) - \delta_+\left(\frac{h}{2}\right)\right| \approx 0.26 \cdot 10^{-1}$$

risulti essere molto accurata.
Nota: a quanto pare l'estrapolazione non la chiede allo scritto (ma fa parte di cultura generale)

Lezione 19 – Norme vettoriali e matriciali e Lezione 20 – Condizionamento di matrici e sistemi lineari

I metodi dell'algebra lineare numerica sono pervasivi e trovano moltissime applicazioni, per esempio negli approcci di linearizzazione, in particolare per sistemi di vettori e matrici. Ci occuperemo solamente della soluzioni di sistemi lineari, in particolare per il *metodo di eliminazione di Gauss* ed il *metodo dei minimi quadrati*, agendo sulle successioni di vettori.

Citiamo il concetto di norma vettoriale:

Una NORMA vettoriale in $\mathbb{R}^n$ è una funzione

$$\|\cdot\| : \mathbb{R}^n \to [0, +\infty), \quad x \mapsto \|x\|$$

con le seguenti proprietà:

1. $\|x+y\| \leq \|x\| + \|y\|, \quad \forall x, y \in \mathbb{R}^n$ [per la disuguaglianza triangolare]
2. $\|\alpha x\| = |\alpha| \|x\|, \quad \forall \alpha \in \mathbb{R}, \quad \forall x \in \mathbb{R}^n$
3. $\|x\| = 0 \iff x = 0 = (0, \dots, 0)$ [vettore nullo]

Si dimostra facilmente che vale

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

Qualche esempio di norma:

- norma 1 $\|x\|_1 = \sum_{i=1}^{n} |x_i|$
- norma 2, euclidea $\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2} = \sqrt{(x,x)}$
- norma infinito, norma del massimo modulo: $\|x\|_\infty = \max_{1 \leq i \leq n} |x_i|$

Citiamo questi per capire, nel caso di una classica norma ||.||, la distanza e gli errori tra un vettore vicino ad un altro, capendo l'errore su ciascuno di questi:

$$dist_{\|\cdot\|}(x,y) = \|x-y\|$$

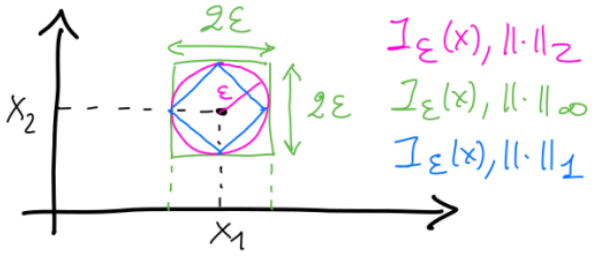
Possiamo quindi definire un intorno chiuso e poi con successiva forma geometrica:

$$I_\varepsilon(x) = \{y \in \mathbb{R}^n : \|x-y\| \leq \varepsilon\}$$

cioè l'insieme degli infiniti vettori che distano non più di ε da x (detto centro dell'intorno).
Un intorno aperto è definito tramite la diseguaglianza stretta

$$\dot{I}_\varepsilon(x) = \{y \in \mathbb{R}^n : \|x-y\| < \varepsilon\}$$

La forma geometrica degli intorni dipende dalla norma, come si vede in questo disegno per $n = 2$



in particolare dicendo che questo intorno per ogni dimensione forma un cerchio (caso norma 2), una ipersfera (quadrato di lato 2ε), ecc.

Una proprietà importante è che nei reali R^n le norme sono "*equivalenti*" e date:

$\|\cdot\|_A$ e $\|\cdot\|_B$ $\exists c_1, c_2 > 0$ tali che $c_2\|x\|_B \leq \|x\|_A \leq c_1\|x\|_B$.
Ad esempio, siccome

$$\|x\|_2^2 = \sum_{i=1}^{n} x_i^2 \leq \sum_{i=1}^{n} (max|x_i|)^2 = \|x\|_\infty^2 \sum_{i=1}^{n} 1 = n\|x\|_\infty^2$$

e

$$\|x\|_\infty^2 = \max_i |x_i|^2 \leq \sum_{i=1}^{n} x_i^2$$

si ha che

$$\underset{=c_2}{\frac{1}{}} \cdot \|x\|_\infty \leq \|x\|_2 \leq \underset{=c_1}{\sqrt{n}} \|x\|_\infty$$

Avendo delle distanze possiamo dire quando una successione $\{x^{(k)}\}$ di vettori di $\mathbb{R}^n$ converge a $x \in \mathbb{R}^n$, cioè $dist_{\|\cdot\|}(x, x^{(k)}) = \|x - x^{(k)}\| \to 0, \ k \to \infty$.

La nozione di convergenza tra coppie di norme, quindi, non dipende dalla norma.
Vediamo la *norma matriciale*:

$$A \in \mathbb{R}^{n\times n} = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ a_{21} & \dots & a_{2n} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{pmatrix}$$

con questa che rappresenta operatore lineare di trasformazione dei vettori, con trasformazione anch'essa lineare usando la successiva regola:

$$Ax = A\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \left(\sum_{j=1}^{n} a_{ij}x_j\right)_{1\le i\le n}$$

La trasformazione è lineare perchè $\forall \alpha, \beta \in \mathbb{R}$ e $x, y \in \mathbb{R}^n$

$$A(\alpha x + \beta y) = \alpha Ax + \beta Ay$$

Fissata una norma vettoriale possiamo definire:

$$\|A\| = \sup_{x\neq 0} \frac{\|Ax\|}{\|x\|}$$

Questo tipo di norma matriciale è definito *norma indotta* della norma vettoriale ||.||
Utile enunciare 3 proprietà caratteristiche delle norme:

- la disuguaglianza triangolare viene da

  $$\|(A_1 + A_2)x\| = \|A_1x + A_2x\| \le \|A_1x\| + \|A_2x\|$$

  quindi

  $$\begin{aligned} \sup_{x\neq 0} \frac{\|(A_1 + A_2)x\|}{\|x\|} &\le \sup_{x\neq 0} \frac{\|A_1x\| + \|A_2x\|}{\|x\|} \\ &\le \sup_{x\neq 0} \frac{\|A_1x\|}{\|x\|} + \sup_{x\neq 0} \frac{\|A_2x\|}{\|x\|} \\ &= \|A_1\| + \|A_2\| \end{aligned}$$

- per $\alpha \in \mathbb{R}$

  $$\|\alpha A\| = \sup_{x\neq 0} \frac{\|\alpha Ax\|}{\|x\|} = \sup_{x\neq 0} \frac{|\alpha|\|Ax\|}{\|x\|} = |x|\|A\|$$

- se $\|A\| = 0$ allora

  $$\sup_{x\neq 0} \frac{\|Ax\|}{\|x\|} = 0$$

  quindi $\|Ax\| = 0 \quad \forall x \Rightarrow A = 0$ [matrice nulla]

Dunque le *disuguaglianze fondamentali per le norme matriciali indotte*:

i. $\|Ax\| \le \|A\| \cdot \|x\| \quad \forall A \in \mathbb{R}^{n\times n} \quad \forall x \in \mathbb{R}^n$

ii. $\|AB\| \le \|A\| \cdot \|B\| \quad \forall A, B \in \mathbb{R}^{n\times n}$

Per la dimostrazione:

1) Per quando riguarda (i) basta ricordare la definizione di norma indotta, se $\sup_{x\neq 0} \frac{\|Ax\|}{\|x\|} = \|A\|$ allora

   $$\forall x \quad \frac{\|Ax\|}{\|x\|} \le sup = \|A\|$$

2) Invece (ii) viene dal significato del prodotto AB come composizione (trasformazione lineare COMPOSTA) delle due trasf. lineari A e B

$$ABx = A(Bx)$$

Quindi

$$\|ABx\| = \|A(Bx)\| \underbrace{\leq \|A\| \cdot \|Bx\| \leq \|A\| \cdot \|B\| \cdot \|x\|}_{\text{per la (1)}}$$

da cui $\forall x \neq 0$

$$\frac{\|ABx\|}{\|x\|} \leq \|A\|\|B\|$$

e perciò

$$\|AB\| = \sup_{x \neq 0} \frac{\|ABx\|}{\|x\|} \leq \|A\|\|B\|$$

La seconda permette di dire che ogni norma matriciale rispetta il prodotto (dato che esso non gode di proprietà commutativa). Ad esempio:

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

(si verifica facilmente che ha le 3 proprietà caratteristiche di una norma).
Possiamo fare un semplice esempio con $n = 2$

$$A = B = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad AB = A^2 = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}$$

Abbiamo $\|A\| = 1$ e $\|A^2\| = 2 > \|A\|^2 = 1$.
Quindi

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

non rispetta la moltiplicazione (e di conseguenza non può essere indotta da alcuna norma vettoriale).

Vediamo ora le *norme indotte notevoli*:

- La prima, che viene chiamata $\|A\|_\infty$ è

$$\|A\|_\infty = \sup_{x \neq 0} \frac{\|Ax\|_\infty}{\|x\|_\infty} = \max_{1 \leq i \leq n} \sum_{j=1}^{n} |a_{ij}|$$

cioè è il massimo al variare delle righe di A delle somme dei moduli degli elementi della riga.
Non è difficile verificare che vale

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

infatti:

$$\begin{aligned} \|Ax\|_\infty &= \max_{1 \leq i \leq n} |(Ax)_i| \quad \leftarrow \text{componente } i\text{-esima di } Ax \\ &= \max_{1 \leq i \leq n} \left| \sum_{j=1}^{n} a_{ij} x_j \right| \\ &\leq \max_{1 \leq i \leq n} \sum_{j=1}^{n} |a_{ij}||x_j| \\ &\leq \max_{1 \leq i \leq n} \|x\|_\infty \sum_{j=1}^{n} |a_{ij}| \\ &= \|x\|_\infty \max_{1 \leq i \leq n} \sum_{j=1}^{n} |a_{ij}| \end{aligned}$$

e quindi $\forall\, x \neq 0$

$$\frac{\|Ax\|_\infty}{\|x\|_\infty} \leq \max_{1 \leq i \leq n} \sum_{j=1}^{n} |a_{ij}|$$

Per $||.||_2$ in **R** si ottiene quella chiamata $||A||_2$ e vale:

$$\|A\|_2 = \sup_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \sqrt{\max_{1 \leq i \leq n} |\lambda_i(A^t A)|}$$

Scritto da Gabriel

dove con $\lambda_i(B)$ indichiamo gli n autovalori (contati con la loro molteplicità) di una matrice $B \in \mathbb{R}^{n\times n}$, che in generale sono numeri complessi, essendo gli zeri di un polinomio di grado n, il "polinomio caratteristico"

$$\mathcal{X}_B(\lambda) = det(\lambda I - B)$$

con

$$I = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix} \quad \text{matrice identità}$$

Quindi $\|A\|_2$ è il massimo dei moduli degli autovalori di

$$B = A^t A$$

Enunciamo due risultati importanti:

5.1.3 TEOREMA (sulla localizzazione degli autovalori di una matrice)

Sia $A \in \mathbb{R}^{n\times n}$ e $\|A\|$ una qualsiasi norma matriciale indotta.
Allora gli autovalori di A stanno nel cerchio chiuso del piano complesso di centro l'origine e raggio $\|A\|$. In formule:

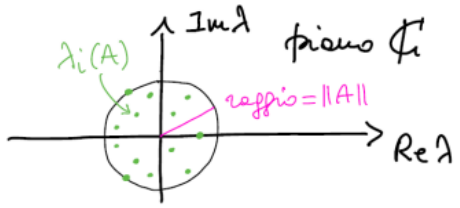
$$det(\lambda I - A) = 0 \Rightarrow |\lambda| \leq \|A\|$$

Ricordando che un autovalore si riferisce alle matrici quadrate e:

Si dice che lo scalare $\lambda_0 \in \mathbb{K}$ è un **autovalore** della matrice quadrata A se esiste un vettore colonna non nullo $\mathbf{v} \in \mathbb{K}^n$ tale che

$$A\mathbf{v} = \lambda_0 \mathbf{v}$$

Graficamente rappresentato come:



Per la dimostrazione:

Se $\lambda \in \mathbb{C}$ è autovalore di A (ricordiamo ancora che gli autovalori di una matrice sono in generale complessi; se la matrice è reale vanno a coppie di complessi coniugati perché $det(\lambda I - A)$ ha coefficienti reali), per definizione $\exists\, x \neq 0$ autovettore tale che

$$Ax = \lambda x \quad (\text{cioè } (\lambda I - A)x = 0)$$

Usando la norma vettoriale che induce $\|A\|$

$$\|\lambda x\| \overset{(*)}{=} |\lambda|\|x\| = \|Ax\| \leq \|A\|\|x\|$$

[$(*)$ 2° proprietà di una norma, vale anche per $\lambda \in \mathbb{C}$]
e dividendo per $\|x\|$

$$|\lambda| \leq \|A\|$$

Un teorema utile (*ma non richiesto a quanto sembra, utile sapere l'enunciato cit. prof*):

5.1.4 TEOREMA (sull'invertibilità di $I - A$ per $\|A\| < 1$)

Sia $A \in \mathbb{R}^{n_x n}$ tale che $\|A\| < 1$, dove $\|A\|$ è una qualsiasi norma matriciale indotta.
Allora $I - A$ è invertibile e vale la stima:

$$\|(I - A)^{-1}\| \leq \frac{1}{1 - \|A\|}$$

Ora per la lezione 20 usiamo le due disuguaglianze fondamentali per le norme matriciali indotte:

(i) $\|Ax\| \leq \|A\| \cdot \|x\|$ ($1°$ diseguaglianza fondamentale)

(ii) $\|AB\| \leq \|A\| \cdot \|B\|$ ($2°$ diseguaglianza fondamentale)

vedendo come viene stimata la risposta agli errori di un sistema lineare non singolare:

$Ax = b,\ A \in \mathbb{R}^{n \times n},\ b, x \in \mathbb{R}^n,\ det(A) \neq 0$

In merito alle *perturbazioni in un sistema lineare*, sapendo che il vettore termine noto e la matrice stessa possono essere affetti da errori, risolviamo un sistema "perturbato", indicato con la tilda:

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

dove $\tilde{b} = b + \delta b,\ \tilde{A} = A + \delta A$ cioè

$$\tilde{b} = \begin{pmatrix} \tilde{b}_1 \\ \vdots \\ \tilde{b}_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix} + \begin{pmatrix} \delta b_1 \\ \vdots \\ \delta b_n \end{pmatrix}$$

$$\tilde{A} = \begin{pmatrix} \tilde{a}_{11} & \dots & \tilde{a}_{1n} \\ \vdots & & \vdots \\ \tilde{a}_{n1} & \dots & \tilde{a}_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{pmatrix} + \begin{pmatrix} \delta a_{11} & \dots & \delta a_{1n} \\ \vdots & & \vdots \\ \delta a_{n1} & \dots & \delta a_{nn} \end{pmatrix}$$

sono il vettore termine noto "perturbato" e la matrice "perturbata" a cui corrisponde un vettore soluzione "perturbato" $\tilde{x} = x + \delta x$

$$\tilde{x} = \begin{pmatrix} \tilde{x}_1 \\ \vdots \\ \tilde{x}_n \end{pmatrix} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} \delta x_1 \\ \vdots \\ \delta x_n \end{pmatrix}$$

Stimare δx in funzione di δA e δb è detto problema di "condizionamento" del sistema lineare.

5.2.2 PROPOSIZIONE 1 (errore sulla soluzione di una sistema lineare generato da una perturbazione del termine noto)

Sia $A \in \mathbb{R}^{n\times n}$ una matrice non singolare, $x \in \mathbb{R}^n$ soluzione del sistema $Ax = b$, $b \neq 0$ e $\tilde{x} = x + \delta x$ soluzione del sistema perturbato $A\tilde{x} = \tilde{b} = b + \delta b$.
Fissata una norma vettoriale $\|\cdot\|$ in $\mathbb{R}^n$, vale la seguente stima dell'errore "relativo" su x

$$\frac{\|\delta x\|}{\|x\|} \leq k(A)\frac{\|\delta b\|}{\|b\|}$$

dove

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

prodotto della norma indotta di A e A^{-1}, è detto "INDICE (o anche numero) DI CONDIZIONAMENTO" della matrice A (nella norma indotta).

Per la dimostrazione:

Osserviamo che $x = A^{-1}b \neq 0$ quindi ha senso stimare l'errore relativo in norma (cioè l'errore assoluto $\|\delta x\|$ diviso per la "lunghezza" di x, cioè $\|x\|$).
Ora

$$\tilde{x} = x + \delta x = A^{-1}\tilde{b} = A^{-1}(b + \delta b) = A^{-1}b + A^{-1}\delta b$$

da cui otteniamo la stima dell'errore assoluto

$$\|\delta x\| = \|A^{-1}\delta b\| \underset{1^o dis. fond.}{\leq} \|A^{-1}\| \cdot \|\delta b\|$$

Per stimare l'errore relativo dobbiamo stimare da sopra $\frac{1}{\|x\|}$, cioè da sotto $\|x\|$.
Siccome x è la soluzione

$$\|b\| = \|Ax\| \underset{1^o dis. fond.}{\leq} \|A\| \cdot \|x\|$$

da cui

$$\|x\| \geq \frac{\|b\|}{\|A\|}$$

e

$$\frac{1}{\|x\|} \leq \frac{\|A\|}{\|b\|}$$

perciò

$$\frac{\|\delta x\|}{\|x\|} \leq \frac{\|A^{-1}\| \cdot \|\delta b\|}{\|x\|} \leq \|A^{-1}\| \cdot \|A\| \cdot \frac{\|\delta b\|}{\|b\|} = k(A) \cdot \frac{\|\delta b\|}{\|b\|}$$

■

Osserviamo due cose:

Proprietà 1

$k(A) \geq 1$: infatti

$$k(A) = \|A^{-1}\| \cdot \|A\| \underset{1^o dis. fond.}{\geq} \|A^{-1}A\| = \|I\| = 1$$

Quindi $k(A)$ nella stima ha il ruolo di "amplificatore" dell'errore sui dati.

Per $k(A) >> 1$ si dice che il sistema è *mal condizionato*, quindi piccole perturbazioni dei dati possono portare ad errori molto grandi sulla soluzione. Può succedere inoltre che per alcuni valori di k(A) si abbia un sistema "estremamente" mal condizionato a causa dell'arrotondamento delle componenti di *b*:

$$\frac{\|\delta x\|}{\|x\|} > 1$$

(cioè l'errore diventa maggiore del 100%).

Nonostante situazioni estreme, non ha molto senso calcolare la soluzione del sistema con algoritmi classici ma vanno costruiti con metodi che limitano la perdita di precisione, ad esempio la minimizzazione dell'errore. In ogni caso, è comunque importante stimare k(A) in qualche norma per avere idea di quanta precisione poter perdere rispetto alla precisione con cui sono noti i dati, poiché k(A) dipende dalla norma indotta che si usa.

Scritto da Gabriel

Vediamo un esempio:

Consideriamo il sistema 2×2

$$\begin{cases} 7x_1 + 10x_2 = b_1 \\ 5x_1 + 7x_2 = b_2 \end{cases}$$

con matrice (e inversa)

$$A = \begin{pmatrix} 7 & 10 \\ 5 & 7 \end{pmatrix}, \; A^{-1} = \begin{pmatrix} -7 & 10 \\ 5 & -7 \end{pmatrix}$$

Prendiamo $b = \binom{b_1}{b_2} = \binom{1}{0.7}$ a cui corrisponde la soluzione $x = \binom{x_1}{x_2} = A^{-1}b = \binom{0}{0.1}$ e consideriamo il sistema perturbato

$A\tilde{x} = \tilde{b} = \binom{\tilde{b}_1}{\tilde{b}_2} = \binom{1.01}{0.69}$ con soluzione $\tilde{x} = \binom{\tilde{x}_1}{\tilde{x}_2} = A^{-1}\tilde{b} = \binom{-0.17}{0.22}$ (soluzioni calcolabili "a mano" per sostituzione)

Si vede che $\delta b = \tilde{b} - b = \binom{0.01}{-0.01}$ e

$$\frac{\|\delta b\|_\infty}{\|b\|_\infty} = \frac{max\{|\delta b_1|, |\delta b_2|\}}{max\{|b_1|, |b_2|\}} = \frac{0.01}{1} = 10^{-2} = 1\%$$

mentre $\delta x = \tilde{x} - x = \binom{-0.17}{0.12}$ e

$$\frac{\|\delta x\|_\infty}{\|x\|_\infty} = \frac{max\{|\delta x_1|, |\delta x_2|\}}{max\{|x_1|, |x_2|\}} = \frac{0.17}{0.1} = 1.7 = 170\%$$

Quindi un errore relativo (in $\|\cdot\|_\infty$) dell'1% su b ha come effetto un errore relativo del 170% su x, rendendo la soluzione $\tilde{x}$ inaccettabile.
Il motivo è ben spiegato calcolando $K_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty$. Infatti

$$\|A\|_\infty = max\left\{ \overset{\text{somme moduli 1° riga}}{10+7} \; , \; \overset{\text{somme moduli 2° riga}}{5+7} \right\} = 17$$

$$\|A^{-1}\|_\infty = max\{7+10, 5+7\} = 17$$

(il fatto che siano uguali è casuale), quindi

$$K_\infty(A) = 17 \cdot 17 = 289$$

Dove l'indice di condizionamento è piccolo e conta il prodotto:

$$k(A) \cdot \frac{\|\delta b\|}{\|b\|}$$

che qui è > 1 perchè l'errore sui dati è solo dell'1%.
Con

$$\frac{\|\delta b\|_\infty}{\|b\|_\infty}$$

dell'ordine di 10^{-k} ci aspetteremmo un errore

$$\frac{\|\delta x\|_\infty}{\|x\|_\infty}$$

dell'ordine di 10^{2-k} (quindi ad esempio per $k = 6$ avremmo una soluzione $\tilde{x}$ che fa un errore relativo dell'ordine di $10^{-4} = 0.01\%$, che potrebbe essere più che accettabile in molte applicazioni pratiche).
Quello che conta è che se abbiamo una stima degli errori sui dati tipo

$$\frac{\|\delta b\|}{\|b\|} \approx \varepsilon$$

allora crescendo l'ordine di grandezza di $k(A)$ sappiamo che possiamo aspettarci un errore

$$\frac{\|\delta x\|}{\|x\|} \lesssim k(A) \cdot \varepsilon$$

e quindi possiamo decidere se la soluzione sarà o meno accettabile.

Usiamo poi la proprietà 2 della stima da sotto:

Scritto da Gabriel

$$k(A) \geq \frac{\max_i |\lambda_i(A)|}{\min_i |\lambda_i(A)|}$$

osservando che A è invertibile e non ha autovalori nulli. Vale la stima:

$$|\lambda_i(B)| \leq \|B\|$$

Quindi applicando la stima con $B = A$ otteniamo

$$\|A\| \geq \max_i |\lambda_i(A)|$$

D'altra parte, chi sono gli autovalori di A^{-1}? Se λ è autovalore di A, allora $\exists x \neq 0$ (autovettore) tale che $Ax = \lambda x$. Moltiplicando a sx per A^{-1}

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

vedendo che gli A^{-1} sono *reciproci* degli autovalori di A.

Applicando la stima di localizzazione:

$$\|A^{-1}\| \geq \max_i |\lambda_i(A^{-1})| = \max_i \left|\frac{1}{\lambda_i(A)}\right| = \frac{1}{\min_i |\lambda_i(A)|}$$

quindi

$$k(A) = \|A\| \cdot \|A^{-1}\| \geq \frac{\max_i |\lambda_i(A)|}{\min_i |\lambda_i(A)|}$$

Osserviamo anche da questa stima da sotto otteniamo subito $k(A) \geq 1$ perché

$$\max_i |\lambda_i(A)| \geq \min_i |\lambda_i(A)|$$

Osserviamo anche che la disuguaglianza può diventare uguaglianza in certi casi: ad esempio, se A è simmetrica si ha $\|A\|_2 = \max_i |\lambda_i(A)|$ e lo stesso vale per A^{-1} che resta simmetrica

$$\|A^{-1}\|_2 = \max_i |\lambda_i(A^{-1})| = \frac{1}{\min_i |\lambda_i(A)|}$$

quindi

$$k_2(A) = \frac{\max_i |\lambda_i(A)|}{\min_i |\lambda_i(A)|}$$

5.2.3 PROPOSIZIONE 2 (errore sulla soluzione di un sistema lineare generato da una perturbazione della matrice)

Sia $A \in \mathbb{R}^{n \times n}$ una matrice non singolare, $x \in \mathbb{R}^n$ soluzione del sistema $Ax = b$, $b \neq 0$ e $\tilde{x} = x + \delta x$ soluzione del sistema perturbato $\tilde{A}\tilde{x} = b$, $\tilde{A} = A + \delta A$.
Fissata una norma vettoriale $\|\cdot\|$ in $\mathbb{R}^n$, vale la seguente stima dell'errore "relativo" su x

$$\frac{\|\delta_x\|}{\|\tilde{x}\|} \leq k(A) \cdot \frac{\|\delta A\|}{\|A\|}$$

Per la dimostrazione:

Da $\tilde{A}\tilde{x} = (A + \delta A)(x + \delta x) = b$ otteniamo

$$Ax + A\delta x + \delta A\tilde{x} = b$$

cioè

$$\delta x = A^{-1}(-\delta A\tilde{x}) = -A^{-1}\delta A\tilde{x}$$

Quindi

$$\begin{aligned} \|\delta x\| &\leq \|A^{-1}\| \cdot \|\delta A\tilde{x}\| \\ &\leq \|A^{-1}\| \cdot \|\delta A\| \cdot \|\tilde{x}\| \end{aligned}$$

e perciò

$$\begin{aligned} \frac{\|\delta x\|}{\|\tilde{x}\|} &\leq \|A^{-1}\| \cdot \|\delta A\| \\ &= \|A\| \cdot \|A^{-1}\| \cdot \frac{\|\delta A\|}{\|A\|} \\ &= k(A) \cdot \frac{\|\delta A\|}{\|A\|} \end{aligned}$$

Citiamo solo questo teorema (con dimostrazione facoltativa che di certo non inserisco):

5.2.4 TEOREMA (caso generale perturbazioni)

Sia $A \in \mathbb{R}^{n\times n}$ una matrice non singolare, $x \in \mathbb{R}^n$ soluzione del sistema $Ax = b$, $b \neq 0$ e $\tilde{x} = x + \delta x$ soluzione del sistema perturbato $\tilde{A}\tilde{x} = \tilde{b}$, $\tilde{A} = A + \delta A$, $\tilde{b} = b + \delta b$.
Fissata una norma vettoriale $\|\cdot\|$ in $\mathbb{R}^n$, vale la seguente stima dell'errore "relativo" su x per $k(A) \cdot \frac{\|\delta A\|}{\|A\|} < 1$

$$\frac{\|\delta x\|}{\|x\|} \leq \frac{k(A)}{1 - k(A) \cdot \frac{\|\delta A\|}{\|A\|}} \cdot \left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|b\|} \right)$$

La soluzione numerica di sistemi fortemente mal condizionati viene compensate da un indice di condizionamento enorme con limitati campi applicativi (ambito medico). Se non si usano tecniche particolare di soluzione, piccoli errori di misura possono portare ad errori inaccettabili sulla ricostruzione dell'immagine. Ad esempio per famigle di sistemi con matrici dipendenti da un parametro *h* tale che:

$$e(h) = \|x - x_h\| \longrightarrow 0$$

$$k(A_h) < k(A), \quad k(A_h) \longrightarrow k(A) \quad \text{per } h \to 0$$

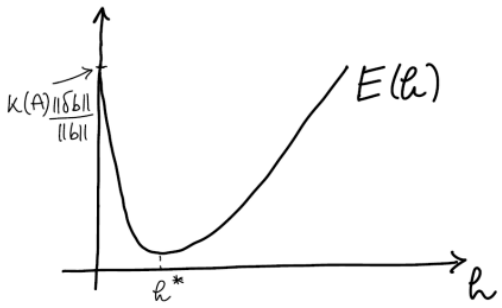
Chiamiamo $\tilde{x}_h$ la soluzione di $A_h\tilde{x}_h = \tilde{b}$ si può scrivere una stima del tipo ($\delta x_h = \tilde{x}_h - x_h$)

$$\begin{aligned} \frac{\|x - \tilde{x}_h\|}{\|x\|} &\leq \frac{\|x - x_h\|}{\|x\|} + \frac{\|\delta x_h\|}{\|x\|} \\ &= \frac{e(h)}{\|x\|} + \frac{\|\delta x_h\|}{\|x_h\|}\frac{\|x_h\|}{\|x\|} \\ &\leq \frac{e(h)}{\|x\|} + k(A_h)\frac{\|\delta b\|}{\|b\|}\left(1 + \frac{e(h)}{\|x\|}\right) = E(h) \end{aligned}$$

dove

$$E(h) \longrightarrow k(A)\frac{\|\delta b\|}{\|b\|} \quad \text{per } h \to 0$$

con tale grafico:



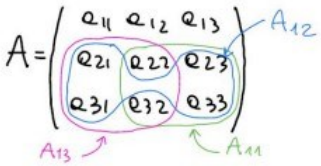
11/05/2022: Lezione 21 e 22 Riduzione a forma triangolare col metodo di eliminazione gaussiana (MEG), calcolo del determinante, Fattorizzazione LU

Siamo interessati a calcolare il determinante attraverso la riduzione di una matrice quadrata non singolare a forma triangolare. Avendo $A \in R^{nxn}$, il determinante di A permette di capire le proprietà di una matrice quadrata, infatti se $det(A) \neq 0$ allora $Ax = b$ ha soluzione unica per ogni b, dunque A *è invertibile*.
Esso può avere una definizione ricorsiva, quindi (n – 1) x (n – 1).
Usiamo la formula di Laplace, così descritta:

$$det(A) = A \text{ se } A \in \mathbb{R}$$

$$det(A) = \sum_{j=1}^{n} (-1)^{i+j} a_{ij} det(A_{ij})$$

(sviluppo secondo la riga i) dove A_{ij} è la sottomatrice $(n-1) \times (n-1)$ ottenuta da A cancellando riga i e colonna j, ad esempio per $n = 3$ e $i = 1$



Da tale definizione, con i conti espliciti, così i determinanti di matrici piccole (2 x 2 o massimo 3 x 3):

$$A \in \mathbb{R}^{2\times 2} \Rightarrow det(A) = a_{11}a_{22} - a_{12}a_{21}$$
$$A \in \mathbb{R}^{3\times 3} \Rightarrow det(A) = a_{11}det(A_{11}) - a_{12}det(A_{12}) + a_{13}det(A_{13})$$
$$= a_{11}(a_{22}a_{33} - a_{23}a_{32}) - a_{12}(a_{21}a_{33} - a_{23}a_{31}) + a_{13}(a_{21}a_{32} - a_{22}a_{31})$$

Vediamo una tabella che descrive il calcolo corrispondente ad 1Gflops (10^9 operazioni aritmetiche al secondo, velocità attuale di un PC) ed 1Pflops (10^{15} operazioni aritmetiche al secondo):

n	$1Gflops$	$1Pflops$
10	10^{-3} sec	10^{-9} sec
15	40 min	$2 \cdot 10^{-3}$ sec
20	10^2 anni	80 min
25	10^9 anni	10^3 anni
100	10^{141} anni	10^{135} anni

Queste sono operazioni su matrici, ma il tempo effettivo di calcolo diventa davvero gigantesco.

Stimiamo il # di moltiplicazioni, chiamandolo c_n^{molt}.

È chiaro che dovendo calcolare n determinanti $(n-1)\times(n-1)$ (il prodotto per $(-1)^{i+j}$ è solo un cambio di segno per $i+j$ dispari) si ha che

$$c_n^{molt} > nc_{n-1}^{molt}$$

e chiaramente

$$c_2^{molt} = 2$$

Quindi

$$\begin{aligned} c_3^{molt} &> 3 \cdot c_2^{molt} = 3 \cdot 2 \\ c_4^{molt} &> 4 \cdot c_3^{molt} = 4 \cdot 3 \cdot 2 \\ &\vdots \\ c_n^{molt} &> n(n-1) \cdot \ldots \cdot 4 \cdot 3 \cdot 2 = n! \end{aligned}$$

Quindi il costo computazionale della formula di Laplace è $c_n^{lapl} > n!$
Usiamo quindi il *metodo di eliminazione gaussiana (meg)*, basato su 2 proprietà fondamentali del determinante che riguardano trasformazioni della matrice, basandoci sul lavoro per righe.

Le proprietà sono:

1. sostituendo alla riga k la somma della riga k con la riga i moltiplicata per uno scalare, in simboli

$$\mathcal{R}_k \underset{\text{assegnazione}}{:=} \underset{\text{riga } k\text{-esima}}{\mathcal{R}_k} + \alpha \underset{\text{riga } i\text{-esima}}{\mathcal{R}_i}, \ \alpha \in \mathbb{R}$$

   il determinante non cambia

2. scambiando due righe il determinante cambia segno

Queste 2 trasformazioni elementari sono alla base del meg (per righe), permettendo di ridurre una matrice quadrata non singolare in forma triangolare. Prendendo un esempio di matrice 3 x 3:

$$A = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 2 & 3 \\ -1 & -3 & 0 \end{pmatrix}$$

L'idea, come noto, è l'uso ripetuto della prima regola per mettere zeri in ogni colonna sotto la diagonale principale, arrivando a questa matrice sotto, che sarebbe triangolare superiore, con stesso determinante:

$$U = \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

Si vede quindi che (usando di nuovo la formula di Laplace, che ricordiamo calcolare la somma dei prodotti degli elementi di una riga o di una colonna per i rispettivi complementi algebrici, sviluppata sull'ultima riga):

$$det(U) = \prod u_{ii} = u_{11}u_{22}u_{33}$$

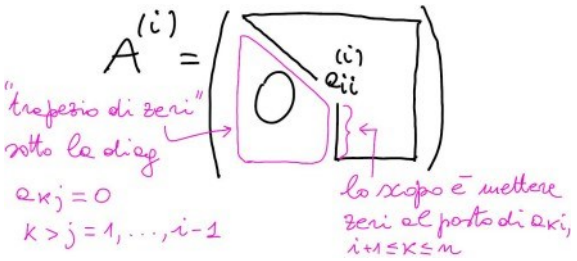
$$\underset{=A^{(1)}}{A} = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 2 & 3 \\ -1 & -3 & 0 \end{pmatrix} \overset{R_2:=R_2+(-2)R_1}{\longrightarrow} \begin{pmatrix} 1 & 2 & 1 \\ 0 & -2 & 1 \\ -1 & -3 & 0 \end{pmatrix}$$

$$\overset{R_3:=R_3+R_1}{\longrightarrow} A^{(2)} \begin{pmatrix} 1 & 2 & 1 \\ 0 & -2 & 1 \\ 0 & -1 & 1 \end{pmatrix} \overset{R_3:=R_3+(-\frac{1}{2})R_2}{\longrightarrow} A^{(3)} = U = \begin{pmatrix} 1 & 2 & 1 \\ 0 & -2 & 1 \\ 0 & 0 & \frac{1}{2} \end{pmatrix}$$

avendo poi che, non essendoci stati scambi di righe:
det(A) = det(U) = -1
Quindi il *meg* consiste in una sequenza di n – 1 trasformazioni:

$$A^{(1)} = A \to A^{(2)} \to A^{(3)} \to \ldots \to A^{(n)} = U$$

dunque al passo i-esimo la struttura schematica di A^i è:



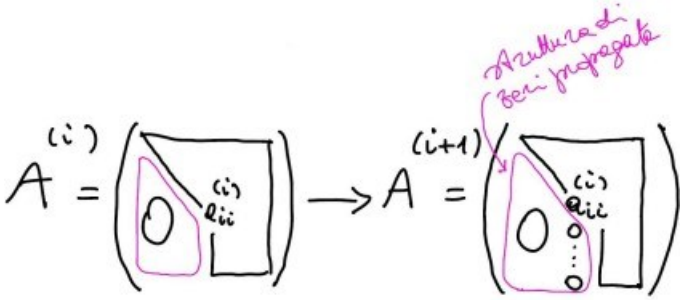
Se $a_{ii}^{(i)} \neq 0$ (cioè se l'elemento diagonale di $A^{(i)}$ è non nullo) si può propagare a destra la struttura del trapezio di zeri con le trasformazioni

$$\mathcal{R}_k^{(i+1)} := \mathcal{R}_k^{(i)} + \left(-\frac{a_{ki}^{(i)}}{a_{ii}^{(i)}}\right) \cdot \mathcal{R}_i^{(i)},\ i+1 \le k \le n$$

che mettono lo zero al posto k, i visto che

$$a_{ki}^{(i+1)} = a_{ki}^{(i)} + \left(-\frac{a_{ki}^{(i)}}{a_{ii}^{(i)}}\right) \cdot a_{ii}^{(i)} = 0$$

Il passo i-esimo per matrici non singolari è:



Si cita a questo punto un algoritmo scritto in pseudocodice del MEG (qui a destra).

Un paio di osservazioni sul *pivoting* (che sarebbe l'operazione che sceglie una serie di elementi come il primo e fa determinate operazioni (es. Gauss-Jordan), rendendo questi elementi tutti uguali a zero/tutti diversi da zero):

- mettere qualcosa di diverso da zero sulla diagonale, senza alterare la struttura di zeri a sinistra e cercando sotto la diagonale per l'eventuale scambio.
  Sappiamo che se $b_{ii} = 0$ e $b_{ki} = 0$ per ogni k allora det(B) = 0.
- si ha poi un motivo di stabilità numerica dato che, usando elementi diagonali piccoli che compaiono nel denominatore nel coefficiente (b_{ki} / b_{ii}), si ha la creazione di numeri grandi ed arrotondati, che portano ad una forte perdita di precisione.

Algorithm 1 PSEUDO-CODICE DEL MEG

Require: $n \in \mathbb{N},\ A \in \mathbb{R}^{n\times n},\ \varepsilon > 0$ (tolleranza)
Ensure: $B = U$ e `determ` $= det(A)$

```
// inizializzazione
B ← A; // B matrice ausiliaria
s ← 0; // contatore del numero di scambi
for i = 1, ..., n − 1 do
  // pivoting
  "cerca p ≥ i : |b_pi| ≥ |b_ki|, i ≤ k ≤ n"
  if |b_pi| < ε then
    "esci con warning: matrice (quasi) singolare"
  end if
  if p > i then
    "scambia R_p con R_i"
    s++;
  end if
  // azzeramenti in colonna i sotto la diagonale
  for k = i + 1, ..., n do
    R_k ← R_k + (−b_ki/b_ii) · R_i
  end for
end for
determ = (−1)^s · ∏_{i=1}^{n} b_ii
```

Output: $B = U$ e determ$=det(A)$

Si pone quindi un'operazione utile per generare una *stabilizzazione* del meg.
Il costo computazionale viene così calcolato:

$$\begin{aligned} c_n^{meg} &\sim \sum_{i=1}^{n-1}\sum_{k=i+1}^{n} 2n \\ &= 2n\sum_{i=1}^{n-1}(n-i) \\ &\underset{j=n-i}{=} 2n\sum_{j=1}^{n-1} j \\ &= 2n\cdot\frac{n(n-1)}{2} \\ &= n^3 - n^2 \sim n^3, \quad n\to\infty \end{aligned}$$

Esso quindi ha costo pari a (n x N)^3; tuttavia non ha molto senso fare operazioni vettoriali su tutti i vettori riga perché all'iterazione i-esima i primi elementi delle righe sono nulli e tali restano nella combinazione lineare. Similmente anche l'operazione di annullamento non va fatta dato che, per effetto di successivi arrotondamenti, conviene assegnare direttamente il valore 0.
L'operazione vettoriale va fatta qui:

$$\mathcal{R}_k[i+1:n] := \mathcal{R}_k[i+1:n] + \left(-\frac{b_{ki}}{b_{ii}}\right) * \mathcal{R}_i[i+1:n]$$

con un costo di $2(n-i)+1$ flops.
Allora

$$\begin{aligned} c_n^{meg} &\sim \sum_{i=1}^{n-1}\sum_{k=i+1}^{n} 2(n-i) \\ &= 2\sum_{i=1}^{n-1}(n-i)^2 \\ &\underset{j=n-i}{=} 2\sum_{j=1}^{n-1} j^2 \sim \frac{2}{3}n^3 \end{aligned}$$

ottenendo il costo asintotico effettivo del *meg* visto che

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

Il meg ha costo computazionale di tipo potenza (o meglio *polinomiale*) in "n", rendendolo adatto a calcolare determinanti di matrici "grandi". Riassumendo si vede che c_n^{meg} è asintoticamente proporzionale ad n^3 :

TABELLA 2 (det con Laplace e meg)

	$1Gflops$		$1Pflops$	
n	*Lapl*	*meg*	*Lapl*	*meg*
10	$10^{-3}sec$	$10^{-6}sec$	$10^{-9}sec$	$10^{-12}sec$
15	$40min$	$3\cdot10^{-6}sec$	$2\cdot10^{-3}sec$	$3\cdot10^{-12}sec$
20	$10^{2}anni$	$8\cdot10^{-6}sec$	$80min$	$8\cdot10^{-12}sec$
25	$10^{-9}anni$	$10^{-5}sec$	$10^{3}anni$	$10^{-11}sec$
100	$10^{141}anni$	$10^{-3}sec$	$10^{135}anno$	$10^{-9}sec$
1000		$1sec$		$10^{-6}sec$

Fino ad ora quindi si è visto che con il pivoting superiamo il problema della comparsa di un elemento nullo sulla diagonale ed ha un effetto stabilizzante sull'algoritmo (evitando quindi la generazione di numeri arrotondati troppo grandi e quindi limitando la perdita di precisione nella sottrazione).
Le operazioni di annullamento e scambio si possono in realtà scrivere in forma matriciale, adoperando matrici elementari di trasformazione.

Cominciamo con lo *scambio di 2 righe*, in particolare operando lo scambio della riga *k* con la riga *i* di una generica matrice B ottenuta moltiplicando a sinistra B per la matrice identità in cui sono state scambiate le righe *k* ed *i.*

Prendiamo un esempio 3 x 3:

$$S_{2,1}B = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} = \begin{pmatrix} b_{21} & b_{22} & b_{23} \\ b_{11} & b_{12} & b_{13} \\ b_{31} & b_{32} & b_{33} \end{pmatrix}$$

ha esattamente l'effetto di scambiare la riga 2 con la riga 1 di B.
Si osservi che $det(S_{2,1}) = -1$ e che $S_{2,1} \cdot S_{2,1} = I$ cioè $S_{2,1}^{-1} = S_{2,1}$.
In generale

$$S_{k,i} = \begin{pmatrix} 1 & 0 & \dots & 0 & \dots & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 & \dots & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots & & \vdots & & \vdots \\ 0 & 0 & \dots & 0 & \dots & 1 & \dots & 0 \\ \vdots & \vdots & & \vdots & & \vdots & & \vdots \\ 0 & 0 & \dots & 1 & \dots & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots & & \vdots & & \vdots \\ 0 & 0 & \dots & 0 & \dots & 0 & \dots & 1 \end{pmatrix} \begin{matrix} \\ \\ \\ \text{i} \\ \\ \text{k} \\ \\ \\ \end{matrix}$$

(i sopra la colonna i, k sopra la colonna k)

e $\quad S_{k,i}^{-1} = S_{k,i}, \quad det(S_{k,i}) = -1.$

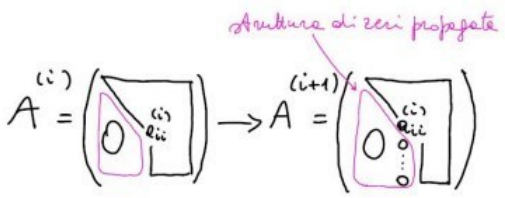
Questa ultima uguaglianza spiega in forma matriciale perché scambiare 2 righe cambia segno al determinante, ricordando che il determinante del prodotto di due matrici è il prodotto dei determinanti:

$$det(S_{k,i}B) = det(S_{k,i})det(B) = -det(B)$$

In merito invece alla *moltiplicazione tra due righe*, l'altra operazione vettoriale chiave del *meg* al passo i-esimo è:

$$\mathcal{R}_k^{(i+1)} := \mathcal{R}_k^{(i)} + (\overbrace{-m_{ki}}^{moltiplicatore})\mathcal{R}_i^{(i)}$$

dove $m_{ki} = \frac{a_{ki}^{(i)}}{a_{ii}^{(i)}}$, $i+1 \le k \le n$, detto "moltiplicatore" ha lo scopo di propagare la struttura di zeri



In generale l'operazione tra righe di una matrice B: $\mathcal{R}_k := \mathcal{R}_k + \alpha\mathcal{R}_i$

corrisponde a moltiplicare a sinistra per la matrice elementare, ottenuta dalla matrice identità mettendo α al punto di 0 come elemento k, i.

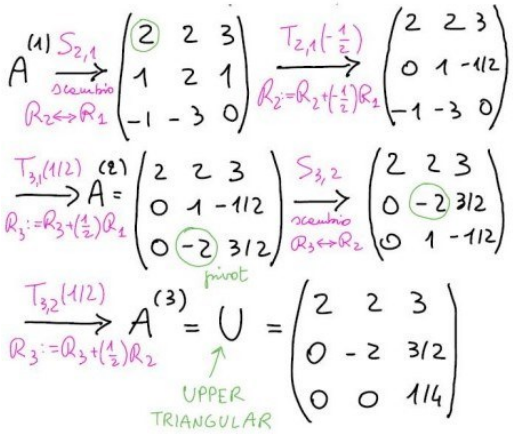
$$T_{k,i} = \begin{pmatrix} 1 & 0 & \dots & 0 & \dots & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 & \dots & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots & & \vdots & & \vdots \\ 0 & 0 & \dots & 1 & \dots & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots & & \vdots & & \vdots \\ 0 & 0 & \dots & \alpha & \dots & 1 & \dots & 0 \\ \vdots & \vdots & & \vdots & & \vdots & & \vdots \\ 0 & 0 & \dots & 0 & \dots & 0 & \dots & 1 \end{pmatrix} \begin{matrix} \\ \\ \\ \text{i} \\ \\ \text{k} \\ \\ \\ \end{matrix}$$

(i sopra la colonna i, k sopra la colonna k)

In un esempio 3 x 3:

$$T_{3,1}(\alpha)B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \alpha & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ \alpha b_{11} + b_{31} & \alpha b_{12} + b_{32} & \alpha b_{13} + b_{33} \end{pmatrix}$$

Si osservi poi che è una matrice triangolare inferiore con 1 sulla diagonale principale e dunque $\det(T_{k,i}(\alpha))=1$ e $T_{k,i}(-\alpha)T_{k,i}(\alpha) = I$ cioè $(T_{k,i}(\alpha))^{-1} = T_{k,i}(-\alpha)$. Abbiamo poi applicando il pivoting:



cioè:

$$U = A^{(3)} = T_{3,2}\left(\frac{1}{2}\right)\ S_{3,2}\ \underbrace{T_{3,1}\left(\frac{1}{2}\right)\ T_{2,1}\left(-\frac{1}{2}\right)\ S_{2,1}A}_{A^{(2)}}$$

avendo che det(U) = det(A) = -1, dato che è avvenuto un numero pari di scambi.
In generale il passo i-esimo del meg diventa:

$$A^{(i+1)} = T_{n,i}(-m_{n,i})\ T_{n-1,i}(-m_{n-1,i}) \dots T_{i+2,i}(-m_{i+2,i})\ T_{i+1,i}(-m_{i+1,i})\ S_{p_i,i}\ A^{(i)}$$

mentre la sequenza completa di trasformazioni è:

$$\begin{aligned} U = A^{(n)} = & T_{n,n-1}(-m_{n,n-1})\ S_{p_{n-1},n-1} \\ & T_{n,n-2}(-m_{n,n-2})\ T_{n-1,n-2}(-m_{n-1,n-2})\ S_{p_{n-2},n-2} \\ & \dots\ S_{p_2,2}\ T_{n,1}(-m_{n,1})\ \dots\ T_{2,1}(-m_{2,1})\ S_{p_1,1}\ A \end{aligned}$$

A questo punto vediamo che il *meg* permette di ottenere una fattorizzazione della matrice di partenza nel prodotto di due matrici triangolari, una superiore ed una inferiore.
Per semplicità ci limitiamo al caso "senza scambi", quindi in cui il pivoting trova l'elemento di max modulo già sulla diagonale.
Ci sono quindi classi di matrici per cui si può dimostrare, quindi:

- matrici a diagonale strettamente dominante

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}| \quad \forall i$$

- matrici simmetriche definite positive

Limitiamoci per capire meglio a matrici 3×3 con

$$U = A^{(3)} = T_{3,2}(-m_{3,2})\ T_{3,1}(-m_{3,1})\ T_{2,1}(-m_{2,1})\ A$$

Posto $\mathcal{L} = T_{3,2}(-m_{3,2})\ T_{3,1}(-m_{3,1})\ T_{2,1}(-m_{2,1})$ abbiamo

$$U = \mathcal{L}A \Rightarrow A = LU \text{ con } L = \mathcal{L}^{-1}$$

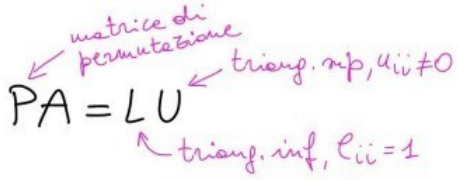
Ma come è fatta L?

$$\begin{aligned} L = \mathcal{L}^{-1} &= \overbrace{(T_{2,1}(-m_{2,1}))^{-1}\ (T_{3,1}(-m_{3,1}))^{-1}\ (T_{3,2}(-m_{3,2}))^{-1}}^{\text{ordine invertito}} \\ &= T_{2,1}(m_{2,1})\ T_{3,1}(m_{3,1})\ T_{3,2}(m_{3,2}) \end{aligned}$$

ricordando che L è una matrice triangolare inferiore:

$$L = \begin{pmatrix} 1 & 0 & 0 \\ m_{2,1} & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m_{3,1} & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & m_{3,2} & 1 \end{pmatrix}$$
$$= \begin{pmatrix} 1 & 0 & 0 \\ m_{2,1} & 1 & 0 \\ m_{3,1} & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & m_{3,2} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ m_{2,1} & 1 & 0 \\ m_{3,1} & m_{3,2} & 1 \end{pmatrix}$$

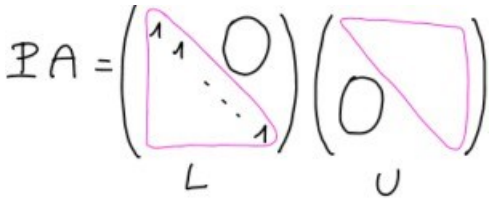
Ciò è vero anche nel caso generale in cui il triangolo sotto la diagonale principale contenga i moltiplicatori del meg. È comunque possibile ottenere una fattorizzazione del tipo:



avendo che

- U è matrice triangolare superiore ottenuta alla fine del meg
- P matrice di permutazione, dunque invertibile contenente solo 0/1 ed ottenuta come prodotto di matrici di scambio
- L matrice triangolare inferiore con $l_{ii} = 1$ e con i moltiplicatori opportunamente rimescolati nel triangolo inferiore sotto la diagonale

e graficamente:



Concludendo, le fattorizzazioni permettono di decomporre una matrice senza struttura nel prodotto di matrici strutturate ed il costo computazionale della fattorizzazione PA = LU ottenuta con il *meg* coincide con il costo del *meg* stesso, dato che P ed L sono costruite con gli scambi e con i moltiplicatori utilizzati durante il processo di eliminazione.
Dunque:

$$c_n^{LU} = c_n^{meg} \sim \frac{2}{3}n^3, \quad n \to \infty$$

16/05/2022: Lezione 23 - MEG e soluzione di sistemi lineari: sistemi triangolari, uso della fattorizzazioneLU, inversione di matrici, MEG e condizionamento

Si parte dal riconsiderare *PA=LU*, come metodo di riduzione di una matrice triangolare superiore come metodo per la ***fattorizzazione LU*** (Lower Upper, prodotto di una triangolare inferiore per una triangolare superiore).
L'approccio standard è la risoluzione di un sistema lineare *Ax = b* con A non singolare aggiungendo ad *A* una colonna uguale al vettore termine noto *b* applicando le trasformazioni che portano alla forma cosiddetta *orlata* (perché aggiungiamo una colonna alla matrice):

$$[A^{(1)}|b^{(1)}] = [A|b] \to [A^{(2)}|A^{(2)}] \to$$
$$\ldots \to [A^{(n)}|b^{(n)}] = [U|\beta]$$

Le trasformazioni vengono applicate anche al vettore termine noto, ottenendo un sistema triangolare

superiore *Ux = β*, equivalente all'esistenza del sistema lineare di partenza *Ax = b* (dato che hanno la stessa soluzione) e a moltiplicare un sistema per una matrice invertibile.

Una volta che si ha il sistema *Ux = β* (immagine sotto), si applica il metodo della *sostituzione all'indietro/backward substitution* (consistente nella risoluzione di equazioni lineari algebriche su *Ux = β*):

$$\begin{pmatrix} u_{11} & u_{12} & \dots & u_{1n-1} & u_{1n} \\ & u_{22} & \dots & u_{2n-1} & u_{2n} \\ & \ddots & & \vdots & \vdots \\ & & \ddots & u_{n-1n-1} & u_{n-1n} \\ & & & & u_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{n-1} \\ \beta_n \end{pmatrix}$$

avendo una strutturazione progressiva così data per le incognite; l'ultima equazione ne ha una sola:

$$\begin{cases} \beta_1 = u_{11}x_1 + u_{12}x_2 + \dots + u_{1n-1}x_{n-1} + u_{1n}x_n \\ \beta_2 = u_{22}x_2 + \dots + u_{2n-1}x_{n-1} + u_{2n}x_n \\ \vdots \\ \beta_{n-1} = u_{n-1n-1}x_{n-1} + u_{n-1n}x_n \\ \beta_n = u_{nn}x_n \end{cases} \qquad x_n = \frac{\beta_n}{u_{nn}}$$

sostituendola nella penultima equazione, tale che diventi nella sola incognita x_{n-1}

$$u_{n-1n-1}x_{n-1} + u_{n-1n}\overbrace{x_n}^{=\beta_n/u_{nn}} = \beta_{n-1}$$

da cui

$$x_{n-1} = \frac{1}{u_{n-1n-1}}(\beta_{n-1} - u_{n-1n}\overbrace{x_n}^{=\beta_n/u_{nn}})$$

ripetendo il procedimento a tutte le righe calcolandone le incognite:

$$x_i = \frac{1}{u_{ii}}\left(\beta_i - \sum_{j=i+1}^{n} u_{ij}x_j\right), \; i = n, n-1, \dots, 1$$

avendo il costo computazionale della sostituzione all'indietro dato dalle *n* divisioni, *n – i* moltiplicazioni e *n – i* somme algebriche:

$$\begin{aligned} c_n^{BS} &= n + \sum_{i=1}^{n-1} 2(n-i) \\ &= n + 2\sum_{j=1}^{n-1} j \\ &= n + 2\frac{n(n-1)}{2} = n^2 \; flops \end{aligned}$$

avendo che il costo computazione della soluzione di un sistema col MEG:

$$\begin{aligned} c_n^{Sist1} &= c_n^{meg} + \underbrace{2\sum_{i=1}^{n-1}}_{=\frac{2n(n-1)}{2}\sim n^2} (n-i) + c_n^{BS} \\ &\sim \frac{2}{3}n^3 + n^2 + n^2 = \frac{2}{3}n^3 + 2n^2 \; flops \end{aligned}$$

Possiamo usare il MEG per risolvere sistemi lineari, passando per la fattorizzazione LU, considerando sempre un sistema *Ax = b* con det ≠ 0 e applicando il MEG con il pivoting ottengo:

$$PA = LU$$

Il sistema di partenza evidentemente è equivalente al sistema

$$PAx = LUx = \underline{P}b \qquad \begin{cases} Ly = \underline{P}b \\ Ux = y \end{cases}$$

Per poter risolvere *LUx = <u>P</u>b* si risolve in cascata la coppia di sistemi, che sono triangolari:

Analogamente un sistema triangolare inferiore si può risolvere con la *sostituzione in avanti/forwardsubstitution,* partendo dalla forma estesa di *Ly = c*:

$$\begin{pmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}$$

cioè

$$\begin{cases} l_{11}y_1 = c_1 \\ l_{21}y_2 + l_{22}y_2 = c_2 \\ \vdots \\ l_{n1}y_1 + l_{n2}y_2 + \cdots + l_{nn}y_n = c_n \end{cases}$$

dove $l_{ii} \neq 0 \ \forall i$ (L non singolare).

Al passo i-esimo in formule:

$$y_i = \frac{1}{l_{ii}}\left(c_i - \sum_{j=1}^{i-1} l_{ij}y_j\right), \ i = 1, \dots, n$$

con un costo che per la sostituzione in avanti è n^2 flops, ma per una soluzione di un sistema via MEG ed LU:

$$c_n^{Sist2} = c_n^{meg} + c_n^{FS} + c_n^{BS} \sim \frac{2}{3}n^3 + 2n^2 \ flops$$

che coincide con il costo della soluzione standard via MEG, dato che il risultato concorda con le trasformazioni del metodo all'ultima colonna della matrice orlata; nella pratica è un modo di calcolare β aposteriori dopo aver ridotto A con forma triangolare e aver calcolato i moltiplicatori.
Situazione utile nella pratica è il calcolo e risoluzione di diversi sistemi con la stessa matrice non singolare A,con termine noto variabile e con M grande:

$$Ax^{(k)} = b^{(k)}, \ 1 \leq k \leq M$$

Applicare il metodo standard M volte avrebbe un costo $2/3n^3M$ flops.
È molto più efficiente calcolare una volta per tutte la fattorizzazione PA = LU per risolvere la sequenza diM sistemi triangolari accoppiati:

$$\begin{cases} Ly^{(k)} = \underline{P}b^{(k)} \\ Ux^{(k)} = y^{(k)} \end{cases}$$

con un costo complessivo in flops

$$c_n^{(2)} \sim \frac{2}{3}n^3 + 2n^2M \ll \frac{2}{3}n^3M \sim c_n^{(1)}$$

Ora parliamo dell'*inversione di matrici*, usando una proprietà del prodotto matrice-vettore, interpretabilecome combinazione lineare delle colonne di *B* tramite i coefficienti di *c*, quindi:

$$z = Bc = c_1 \underbrace{C_1(B)}_{\text{colonna 1 di } B} + \cdots + c_n \underbrace{C_n(B)}_{\text{colonna } n \text{ di } B}$$

Se applichiamo questa osservazione con

$$c = e^{(i)} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

cioè $c_i = 1$ e $c_j = 0$, $j \neq i$, otteniamo

$$Be^{(i)} = C_i(B)$$

Allora per $B = A^{-1}$

$$C_i(A^{-1}) = A^{-1}e^{(i)} \Leftrightarrow AC_i(A^{-1}) = e^{(i)}$$

cioè $C_i(A^{-1})$ è la soluzione di

$$Ax^{(i)} = e^{(i)},\ 1 \leq i \leq n$$

È possibile calcolare l'inversa "per colonne" risolvendo *M = n* sistemi lineari, ognuno con una matrice A, incui il termine noto varia tra gli n vettori coordinati della base canonica.
Usando n volte in calcolo del metodo standard avremmo un costo $2/3n^4$.

Invece calcolando una volta col *meg* la fattorizzazione LU e risolvendo le $M = n$ coppie di sistemi triangolari

$$\begin{cases} Ly^{(i)} = \underline{P}e^{(i)} \\ Ux^{(i)} = y^{(i)} \end{cases}$$

si ha un costo

$$c_n^{(2)} \sim \frac{2}{3}n^3 + 2n^2n = \frac{8}{3}n^3\ flops$$

cioé l'inversa si può calcolcare col *meg* (via LU) a costo cubico nella dimensione (e questo è in sostanza il metodo che usano tutti i sistemi di calcolo, ad es. il Matlab, quando si usa una function predefinita tipo "inv(A)").

Il MEG discute alcuni aspetti:
la generazione di numeri arrotondati sempre più grandi genera una potenziale instabilità, in assenza di controllo sull'elemento diagonale. Il problema viene risolto dal *pivoting, con effetto stabilizzante,* avendoche PA è vicinissima ad LU in termini relativi:

$$\frac{\|\underline{P}A - LU\|_2}{\|\underline{P}A\|_2} \approx \varepsilon_M$$

Tuttavia, quando si va a risolvere un sistema, l'instabilità è problematica se A è mal condizionata, prendendo ad esempio la *matrice di Hilbert* (matrice scarsamente condizionata che ha come terminegenerale 1/i+j-1):

$$H_n = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+1} \\ \vdots & & & & \\ \frac{1}{n} & \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{2n-1} \end{pmatrix} \in \mathbb{R}^{n \times n}$$

con H_M simmetrica e definita positiva;per questa ultima caratteristica, il pivoting del MEG non agisce nel processo di eliminazione.
Per esempio con *lu* in Matlab pern=13:

$$\frac{\|H_{13} - \tilde{L}\tilde{U}\|_2}{\|H_{13}\|_2} \approx 3.6 \cdot 10^{-17}$$

dove $\tilde{L}$ e $\tilde{U}$ sono i fattori triangolari calcolati in precisione doppia. Invece risolvendo il sistema $H_{13}x = b = H_{13}u$ con $u = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbb{R}^{13}$ la cui soluzione esatta per costruzione $x = u$, detta $\tilde{x} = x + \delta x$ la soluzione calcolata in Matlab

$$\frac{\|\tilde{x} - x\|_2}{\|x\|_2} = \frac{\|\delta x\|_2}{\|x\|_2} \approx 17.2$$

cioè si fa un errore relativo $> 1000\%$!

L'errore è dovuto alla crescita esponenziale del condizionamento di H_n in *n*, dato dal fatto che il MEGcorrisponde alla risoluzione del sistema:

$$\tilde{H}_{13}\tilde{x} = \tilde{L}\tilde{U}\tilde{x} = b$$

e prendendo la stima (lezione 20):

$$\frac{\|\delta x\|}{\|\tilde{x}\|} \leq k(A)\frac{\|\delta A\|}{\|A\|}$$

aspettandosi quindi una amplificazione che spieghi bene la perdita di tutta la

$$\frac{\|\delta H_{13}\|_2}{\|H_{13}\|_2} \approx 3.6 \cdot 10^{-17}$$

precisione, come:Dunque con matrici molto mal condizionate:

- si lavora in precisione estesa, specie se presenti altre fonti di errore oltre gli arrotondamenti
- si rinuncia ad usare metodi come il MEG, ricorrendo ad algoritmi speciali per sistemi malcondizionati

16/05/2022: Lezione 24 - Sistemi lineari sovradeterminati, minimi quadrati, sistema delle equazioninormali, fattorizzazione QR

Parliamo di sistemi sovradeterminati, con più righe che colonne, in particolare un sistema $R^{m \times n}$ si ha m > n.

$$\begin{pmatrix} a_{11} & \dots & a_{1n} \\ a_{21} & \dots & a_{2n} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_n \end{pmatrix}$$

Generalmente questi sistemi non hanno soluzione in senso classico, dato che non è detto esista unax ∈ R^n | Ax = b, in particolare:

$$Ax = x_1 C_1(A) + \cdots + x_n C_n(A)$$

ha soluzione se e solo se

$$b \in \langle C_1(A), \dots, C_n(A) \rangle \subset \mathbb{R}^m$$

Dato che in generale non esiste una soluzione tale che $dist_2$ (b, Ax) = $||b - Ax||_2 = 0$, possiamo cercare *x* inmodo tale da minimizzare la distanza e risolvere il problema di minimo in R^n:

$$\min_{x \in \mathbb{R}^n} \phi(x), \ \phi(x) = \|b - Ax\|_2^2$$

dove

$$\begin{aligned} \phi(x) &= \sum_{i=1}^{m} (b - Ax)_i^2 = \sum_{i=1}^{m} (b_i - (Ax)_i)^2 \\ &= \sum_{i=1}^{m} \left(b_i - \left(\sum_{j=1}^{n} a_{ij} x_j \right) \right)^2 \end{aligned}$$

In questa situazione, l'approssimazione polinomiale ai minimi quadrati realizza:

$$\begin{aligned} \min_{c \in \mathbb{R}^{k+1}} \phi(c), \ \phi(c) &= \sum_{i=1}^{N} (y_i - (Vc)_i)^2 \\ &= \|y - Vc\|_2^2 \end{aligned}$$

Il problema che affrontiamo è quello ai *minimi quadrati* per un sistema sovradeterminato.

Citiamo il *teorema per il sistema delle equazioni "normali" per la soluzione ai minimi quadrati di un sistemalineare sovradeterminato*:

$$\text{Il vettore } x \in \mathbb{R}^n \text{ minimizza } \phi(x) = \|b - Ax\|_2^2$$
$$\Updownarrow$$
$$\text{risolvere il sistema lineare } n \times n \ A^t A x = A^t b \text{ (sistema delle equazioni normali)}$$

In merito alla dimostrazione:

$$\begin{aligned} \phi(x + z) &= \|b - A(x + z)\|_2^2 \\ &= (b - A(x + z), b - A(x + z)) \quad \longleftarrow \text{prod. scalare in } \mathbb{R}^m \\ &= (b - Ax - Az, b - Ax - Az) \\ &= (b - Ax, b - Ax) - 2(Az, b - Ax) + (Az, Az) \\ &= \phi(x) + 2(z, A^t(Ax - b)) + \|Az\|_2^2 \end{aligned}$$

Successivamente:

- "⇑" Se $A^tAx = A^tb$ allora

$$\phi(x+z) = \phi(x) + \overbrace{\|Az\|_2^2}^{\geq 0} \geq \phi(x)\ \forall z$$

cioè $\phi(x)$ è minimo

- "⇓" Se $\phi(x)$ è un minimo allora $\forall \varepsilon > 0$ e $\forall v \in \mathbb{R}^n$, $\|v\|_2 = 1$

$$\phi(x+\varepsilon v) = \phi(x) + 2(\varepsilon v, A^t(Ax-b)) + \|\varepsilon v\|_2^2 \geq \phi(x)$$

cioè

$$2\varepsilon(v, A^t(Ax-b)) + \varepsilon^2 \geq 0$$

e dividendo per ε

$$2(v, A^t(Ax-b)) + \varepsilon \geq 0\ \forall \varepsilon, v$$

da cui per $\varepsilon \to 0$

$$(v, A^t(Ax-b)) \geq 0\ \forall v$$

Ma allora prendendo $-v$

$$(-v, A^t(Ax-b)) \geq 0\ \forall v$$

cioè

$$(v, A^t(Ax-b)) \leq 0\ \forall v$$

e quindi

$$(v, A^t(Ax-b)) = 0\ \forall v$$

da cui $A^t(Ax-b) = 0$ perché l'unico vettore ortogonale a tutti i versori è il vettore nullo, cioè $A^tAx = A^tb$

Possiamo quindi dire che la matrice A^tA sia quadrata, simmetrica e semidefinita positiva, infatti:

$$(A^tAz, z) = (Az, Az) = \|Az\|_2^2 \geq 0$$

Conseguentemente A ha rango max = n e la soluzione ai minimi quadrati del sistema sovradeterminato èunica. A^tA tende ad essere mal condizionata, osservando che:

$$k_2(A^tA) = \|A^tA\|_2\|(A^tA)^{-1}\|_2 = \frac{max\lambda_i(A^tA)}{min\lambda_i(A^tA)}$$

essendo A^tA simmetrica e definita positiva ed è questo rapporto di autovalori che tende ad essere grande, perché

$$k_2(A^tA) = (k_2(A))^2 \gg k_2(A) > 1$$

Consideriamo poi il caso di A quadrata e simmetrica:

Per "intuire" cosa succede però consideriamo il caso di A quadrata e simmetrica: allora $A^tA = A^2$ e $\lambda_i(A^tA) = \lambda_i(A^2) = (\lambda_i(A))^2$ sono gli autovalori di A^tA, quindi essendo A^2 simmetrica

$$k_2(A^tA) = k_2(A^2) = \frac{max\lambda_i(A^2)}{min\lambda_i(A^2)}$$
$$= \left(\frac{max|\lambda_i(A)|}{min|\lambda_i(A)|}\right)^2 = (k_2(A))^2 \gg k_2(A)$$

Per la soluzione di $A^tAx = A^tb$ evita di usare direttamente la matrice mal condizionata A^tA ed è

5.6.3 TEOREMA (fattorizzazione QR di una matrice rettangolare)

Sia $A \in \mathbb{R}^{m\times n}$, $m \geq n$ tale che $rango(A) = n$
Allora $\exists Q \in \mathbb{R}^{m\times n}$ ortogonale (cioè $Q^tQ = I$) e $\exists R \in \mathbb{R}^{n\times n}$ tringolare superiore con $det(R) \neq 0$ tali che

$$A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ a_{21} & \dots & a_{2n} \\ \vdots & & \\ \vdots & & \\ a_{m1} & \dots & a_{mn} \end{pmatrix} = QR = \begin{pmatrix} q_{11} & \dots & q_{1n} \\ q_{21} & \dots & q_{2n} \\ \vdots & & \\ \vdots & & \\ q_{m1} & \dots & q_{mn} \end{pmatrix} \begin{pmatrix} r_{11} & \dots & r_{1n} \\ & \ddots & \vdots \\ 0 & & r_{nn} \end{pmatrix}$$

basata sullafattorizzazione QR che scrive una matrice come prodotto di una matrice ortogonale per una matrice triangolare superiore non singolare.

Q è ortogonale, quindi Q^tQ= I ed

Osserviamo che $Q^t \in \mathbb{R}^{n\times m}$ ha per righe le colonne di Q, quindi $Q^tQ = I \in \mathbb{R}^{n\times n}$ cioè

$$\underset{\text{prodotto rige } i\text{-col } j}{\mathcal{R}_i(Q^t)\mathcal{C}_j(Q)} = \underset{\text{prodotto scalare}}{(\mathcal{C}_i(Q), \mathcal{C}_j(Q))} = \underset{\text{delta di Kronecker}}{\delta_{ij}}$$

ovvero le colonne di Q sono vettori ORTONORMALI di $\mathbb{R}^m$.
Siccome R è invertibile

$$QRR^{-1} = Q = AR^{-1}$$

Ora, si può dimostrare l'inversa di una matrice triangolare è triangolare dello stesso tipo (accettiamo questo risultato).
Quindi

$$R^{-1} = \begin{pmatrix} \varrho_{11} & \varrho_{12} & \cdots & \varrho_{1n} \\ & \varrho_{22} & \cdots & \varrho_{2n} \\ & & \ddots & \vdots \\ & & & \varrho_{nn} \end{pmatrix}$$

Per l'interpretazione del prodotto matrice vettore, cioè il prodotto di A per le colonne di R^{-1}:

$$\begin{aligned}
\mathcal{C}_1(AR^{-1}) &= \mathcal{C}_1(Q) = \varrho_{11}\mathcal{C}_1(A) \\
\mathcal{C}_2(AR^{-1}) &= \mathcal{C}_2(Q) \\
&= \varrho_{12}\mathcal{C}_1(A) + \varrho_{22}\mathcal{C}_2(A) \\
\mathcal{C}_3(AR^{-1}) &= \mathcal{C}_3(Q) \\
&= \varrho_{13}\mathcal{C}_1(A) + \varrho_{23}\mathcal{C}_2(A) + \varrho_{33}\mathcal{C}_3(A) \\
&\vdots \\
\mathcal{C}_j(AR^{-1}) &= \mathcal{C}_j(Q) \\
&= \sum_{i=1}^{j} \varrho_{ij}\mathcal{C}_i(A)
\end{aligned}$$

dunque la colonna j-esima di AR^{-1} = Q si ottiene come combinazione lineare delle prime j colonne di A edavendo Q ortogonale, la combinazione lineare ha l'effetto di ortonormalizzare le colonne.

L'algoritmo non è stabile in aritmetica floating-point al crescere del numero di vettori; inoltre, i interessafar vedere come usare la fattorizzazione QR per risolvere il sistema delle equazioni normali per i minimi quadrati.

Sia $A \in \mathbb{R}^{m \times n}$, $m \geq n$, $rango(A) = n$. Fattorizzando $A = QR$, si ha che

$$A^tA = (QR)^tQR = R^tQ^tQR = R^tIR = R^tR$$

e

$$A^tb = R^tQ^tb$$

quindi il sistema $A^tAx = A^tb$ diventa

$$R^tRx = R^tQ^tb$$

ma essendo R (e quindi R^t) invertibile

$$(R^t)^{-1}R^tRx = Rx = (R^t)^{-1}R^tQ^tb = Q^tb$$

cioè il sistema $A^tAx = A^tb$ equivale al sistema triang. sup.

$$Rx = d = Q^tb$$

che si può facilmente risolvere con la sostituzione all'indietro.

Non si hanno molti vantaggi da un punto di vista computazionale, tuttavia si ha un grosso vantaggio dalpunto di vista della stabilità, perché:

$k_2(A^tA) = (k_2(A))^2 \gg k_2(R)$

Andremo quindi sempre a risolvere un sistema perturbato $\tilde{R}\tilde{x} = \tilde{d}$ con R molto meglio condizionata di A^tA.

Si ha che $k_2(R) = k_2(A)$.

Per intuirlo, consideriamo di nuovo il caso di A quadrata $(m = n)$ e simmetrica: allora da

$$A = QR$$

abbiamo che

$$\|R\|_2 = \|Q^tA\|_2 \leq \|Q^t\|_2\|A\|_2 = \|A\|_2$$

perché Q^t è ortogonale e

$$\|Q^t\|_2 = \sqrt{max|\lambda_i(QQ^t)|} = \sqrt{max|\lambda_i(I)|} = 1$$

e allo stesso modo $(Q^{-1} = Q^t)$

$$\|R^{-1}\|_2 = \|(Q^tA)^{-1}\|_2 = \|A^{-1}Q\|_2 \leq \|A^{-1}\|_2\|Q\|_2 = \|A^{-1}\|_2$$

quindi $k_2(R) \leq k_2(A)$ (in realtà si può dire che $k_2(R) = k_2(A)$)