Metodi Matematici per l'Ingegneria

Modulo MAT08

A.A. 2019-2020

3 CFU

Metodi Numerici

Nozioni Introduttive

Docente: Domenico Vitulano

Email: domenico.vitulano@sbai.uniroma1.it

Ufficio: Via A. Scarpa,
Pal. RM002, I piano, Stanza n. 11b
Tel. 06 49766555

Ricevimento:
https://www.sbai.uniroma1.it/vitulano-domenico/analisi-numerica/2019-2020

Programma sintetico

I. Nozioni Introduttive.
Errori e loro propagazione. Rappresentazione dei numeri. Condizionamento di un problema. Stabilità degli algoritmi. Alcuni cenni su MATLAB.

II. Soluzione di equazioni e sistemi di equazioni non lineari
Separazione delle radici. Metodo di bisezione: convergenza, criteri di arresto. Metodi di linearizzazione: metodo di Newton-Raphson, cenni sul metodo delle secanti. Considerazioni in Matlab.
Metodi iterativi a un punto: convergenza, proprietà della successione di approssimazioni. Considerazioni in Matlab.

III. Algebra lineare numerica
Generalità sui sistemi lineari. Condizionamento di un sistema lineare. Metodi diretti: Metodo di eliminazione di Gauss. Fattorizzazione LU.
Generalità sui metodi iterativi: Metodi di Jacobi, di Gauss-Seidel. Criteri di convergenza. Considerazioni in Matlab.

Programma sintetico

I. . . .
II. . . .
III. . . .

IV. Approssimazione di dati e funzioni
Generalità sul problema dell’approssimazione: spazi di funzioni approssimanti, criteri di approssimazione, fonti di errore nell’approssimazione. Espressione di Lagrange del polinomio interpolatore ed espressione dell’errore di troncamento.
Funzioni Spline. Spline naturali. Approssimazione polinomiale ai minimi quadrati. Considerazioni in Matlab.

V. Integrazione numerica
Formule di quadratura interpolatorie: concetti base, grado di precisione, resto ed errore di propagazione. Formule di Newton–Cotes: formula del trapezio, formula di Cavalieri-Simpson. Convergenza delle formule di quadratura. Considerazioni in Matlab.

Programma sintetico

I. . . .
II. . . .
III. . . .
IV. . . .
V. . . .

VI. Soluzione numerica di equazioni differenziali ordinarie
Soluzione numerica del problema di Cauchy, definizioni e concetti base. Errore di troncamento locale, errore globale. Consistenza, stabilità, convergenza dei metodi. Metodi one-step espliciti: metodo di Eulero-Cauchy, Metodo di Heun, Metodi di Runge Kutta. Convergenza dei metodi one-step espliciti. Sistemi di equazioni differenziali ordinarie del primo ordine. Cenni su problemi ai limiti. Considerazioni in Matlab.

VII. Soluzione numerica di equazioni alle derivate parziali
Generalità sulle p.d.e.. Linee caratteristiche. Cenni su schemi numerici per la soluzione di p.d.e. del primo ordine. Considerazioni in Matlab.

Testi consigliati:

Calcolo Numerico, L. Gori, Ed. Kappa, 2006

Esercizi di Calcolo Numerico, L. Gori-M.L. Lo Cascio, F. Pitolli,
Ed. Kappa, 2007

Per consultazione:
S. C. Chapra, R. P. Canale, Numerical Methods for engineers,
Calcolo scientifico, Springer, McGraw Hill, 2010

Il materiale didattico sarà disponibile sul sito:
https://www.sbai.uniroma1.it/vitulano-domenico/analisi-numerica/2019-2020

Obiettivi del corso

Fornire una panoramica dei metodi numerici fondamentali per la soluzione di alcuni problemi di maggior interesse nel settore dell’ingegneria

e

di alcuni “elementi di programmazione” orientati all'uso di algoritmi risolutivi in un ambiente di calcolo integrato

Risultati

- individuare un metodo numerico adatto a risolvere alcuni problemi test
- analizzarne e formularne la soluzione in modo “algoritmico”
- scelta e uso di algoritmi dedicati in Matlab

Cosa è il CALCOLO NUMERICO?

E' quella branca della matematica che

costruisce e analizza

i metodi numerici

adatti a risolvere, con l'aiuto del calcolatore,

differenti problemi matematici

che nascono in varie discipline
(ingegneria, economia, biologia, ...)

e che non possono essere risolti analiticamente

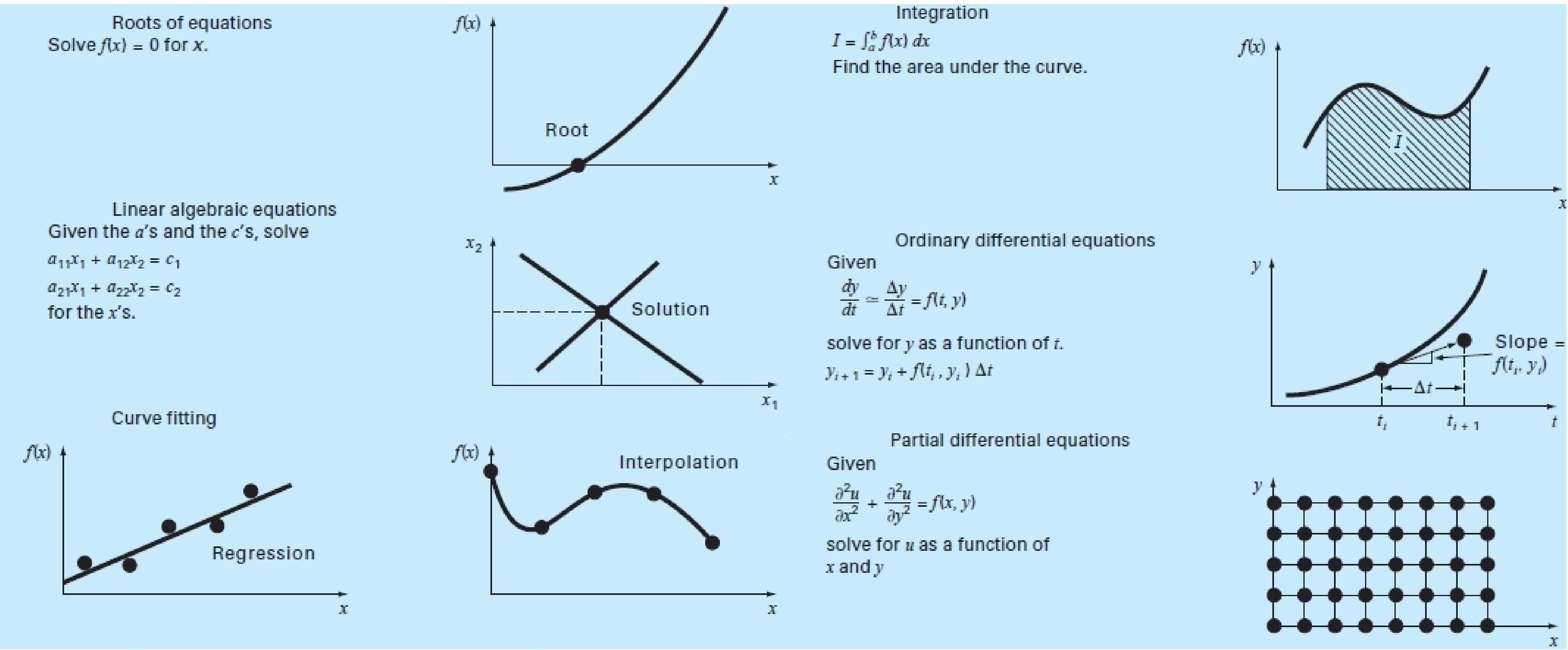
Perché conoscerlo

- Un ingegnere, durante la sua carriera, può aver necessità di usare software commerciali che usano metodi numerici. E' necessario conoscere la teoria e i concetti alla base di questi metodi al fine di poterli usare al meglio e saperne interpretare correttamente i risultati

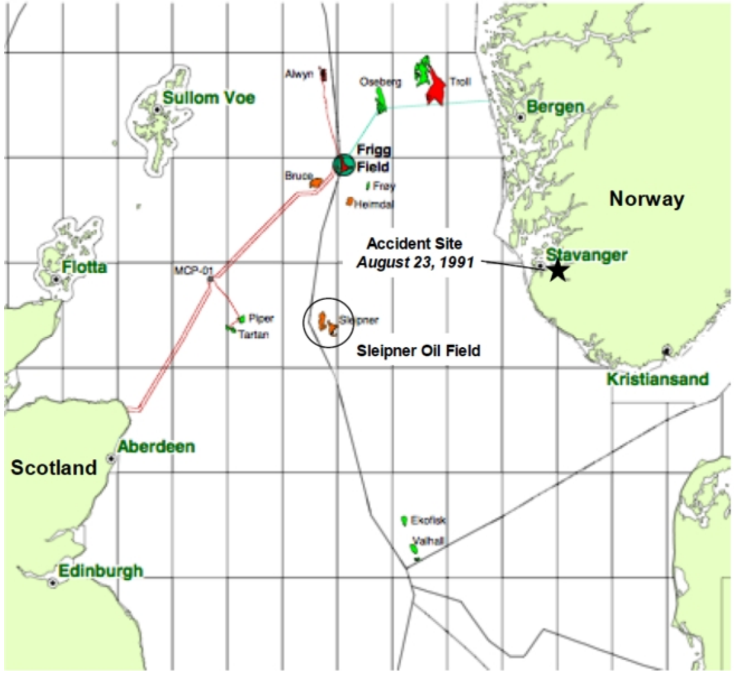
- I metodi numerici offrono strumenti potenti per la soluzione di problemi, soprattutto a seguito dello sviluppo del calcolatore elettronico

- E' possibile scrivere un metodo numerico per un problema specifico che non può essere risolto con i metodi esistenti

Programma sintetico



Perchè studiare Calcolo Numerico: Piattaforma Sleipner



Disastri dovuti ad errori nelle simulazioni numeriche

Affondamento della piattaforma Sleipner A (23 Agosto 1991)

La piattaforma è affondata nel mare del Nord al largo della Norvegia a seguito di un'operazione di zavorramento, provocando un effetto sismico del terzo grado della scala Richter e una perdita stimata in 700 milioni di dollari

Causa: Utilizzo incauto del codice elementi finiti NASTRAN nella fase di progettazione (modello elastico lineare della tricella) in cui gli sforzi di taglio sono stati sottostimati del 47%.

Alcune pareti di cemento non erano abbastanza spesse!!!

Analisi a posteriori: rottura ad una profondità di 62m, in buon accordo con la profondità (65m) a cui si è realmente verificata

Altro esempio: Millennium Bridge (2000) (18.2 + 5 milioni sterline)

Disastri dovuti ad errori nelle simulazioni numeriche

CHIMICA

1976, 10 luglio - Diossine - Italia - Seveso - Nello stabilimento della ICMESA (Givaudan) esplode un reattore, disperdendo nell'ambiente TCDD tetracloro-p-dibenzodiossina. Seimila residenti esposti ai danni a causa del disastro di Seveso.

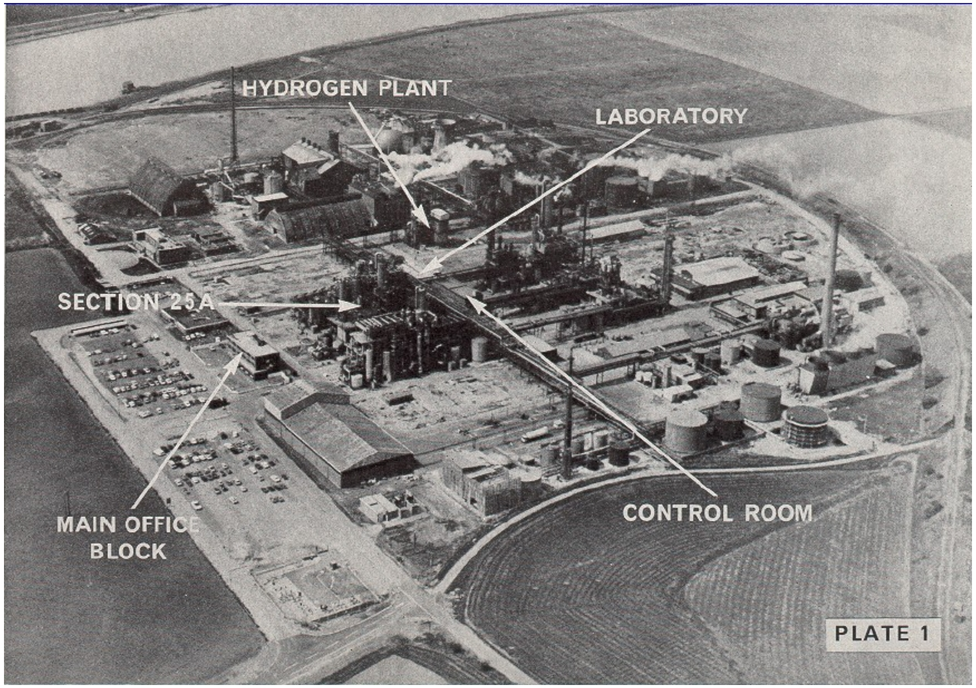
1984 - Metilisocianato - India - Bhopal - esplosione nello stabilimento della Union Carbide con dispersione di 40 tonnellate di Metilisocianato - 100.000 feriti, 2.000 morti a causa del disastro di Bhopal.

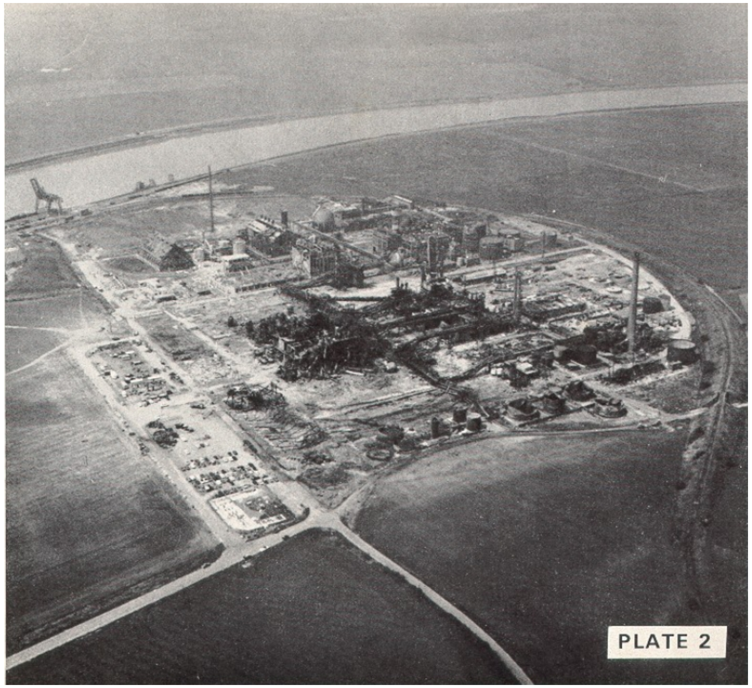
2001 - NH_4NO_3 - Francia - Tolosa - Esplosione da nitrato d'ammonio alla AZF, 31 morti e 2442 feriti

. . .

Stabilimento Nypro di Flixborough (1974)

Evento: *Rilascio accidentale (6 reattori) di cicloesano (sostanza chimica con infiammabilità simile alla benzina) ad alta pressione e temperatura e conseguente esplosione.*
Conseguenze: *decesso di 28 persone, ferimento di altre 104. 100.000 mq interamente distrutti e circa 1800 case e 170 tra attività commerciali e fabbriche furono danneggiate.*
Cause: *...; "la connessione di bypass venne installata senza alcuna valutazione di sicurezza, dimensionamento meccanico e supervisione da parte di ingegneri chimici esperti"*





Problema

Sia U la velocità iniziale di swing, calcolare l'angolo ottimo per cui la distanza percorsa da una pallina da golf, prima che tocchi terra, sia massima

Descrizione del problema fisico:

- misura delle caratteristiche della pallina da golf: raggio, massa, geometria
- forze in gioco: peso della pallina, resistenza dell'aria, vento, ...
- approssimazioni: la palla è una sfera, assenza di vento
- caratteristiche del tiro (swing): velocità iniziale e angolo iniziale

Formulazione di un modello matematico:

- tradurre il problema fisico in equazioni (per es. leggi di conservazione)
- usare le assunzioni fatte nel modello fisico per semplificare il modello matematico

Soluzione del modello matematico (ben posto e ben condizionato!)

- soluzione analitica (raramente)
- ***soluzione numerica*** (***S E M P R E***)

Interpretazione e validazione

- significato fisico del risultato
- identificare e stimare le "fonti" di errore

Formulazione di un modello matematico

Parametri: massa M, raggio a,
coefficiente di resistenza dell' aria $c = \frac{\pi a^2 \rho}{2} C_D$

Variabili: posizione **x** = (x,y), velocità **v** = ($\dot{x}$, $\dot{y}$)

Legge della fisica: seconda legge di Newton

$$\begin{cases} M \frac{d^2 x}{dt^2} = \sum F_{ext} \Rightarrow M \frac{dv}{dt} = Mg - c|v|v \\ x(t=0) = (0,0), \quad v(t=0) = (U\cos\alpha, U\sin\alpha) \end{cases}$$

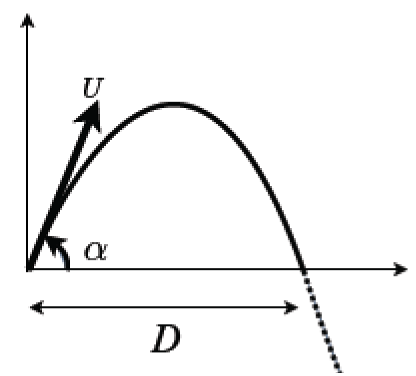
Formulazione di un modello matematico

Trascurando la resistenza dell'aria

$$\begin{cases} M\dfrac{d^2x}{dt^2} = \sum F_{ext} \Rightarrow M\dfrac{dv}{dt} = Mg \cancel{- c|v|v} \\ x(t=0) = (0,0), \qquad v(t=0) = (U\cos\alpha, U\sin\alpha) \end{cases}$$

Le equazioni del moto diventano

$$\begin{cases} \ddot{x} = 0 \\ \ddot{y} = -g \end{cases} \qquad \begin{cases} x(t=0) = 0,\ \dot{x}(t=0) = U\cos\alpha \\ y(t=0) = 0, \quad \dot{y}(t=0) = U\sin\alpha \end{cases}$$



di cui è possibile determinare la soluzione analitica

$$x(t) = Ut\cos\alpha, \qquad y(t) = Ut\sin\alpha - \frac{1}{2}gt^2$$

Da cui è possibile determinare la distanza D percorsa dalla pallina nell'istante in cui tocca di nuovo il campo

$$\begin{cases} x(t_f) = D \\ y(t_f) = 0 \end{cases} \Rightarrow \qquad t_f = \frac{2U\sin\alpha}{g}, \qquad D = \frac{U^2\sin 2\alpha}{g} \qquad \alpha = \frac{\pi}{4}$$

La velocità iniziale è uguale a quella finale !!!!!!

La resistenza dell'aria non può essere trascurata a velocità così alte

Formulazione di un modello matematico

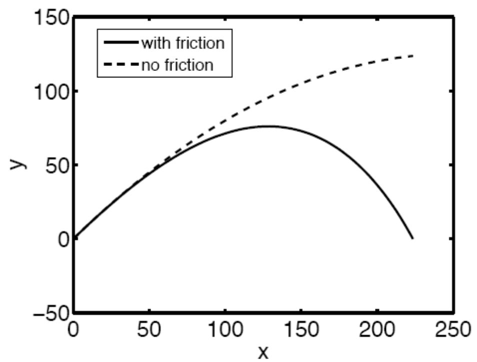
Le equazioni del moto diventano (con la resistenza dell'aria):

$$\begin{cases} M\frac{\mathrm{d}^2x}{\mathrm{d}t^2} = -c\sqrt{\dot{x}^2+\dot{y}^2}\,\dot{x} \\ M\frac{\mathrm{d}^2y}{\mathrm{d}t^2} = -Mg - c\sqrt{\dot{x}^2+\dot{y}^2}\,\dot{y} \end{cases} \qquad \begin{cases} x(t=0)=0,\ \dot{x}(t=0)=U\cos\alpha \\ y(t=0)=0,\quad \dot{y}(t=0)=U\sin\alpha \end{cases}$$

di cui non è possibile determinare la soluzione analitica

Sono necessari metodi numerici per la soluzione di equazioni differenziali

$$M = 46g, \qquad a = 2.1cm, \qquad c = 0.25,$$
$$\mathrm{U} = 70ms^{-1}, \qquad \alpha = 45°,$$
$$g = 9.8ms^{-2}, \qquad \rho = 1Kgm^{-3}$$



Senza resistenza dell'aria la palla percorrerebbe 490m prima di toccare terra

Considerando la resistenza dell'aria e usando un opportuno metodo numerico per la soluzione di equazioni differenziali ordinarie, questa distanza si riduce a 223m circa!

Problema da risolvere: calcolo dell’angolo ottimo per cui la distanza percorsa da una pallina da golf prima che tocchi terra sia massima

Modello matematico: Legge di Newton (Schema su ipotesi esemplificative) ⇒ errori inerenti

Metodo numerico: (la scelta è un arte) Metodo di Eulero, di Runge-Kutta, … ⇒ errori di troncamento

Algoritmo ⇒ stabilità

Soluzione numerica ⇒ errori di arrotondamento

La soluzione numerica è accettabile solo se si sanno stimare gli errori da cui è affetta!!!

Possibili fonti di errori

1. Errore di misura: precisione dello strumento
   (condizionamento)

2. Errore inerente: semplificazioni del modello reale
   (interpretazione del risultato)

3. Errore di troncamento: discretizzazione, iterazioni
   (dall'infinito al finito)

4. Errore di arrotondamento:
   il calcolatore "lavora in precisione finita"
   (il calcolatore non conosce i numeri reali
   ogni numero è una sequenza finita
   di numeri interi (cifre))

Errori di arrotondamento

Il sistema di numeri disponibile su un calcolatore è piuttosto primitivo: è un sistema finito di numeri di lunghezza finita, mentre l'analisi matematica o la geometria hanno a che fare con numeri infiniti di lunghezza infinita

Analisi Matematica | Analisi Numerica

Geometria, Algebra

$\downarrow$ | $\downarrow$

$\mathbb{R}$: *Numeri reali* $\rightarrow$ $\mathbb{F}$: *Numeri macchina*

$\boxed{\text{Errori di arrotondamento}}$

Errori di arrotondamento

Analisi Matematica: $\pi = 3.14159265358979323846264 33...$

$\sqrt{2} = 1.41421356237389504880168 87...$

Analisi Numerica: `>> pi` 3.14159265358979

`>> sqrt(2)` 1.414213562374

Nota: L'arrotondamento è la prima fonte di errore: i dati di input, che hanno in generale un numero infinito di cifre, vengono trasformati dal calcolatore, tramite arrotondamento, in numeri macchina, cioè numeri con un numero finito di cifre

Errori di arrotondamento: esempi

Numeri reali
3.751294196**5**...
...
3.75129419**66**...
...
3.75129419**67**...
...
3.75129419**68**...
...
3.75129419**69**...
...
3.75129419**70**...
...
3.75129419**71**...
...
3.75129419**72**...
...
3.75129419**73**...
...
3.75129419**74**...

$\underbrace{\longrightarrow}_{\text{Arrotondamento}}$

Numeri macchina
3.75129419**7**

Errori di arrotondamento

Errore di arrotondamento = **Numero reale** - **Numero macchina**

Numeri reali	*Errori di arrotondamento*
3.7512941965...	$+0.5\ldots \cdot 10^{-9}$
3.7512941966...	$+0.4\ldots \cdot 10^{-9}$
3.7512941967...	$+0.3\ldots \cdot 10^{-9}$
3.7512941968...	$+0.2\ldots \cdot 10^{-9}$
3.7512941969...	$+0.1\ldots \cdot 10^{-9}$
3.7512941970...	$+0.0\ldots \cdot 10^{-9}$
3.7512941971...	$-0.1\ldots \cdot 10^{-9}$
3.7512941972...	$-0.2\ldots \cdot 10^{-9}$
3.7512941973...	$-0.3\ldots \cdot 10^{-9}$
3.7512941974...	$-0.4\ldots \cdot 10^{-9}$

$$\Rightarrow \quad |\textit{Errore di arrotondamento}| \leq 0.5 \cdot 10^{-9}$$

Se i numeri macchina sono **arrotondati** alla $\mathbf{D}-esima$ cifra **decimale**
$\Rightarrow$ l'**errore di arrotondamento** è compreso nell'intervallo

$$\left[-0.5 \cdot 10^{-\mathbf{D}}, +0.5 \cdot 10^{-\mathbf{D}}\right]$$

Errori di arrotondamento: esempi

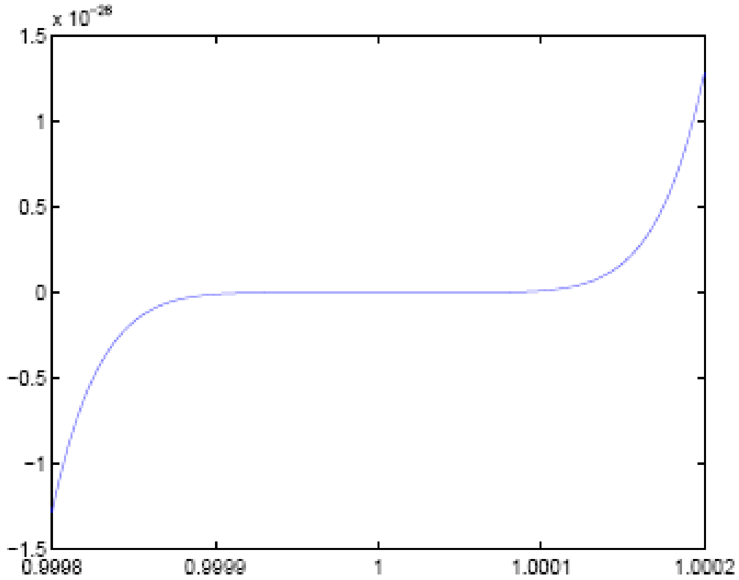
$$q_1(x) = (x-1)^7 \quad \longleftrightarrow \quad \begin{array}{l} q_2(x) = x^7 - 7x^6 + 21x^5 - \\ \quad -35x^4 + 35x^3 - 21x^2 + 7x - 1 \end{array}$$

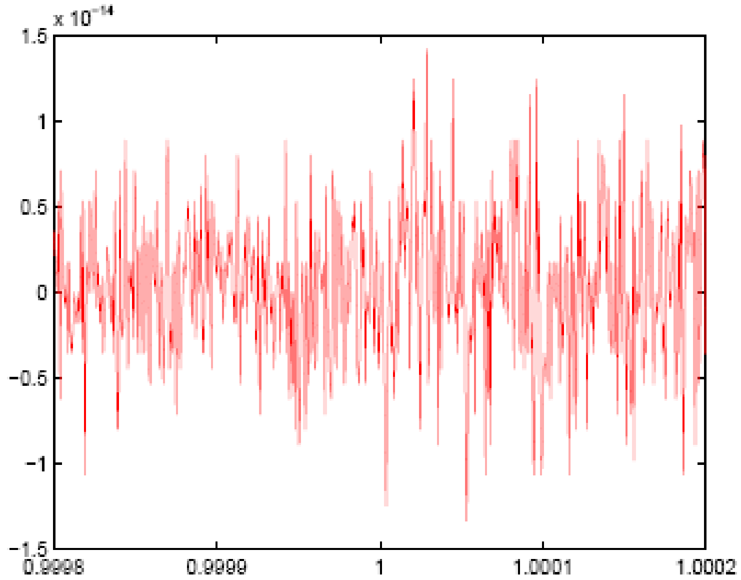
Dal punto di vista dell'**algebra** le quantità $q_1(x)$ e $q_2(x)$ sono **identiche**. Calcoliamo $q_1(x)$ e $q_2(x)$ **numericamente** nell'intervallo $[0.9998, 1.0002]$ utilizzando una **calcolatrice** che lavora con **10 cifre significative**.

x	$q_1(x)$	$q_2(x)$	Valore esatto	Errore di arrotondamento
1	0	0	0	0
1.0001	10^{-28}	-10^{-10}	10^{-28}	$\simeq -10^{-10}$
...	...	...	...	...

Esercizio

Calcolare $q_1(x)$ e $q_2(x)$ **numericamente** nell'intervallo $[0.9998, 1.0002]$ utilizzando il calcolatore.






```
figure(1); fplot('(x-1)^7',[0.9998,1.0002],'b')
figure(2); fplot('x^7-7*x^6+21*x^5-35*x^4+35*x^3-21*x^2+7*x-1',[0.9998,1.0002],'r')
```

Nota: **MATLAB** lavora con **15 cifre significative**.

Procedimento di calcolo e accuratezza del risultato

Sia

$$f(x)=\int_0^1 e^{xt}\,dt \qquad \Rightarrow \qquad \begin{cases} \dfrac{e^x-1}{x} & x\neq 0 \\ 1 & x=0 \end{cases}$$

Supp. che x=0 abbia un certo errore ⇒ $x=1.4 \cdot 10^{-9}$

$$f(x)=\frac{1.000000001-1}{1.4\cdot 10^{-9}}=\frac{0.000000001}{1.4\cdot 10^{-9}}\approx 0.714$$

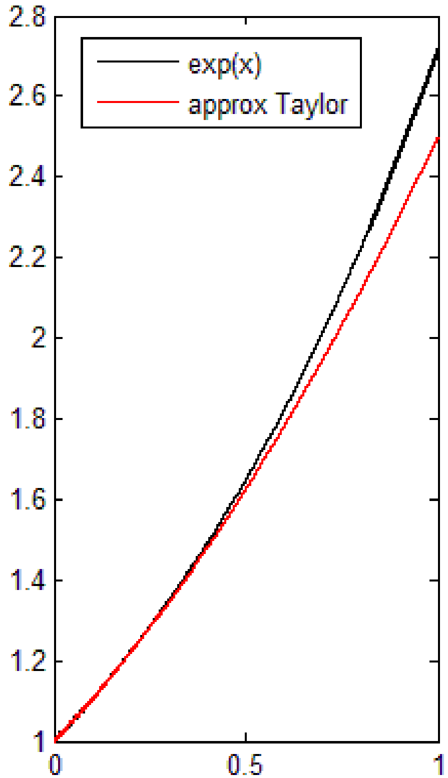
Si perde l'accuratezza della misura! |1-0.714| = 0.286

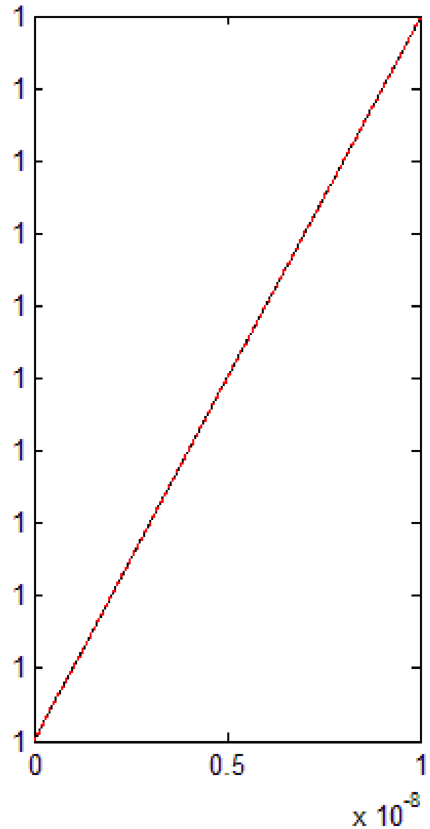
L'errore sul dato iniziale viene amplificato dal procedimento di calcolo !

Procedimento di calcolo e accuratezza del risultato

Consideriamo lo sviluppo in serie di Taylor di *f(x)* in un intorno di 0

$$e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} e^{\xi} \qquad 0 \le \xi < x \le 1$$





L'errore di approssimazione con lo sviluppo in serie è dell'ordine di 10^$^{-28}$

Procedimento di calcolo e accuratezza del risultato

$$\Rightarrow \qquad f(x)=\frac{e^x-1}{x}\approx 1+\frac{x}{2}\approx 1+\frac{1.4}{2}10^{-9}=1+0.7\cdot 10^{-9}$$

L’errore su f(x) è confrontabile con l’errore su x:

<u>Il procedimento di calcolo è fondamentale per contenere l'errore</u>

Rappresentazione dei numeri

Un numero reale x si può rappresentare come una sequenza di infinite cifre decimali (rappresentazione in base 10)

$$\sqrt{2} = 1.4142\ldots\ldots = \frac{1}{10^0} + \frac{4}{10^1} + \frac{1}{10^2} + \frac{4}{10^3} + \frac{2}{10^4} + \ldots$$

$$\pi = 3.14159\ldots = \frac{3}{10^0} + \frac{1}{10^1} + \frac{4}{10^2} + \frac{1}{10^3} + \frac{5}{10^4} + \frac{9}{10^5} + \ldots$$

Ma più in generale, scelta una base β

$$x = x_m\beta^m + x_{m-1}\beta^{m-1} + \ldots + x_1\beta^1 + x_0\beta^0 + x_{-1}\beta^{-1} + \cdots + x_{-m}\beta^{-m}$$

con $\quad 0 \le x_i \le \beta - 1$

Rappresentazione dei numeri

Ma i calcolatori hanno una memoria finita:

- è possibile usare solo una *sequenza finita di cifre*

- i numeri sono rappresentati in *virgola mobile* o *floating point*

Base 2
54.75 = 110110.11 =
$1.1011011 \cdot 2^5$

$$x = m\beta^e$$

mantissa *base* *esponente*

Base 10
265.87 = $2.6587 \cdot 10^2$
0.002658 = $2.658 \cdot 10^{-3}$

- i numeri sono in base binaria ($\beta = 2$) --- sequenza di 0 e 1 (*bits*)

$$m = 1.a_{-1}a_{-2}a_{-3}\ldots.a_{-t}$$

(*virgola mobile normalizzata*)

Rappresentazione dei numeri

Un numero in virgola mobile nella rappresentazione IEEE si scrive

$$x = \underbrace{\pm}_{\text{segno}}\underbrace{(1 + a_{-1}2^{-1} + a_{-2}2^{-2} + a_{-3}\,2^{-3} + \cdots + a_{-t}\,2^{-t})}_{\text{mantissa normalizzata}}2^{e}$$

segno → *s=1 bit*

0 se +
1 se -

mantissa normalizzata → *t bits*

esponente in [L,U] → *n bits*

La scelta di *n* e *t* determina il numero massimo rappresentabile e la sua precisione (numero di cifre decimali)

s	e	e	e	e	e	e	e	e	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a

s=1 bit *n bits* *t bits*

Rappresentazione dei numeri

Nel sistema IEEE

$$x = \pm(1 + a_{-1}2^{-1} + a_{-2}2^{-2} + a_{-3}\,2^{-3} + \cdots + a_{-t}\,2^{-t})2^{e}$$

	s	**n**	**t**	**Numero totale di bits**
Singola precisione	1	8	23	**32**
Doppia precisione	1	11	52	**64**

$\Downarrow$

s	e	e	e	e	e	e	e	e	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a

s=1 *bit* *n=8* *bits* *t=23* *bits*

Rappresentazione dei numeri

Nel sistema IEEE

$$x = \pm(1 + a_{-1}2^{-1} + a_{-2}2^{-2} + a_{-3}\, 2^{-3} + \cdots + a_{-t}\, 2^{-t})2^{e}$$

	s	***n***	***t***	**Numero totale di bits**
Singola precisione	1	8	23	**32**
Doppia precisione	1	11	52	**64**

L'esponente ***e*** può essere sia positivo che negativo

$\Downarrow$

si modifica in modo da memorizzare sempre un numero positivo sommando la quantità $b = \underbrace{0111\ldots11}_{\boxed{n \quad bits}} = 2^{n-2}\ldots+2^{2}+2^{1}+2^{0} = 2^{n-1}-1$

Rappresentazione dei numeri

Vogliamo ora stabilire quali sono i valori massimo U e minimo L dell'esponente ***e***

Allo 0 è associata la sequenza 0000..00 di *n* bits
mentre al NaN (Not a Number --- per esempio 0/0)
è associata la sequenza 111..11 di *n* bits

Quindi il *massimo* di ***e*+** *b* è *111..11 - 000..01 = 111...10*
da cui ***e* ≤ *111..10 - 011..11 = 011..11 = b = U***

Mentre il *minimo* di ***e* +** *b* *risulta > 000..00*
da cui ***e*** > *-b* ovvero ***e*** ≥ *-b +000..01*
quindi *L=-(b-1)*

Rappresentazione dei numeri

Per n= 8 ⇨ b = 127, L = -126 U=127

⇨ ***-126≤ e ≤127***

Per n= 11 ⇨ b = 1023, L = -1022 U=1023

⇨ ***-1022≤ e ≤1023***

Rappresentazione dei numeri

Alla mantissa sono riservati *t* bits e quindi la precisione è di *t+1*
(considerando la normalizzazione)

il più grande numero rappresentabile è

$$\Rightarrow \quad (1.\underbrace{111..11}_{t\ bits})\cdot 2^U = (2-2^{-t})\cdot 2^U = 2^{U+1}(1-2^{-t-1})$$

mentre il più piccolo è $(1.\underbrace{000..00}_{t\ bits})\cdot 2^L = 2^L$

Per tutti i numeri al di fuori dell'intervallo $[2^L, 2^{U+1}(1-2^{-t-1})]$ si ha *underflow* oppure *overflow*

Rappresentazione dei numeri

	***Massimo** ($2^{U+1}(1-2^{-t-1})$)*	***Minimo** (2^L)*
Singola precisione (U=127, t=23, L=-126)	$3.4 \cdot 10^{38}$	$1.2 \cdot 10^{-38}$
Doppia precisione (U=1023, t=52, L=1022)	$1.79 \cdot 10^{308}$	$2.2 \cdot 10^{-308}$

Esercizio:

Scrivere il numero 5.75 in formato IEEE in singola precisione

Esercizio: Scrivere il numero 5.75 in formato IEEE in singola precisione

1. Conversione del numero in base 2

	Quoziente	Resto
5 / 2	2	1
2 / 2	1	0
1 / 2	0	1

	Parte fraz.	n. prima della virgola
$0.75 \cdot 2 = 1.5$	0.5	1
$0.5 \cdot 2 = 1.0$	0.0	1

$5.75 = 101.11$

2. Rappresentazione in virgola mobile normalizzata
   $101.11 \longrightarrow 1.0111 \cdot 2^{2}$
3. Somma 127 all'esponente e trasformalo in base 2:
   $e = 2+127 = 129 = 10000001$
4. Scrivi mantissa: 1.01110000000000000000000
5. Determina il segno: $+ \longrightarrow 0$
6. Scrivi numero completo: $(-1)^{0} \cdot (1.01110000000000000000000) \cdot 2^{129}$

s	e	e	e	e	e	e	e	e	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a

0	1	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Rappresentazione dei numeri

Dovendo rappresentare un numero con un numero finito *p* di cifre, è necessario *arrotondare* il numero

$$x = \pm\left(\sum_{k=0}^{+\infty} x_{-k}\beta^{-k}\right)\beta^{e} \qquad x^{*} = \pm\left(\sum_{k=0}^{p-1} x_{-k}\beta^{-k}\right)\beta^{e}$$

Ci sono due possibilità:

- *Troncamento*: x^*= ***tronc(x)*** $\quad x_{-k} = 0, \quad \forall\, k \geq p$
- *Arrotondamento simmetrico:* x^*= ***tronc(x+0.5·***$\beta^{-p+1}\beta^{e}$***)***

  si aggiunge una unità alla cifra x_{-p+1} *se* $x_{-p} \geq \beta/2 = 5$

Rappresentazione dei numeri

L'errore assoluto che si commette approssimando x con x^* è

$$|x - x^*| \leq \begin{cases} \beta^{-p+1}\beta^{e} & \textit{troncamento} \\ 0.5 \cdot \beta^{-p+1}\beta^{e} & \textit{arrotondamento simmetrico} \end{cases}$$

e **l'errore relativo**

$$\frac{|x - x^*|}{x} \leq \begin{cases} \beta^{-p+1} & \textit{troncamento} \\ 0.5 \cdot \beta^{-p+1} & \textit{arrotondamento simmetrico} \end{cases}$$

Precisione di macchina

NOTA: Nel formato IEEE p-1=t

La rappresentazione dei numeri può essere molto costosa!!!

Gli errori di arrotondamento e la rappresentazione dei numeri non possono essere trascurati in quanto possono alterare in modo disastroso il risultato finale

La rappresentazione dei numeri può essere molto costosa!!!

Imperfezioni Intel Pentium (1994)

Il **Pentium FDIV bug** è stato scoperto dal prof. Thomas Nicely del Lynchburg College nell'estate del 1994, quando, calcolando la costante di Brun (1.902160583104), si accorse che il risultato ottenuto era ben lontano da quanto stimato teoricamente, anche considerando possibili errori di arrotondamento. Al contrario, il calcolo effettuato su una macchina con processore 486 risultò corretto. Il processore sbagliava a calcolare espressioni semplici quali

$$x\,(1\,/\,x)$$

in quanto i numeri erano dati con precisione fino alla quinta cifra decimale.

La Intel fu costretta a sostituire tutti i chip affrontando una spesa di circa **475 milioni di dollari**

La rappresentazione dei numeri può essere molto costosa!!!

Ariane 5

Il primo volo dell'Ariane 5 (giugno 1996) fallì e il razzo si autodistrusse dopo 40 secondi dal lancio a causa di un malfunzionamento del software di controllo.

Ci vollero 10 anni e 7 bilioni di dollari per realizzarlo.

Un dato a 64 bit in virgola mobile venne convertito in un intero a 16 bit con segno.

Questa conversione causò una operazione errata (trap) del processore: il numero in virgola mobile era troppo grande per poter essere rappresentato con un intero a 16 bit. Questo errore scatenò una reazione a catena che causò danni meccanici ai quali seguì l'azionamento del comando di autodistruzione.

Fu necessario quasi un anno e mezzo per capire quale fosse stato il malfunzionamento che aveva portato alla distruzione del razzo!

La rappresentazione dei numeri può essere molto costosa!!!

Missile Patriot

Il 25/02/1991 durante la Guerra del Golfo un missile Patriot fallì l'intercettazione di un missile Scud iracheno a causa di errori di arrotondamento.

Questo errore costò la vita a 28 soldati!

Il Patriot usava un'aritmetica a 24 bit; il tempo era memorizzato dall'orologio interno in decine di secondi e quindi diviso per 10 per ottenere i secondi.

Ma 1/10 in base 2 ha una rappresentazione periodica:
0.0001100110011001100110 0……..

La memorizzazione delle prime 24 cifre causò un errore di circa 0.000000095 secondi, che dopo cento ore introdusse un errore di 0.3433 secondi (100h=360000s fu approssimato con 359999.6567s) che causò un errore di 687 m sulla stima della posizione del missile Scud.

Cancellazione numerica

Consideriamo l'**equazione di secondo grado**

$$ax^2 + bx + c = 0$$

Dall'**algebra** sappiamo che se $\Delta = b^2 - 4ac > 0$,

l'equazione ha due **soluzioni reali e distinte**

$$x_1 = \frac{-b-\sqrt{\Delta}}{2a} \qquad x_2 = \frac{-b+\sqrt{\Delta}}{2a}$$

Calcoliamo x_1 e x_2 **numericamente** con la **calcolatrice**.

a, b, c	x_1	x_2	$ax_1^2 + bx_1 + c$	$ax_2^2 + bx_2 + c$
1 4 3	-3	-1	0	0
1 -206.5 0.01021	$4.945 \cdot 10^{-5}$	206.4999506	$-1.42 \cdot 10^{-6}$	$8.9 \cdot 10^{-6}$

Calcoliamo ora le soluzioni con le formule

$$x_1 = \frac{2c}{-b+\sqrt{\Delta}}, \; x_2 = \frac{c}{ax_1}.$$

a, b, c	x_1	x_2	$ax_1^2 + bx_1 + c$	$ax_2^2 + bx_2 + c$
1 4 3	-3	-1	0	0
1 -206.5 0.01021	$4.9443111 \cdot 10^{-5}$	206.499951	0 (*macchina*)	$9.66 \cdot 10^{-5}$

Esercizio. Ripetere il calcolo delle radici con la propria calcolatrice e con il calcolatore. Confrontare i risultati ottenuti con quelli dati nelle tabelle.

Cosa è successo? Per calcolare x_1 bisogna calcolare la quantità $-b - \sqrt{\Delta}$.

Primo caso: $a = 1, b = 4, c = 3$

$$\rightarrow \sqrt{\Delta} = 2$$

Secondo caso: $a = 1, b = -206.5, c = 0.01021$

$$\rightarrow \sqrt{\Delta} = 206.4999011...$$

In questa caso b è **negativo**, quindi bisogna calcolare la **differenza** tra due numeri molto vicini $\rightarrow$ **cancellazione numerica**.

Cancellazione numerica

Esercizi

1. Calcolare le radici dell'equazione di secondo grado ***x^2-26x+1*** lavorando prima con 3 e poi con 5 cifre significative. Discutere i risultati
2. Stabilire se per i numeri ***a=15.6*** e ***b=15.7*** vale la relazione ***$(a-b)^2=a^2-2ab+b^2$*** lavorando con 3 cifre significative

Soluzione Es. 2

a e b sono dati con 3 cifre significative

$(a-b)^2 = (-0.1)^2 = 0.01$ ✓

$a^2-2ab+b^2 = 243 - 490 + 246 =$ **-1 !!!**

Cancellazione numerica

Supponiamo di voler calcolare la somma di n numeri decimali lavorando con 4 cifre significative

$$S_n = \sum x_k$$

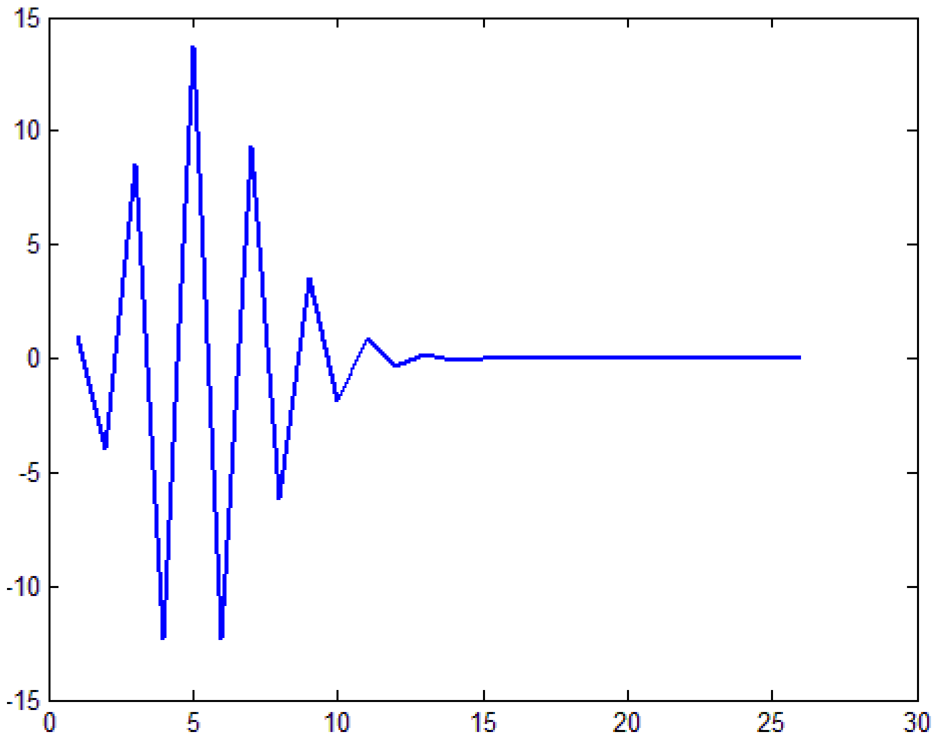
Per esempio, si vuole valutare e^x nel punto x=-5 usando n+1 termini del suo sviluppo in serie di Taylor

$$e^{-5} = 1 + \frac{(-5)}{1!} + \frac{(-5)^2}{2!} + \ldots\ldots + \frac{(-5)^n}{n!}$$

Cancellazione numerica

$$e^{-5} \approx 0.006738$$

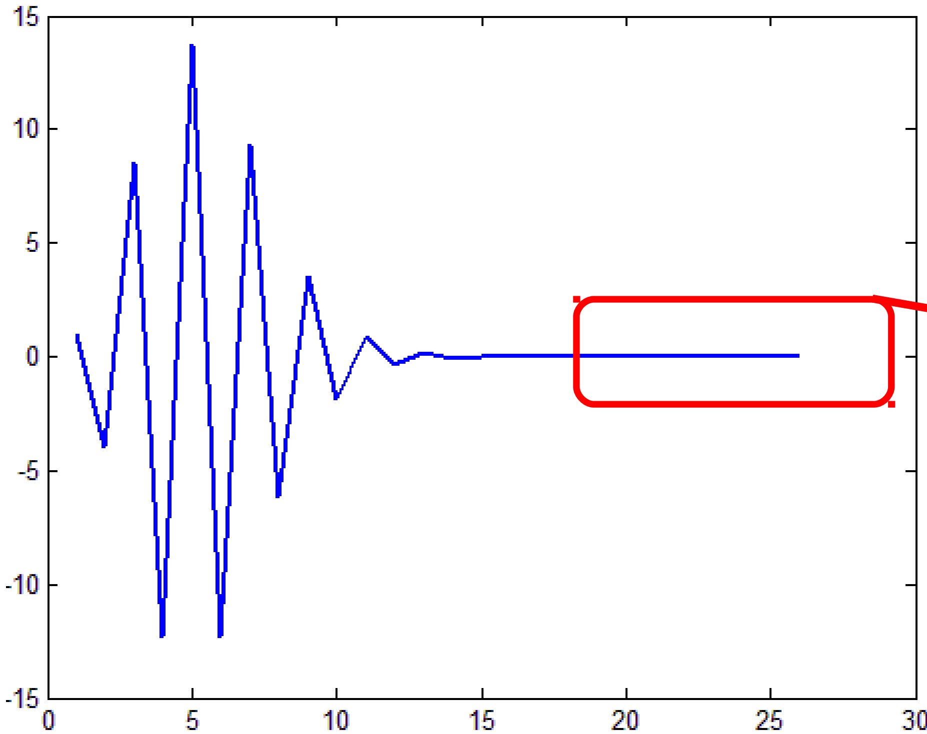
mentre usando l'espressione precedente al variare di n si ha:



grado	Termine della serie	Somma della serie
0	1.000	1.000
1	-5.000	-4.000
2	12.50	8.500
3	-20.83	-12.33
4	26.04	13.71
5	-26.04	-12.33
6	21.70	9.370
7	-15.50	-6.130
…	…	…
…	…	…

Cancellazione numerica

$e^{-5} \approx 0.006738$ mentre usando l'espressione precedente



grado	Termine della serie	Somma della serie
16	0.7293E-1	0.1166
17	-0.2145E-2	0.009518
18	0.5958E-3	0.01011
19	-0.1568E-3	**0.009932**
20	0.3920E-3	**0.009916**
21	0.9333E-5	**0.009912**
22	0.2121E-5	**0.009911**
23	0.4611E-6	**0.009911**
24	0.9607E-7	**0.009911**
25	0.1921E-7	**0.009911**

Cancellazione numerica

Calcolando, invece, prima: $\frac{1}{e} \approx 0.3679$

e poi moltiplicando lo per se stesso 5 volte si ha

$$\left(\left(\left(\left(\frac{1}{e}\,\frac{1}{e}\right)\frac{1}{e}\right)\frac{1}{e}\right)\frac{1}{e}\right) \approx 0.006736$$

che risulta prossimo a $e^{-5} \approx 0.006738$

$|0.006738-0.006736|=2\cdot 10^{-6}$

$$\frac{|0.006738-0.006736|}{0.006738} \approx 2.9 \cdot 10^{-4}$$

Propagazione degli errori di arrotondamento

Sia *fl(x)* il numero *x* rappresentato in floating point e arrotondato e

$$e_x = \frac{fl(x) - x}{x}$$ l'errore corrispondente $\Rightarrow$ $$fl(x) = xe_x + x = x(1 + e_x)$$

Sia *fl(y)* il numero *y* rappresentato in floating point e arrotondato e

$$e_y = \frac{fl(y) - y}{y}$$ l'errore corrispondente $\Rightarrow$ $$fl(y) = ye_y + y = y(1 + e_y)$$

Errore del prodotto e_{xy}:

$$fl(x)fl(y) = x(1+e_x)\ y(1+e_y) = xy\,(1+e_x+e_y+e_xe_y) \approx xy\,(1+\underbrace{e_x+e_y}_{e_{xy}})$$

Propagazione degli errori di arrotondamento

Errore della divisione $e_{x/y}$:

$$\frac{fl(x)}{fl(y)}=\frac{x(1+e_x)}{y(1+e_y)}=\frac{x}{y}(1+e_x)(1-e_y+e_y^2+\ldots)\approx\frac{x}{y}(1+\underbrace{e_x-e_y}_{e_{x/y}})$$

Errore della somma e_{x+y}:

$$fl(x)+fl(y)=$$

$$=x(1+e_x)+y(1+e_y)\approx x+y+xe_x+ye_y=(x+y)\left(1+\overbrace{\frac{x}{x+y}e_x+\frac{y}{x+y}e_y}^{e_{x+y}}\right)$$

Se $xy>0$ ***allora*** $|e_{x+y}|\leq|e_x|+|e_y|$

Se $xy<0$ ***le quantità*** $\left|\frac{x}{x+y}\right|$ e $\left|\frac{y}{x+y}\right|$ ***possono essere molto grandi***

Algoritmo

L'$\boxed{\textbf{algoritmo}}$ è una successione di $\boxed{\textbf{istruzioni}}$, **finita** e **non ambigua**, che consente di ottenere risultati numerici a partire dai dati di input.

L'algoritmo viene implementato su calcolatore tramite un $\boxed{\textbf{linguaggio di programmazione}}$.

Le **istruzioni** sono $\boxed{\textbf{operazioni logiche}}$ o $\boxed{\textbf{operazioni aritmetiche}}$ date seguendo la $\boxed{\textbf{sintassi}}$ del linguaggio di programmazione scelto.

Stabilità di un algoritmo

Anche se l'$\boxed{\text{errore di arrotondamento}}$ è **"piccolo"**, la sua **propagazione** attraverso i calcoli può avere effetti **disastrosi**. Gli errori di arrotondamento possono venire $\boxed{\textbf{amplificati}}$ durante i calcoli così da rendere la soluzione numerica del tutto **inaffidabile**. In questo caso si dice che l'$\boxed{\textbf{algoritmo}}$ è $\boxed{\textbf{instabile}}$.

Se gli errori di arrotondamento $\boxed{\textbf{non}}$ vengono $\boxed{\textbf{amplificati}}$ durante i calcoli si dice che l' $\boxed{\textbf{algoritmo}}$ è $\boxed{\textbf{stabile}}$.

Stabilità di un algoritmo

Per esempio, è accettabile un errore relativo che cresce secondo la legge lineare

$$e_n = c_0 n e_0$$

con c_0 non molto grande

Mentre l'algoritmo è instabile se la crescita dell'errore è di tipo esponenziale; per esempio

$$e_n = c_0^n e_0$$

Stabilità di un algoritmo

Dato $x^2 + 2\,px - q$, con $p^2 + q \geq 0$ eseguiamo un primo algoritmo Matlab che valuta la radice di valore maggiore:

$$y = -p + \sqrt{p^2 + q}.$$

- $p^2 + q \geq 0$ implica radici reali.
- Potenzialmente instabile per $p \gg q$ a causa della sottrazione tra p e $\sqrt{p^2 + q}$ (cancellazione).

Valutiamo la radice con un secondo algoritmo stabile:

$$\begin{aligned} y &= -p + \sqrt{p^2 + q} = \frac{(-p + \sqrt{p^2 + q})(p + \sqrt{p^2 + q})}{(p + \sqrt{p^2 + q})} \\ &= \frac{q}{(p + \sqrt{p^2 + q})} \end{aligned}$$

Stabilità di un algoritmo

p=1000; q=0.018000000081; sol=$0.9*10^{-5}$

```
[ALG.1]  [1]:  0.00000899999977272
[ALG.2]  [1]:  0.00000900000000000
[REL.ERR.][ALG.1]:  2.52e−009
[REL.ERR.][ALG.2]:  0.00e+000
```

Condizionamento di un problema

Consideriamo il **problema** del calcolo di una funzione di una variabile reale f in un generico punto $x \in \mathbb{R}$: $\boxed{y = f(x)}$.

$$x \longrightarrow \boxed{\quad f \quad} \longrightarrow y$$

Vogliamo **misurare** quale effetto produce nel calcolo di y una **perturbazione** $\Delta x = x^* - x$ del dato di input.

Sviluppo in serie di Taylor:

$$\Delta y = y^* - y = f(x^*) - f(x) = f'(x)\Delta x + \ldots\ldots$$

Errore relativo:

$$\left|\frac{\Delta y}{y}\right| \leq \left|\frac{f'(x)}{f(x)}\right| |\Delta x| = \underbrace{\left|\frac{f'(x)x}{f(x)}\right|}\left|\frac{\Delta x}{x}\right|$$

Numero di condizionamento del problema:

$$C_P := \left|\frac{f'(x)x}{f(x)}\right|$$

Se C_P è *"grande"* il problema è **malcondizionato**, cioè a **piccole perturbazioni** dei dati di input corrispondono **grandi variazioni** dei risultati. Se C_P è *"piccolo"* il problema è **ben condizionato**.

Osservazioni sul condizionamento

- Il **condizionamento non dipende** dall'algoritmo né dagli errori di arrotondamento

- Il **condizionamento dipende** dal **problema** e dai **dati di input**: uno stesso problema può essere **ben condizionato** per alcuni valori dei dati, ma **mal condizionato** per altri valori!

- Se il **problema è molto sensibile** alle variazioni dei **dati di input**, allora **nessun algoritmo**, anche se robusto e stabile, **può dare una soluzione robusta e stabile** al problema

Condizionamento: esempi

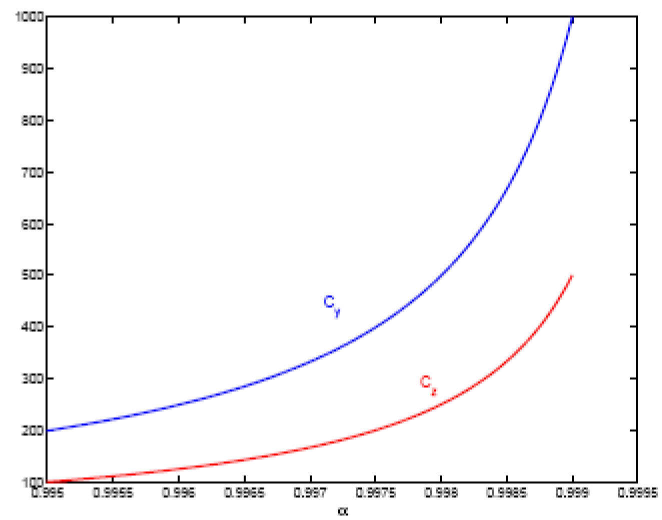
La soluzione del **sistema lineare** $\begin{cases} y + \alpha z = 1 \\ \alpha y + z = 0 \end{cases}$

è data da $y = \frac{1}{1-\alpha^2} = f(\alpha)$, $z = \frac{-\alpha}{1-\alpha^2} = g(\alpha)$.

$(\alpha^2 \neq 1)$

$$C_y = \left| \frac{f'(\alpha)\alpha}{y} \right| = \left| \frac{2\alpha^2}{1-\alpha^2} \right|$$

$$C_z = \left| \frac{g'(\alpha)\alpha}{z} \right| = \left| \frac{1+\alpha^2}{1-\alpha^2} \right|$$



$$\alpha = 0.5555 \rightarrow \begin{cases} y = 1.446299444 \\ z = -0.803419341 \end{cases} \qquad C_y = 0.89$$

$$\alpha = 0.5554 \rightarrow \begin{cases} y = 1.446067105 \\ z = -0.803145670 \end{cases}$$

$$\alpha = 0.9998 \rightarrow \begin{cases} y = 2500.250025 \\ z = -2499.749975 \end{cases}$$

$$\alpha = 0.9999 \rightarrow \begin{cases} y = 5000.250013 \\ z = -4999.749987 \end{cases} \qquad C_y = 5000$$

Riferimenti bibliografici

L. Gori *Calcolo Numerico*

Cap. 1.

Par. 1.1-1.5, 1.6

F I N E

Rappresentazione dei numeri

Conversione da base decimale a base 2: parte intera

- Dividere per 2 il numero e conservare il resto
- Ripetere il passo precedente sul quoziente fino a quando il quoziente diventa 0
- Leggere i resti dall'ultimo al primo

Esempio: convertire il numero 11 in base 2

	Quoziente	Resto
11 / 2	5	1
5 / 2	2	1
2 / 2	1	0
1 / 2	0	1

quindi, $(11)_{10} = (1011)_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$

Rappresentazione dei numeri

Conversione da base decimale a base 2: parte frazionaria

- Moltiplicare per 2 il numero e conservare il numero intero del risultato
- Ripetere il passo precedente sulla parte decimale del risultato fino a quando diventa 0
- Leggere i numeri interi conservati dal primo all'ultimo

Esempio: convertire il numero 0.25 in base 2

	Parte frazionaria	**Numero prima della virgola**
$0.25 \cdot 2 = 0.5$	0.5	0
$0.5 \cdot 2 = 1.0$	0.0	1

quindi, $(0.25)_{10} = (0.01)_2 = 0 \cdot 2^{-1} + 1 \cdot 2^{-2}$

Rappresentazione dei numeri
Esercizio

Non tutti i numeri decimali possono essere rappresentati con un numero finito di cifre in base binaria!

Consideriamo il numero 0.3

	Parte frazionaria	Numero prima della virgola
$0.3 \cdot 2 = 0.6$	0.6	0
$0.6 \cdot 2 = 1.2$	0.2	1
$0.2 \cdot 2 = 0.4$	0.4	0
$0.4 \cdot 2 = 0.8$	0.8	0
$0.8 \cdot 2 = 1.6$	0.6	1

La sequenza si ripete infinite volte

$$(0.3)_{10} \approx (0.01001)_2 \approx 0\cdot 2^{-1}+1\cdot 2^{-2}+0\cdot 2^{-3}+0\cdot 2^{-4}+1\cdot 2^{-5}=0.28125$$

Stabilità di un algoritmo: esempi

Modello matematico: $I_n = \frac{1}{e}\int_0^1 x^n e^x \, dx$

Tramite integrazione per parti si ottiene

$$I_n = \frac{1}{e}\left(e - \int_0^1 n x^{n-1} e^x \, dx\right) = 1 - nI_{n-1}$$

e continuando ...

$$\begin{aligned} I_n &= 1 - nI_{n-1} = 1 - n(1 - (n-1)I_{n-2}) = \\ &= 1 - n + n(n-1)(1 - (n-2)I_{n-3}) = \cdots = \\ &= 1 + \sum_{k=1}^{n-1} (-1)^k n(n-1)\cdots(n-k+1) + (-1)^n n! I_0 \end{aligned}$$

$\uparrow$ Algoritmo

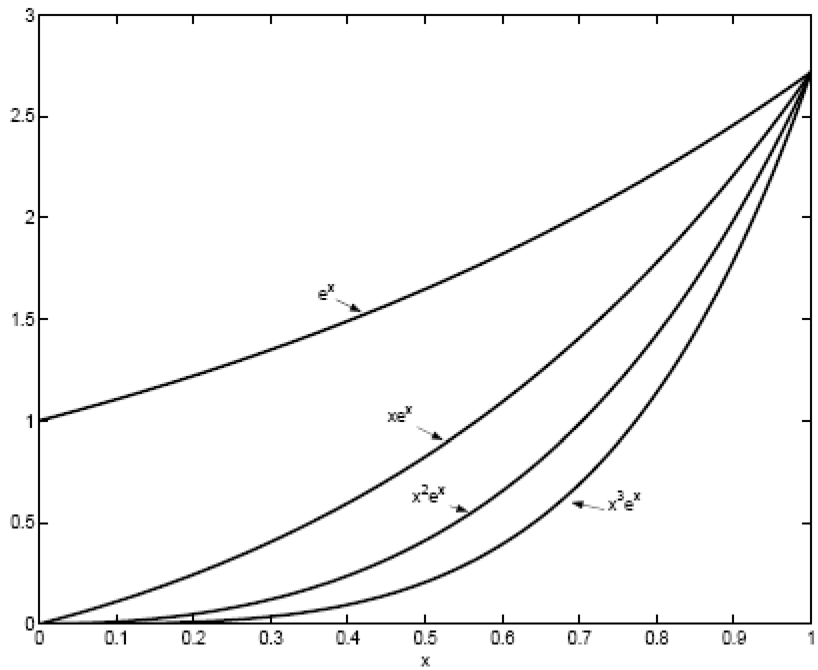
dove $I_0 = \frac{1}{e}\int_0^1 e^x \, dx = 1 - \dfrac{1}{e}$

$I_0 = 0.63212055882856 \rightarrow$ Numero macchina (14 cifre significative)

$I_1 = 1 - I_0 = 0.36787944117144$

$I_2 = 1 - 2 + 2!I_0 = -1 + 2I_0 = 0.26424111765712$

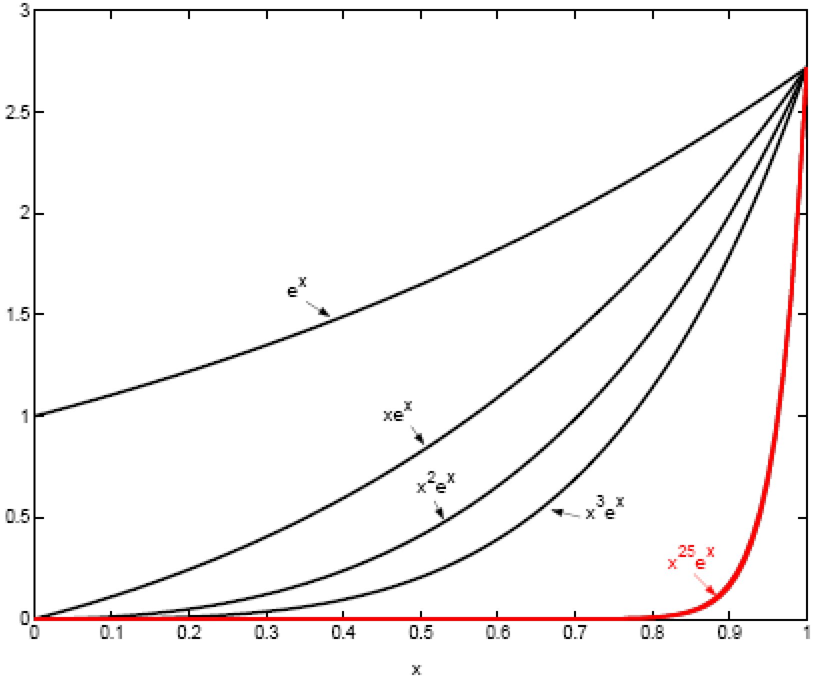
$I_3 = 1 - 3 + 3 \cdot 2 - 3!I_0 = 4 - 6I_0 = 0.20727664702865$



$$I_{25} = 0$$

$$I_{26} = -3.435973836800000e+010$$

Non è possibile!!



Algoritmo: $I_n = 1 + \sum_{k=1}^{n-1} (-1)^k n(n-1)\cdots(n-k+1) + (-1)^n n! I_0 = f(I_0)$

Nei calcoli non abbiamo usato il valore **esatto** $I_0^* = 0.63212055882856\ldots$ ma il valore **arrotondato** $I_0 = 0.63212055882856$.

Come si **propaga** nel calcolo di I_n l'**errore di arrotondamento** sul dato di input $\epsilon_0 = I_0^* - I_0$?

Errore: $\epsilon_n = I_n^* - I_n = f(I_0^*) - f(I_0) = \underbrace{(-1)^n n!}_{\text{Coeff. di amplificazione}}\epsilon_0$

$\Rightarrow$ L'**algoritmo non è stabile**

Un nuovo algoritmo

Modifichiamo l'algoritmo nel modo seguente:

$$\begin{cases} I_n = 1 - nI_{n-1} \quad \Rightarrow \quad I_{n-1} = \dfrac{1 - I_n}{n} \\ I_n \to 0 \quad \text{per} \quad n \to \infty \ \textbf{(comportamento corretto)} \end{cases}$$

Algoritmo: $I_N = 0$, $I_{k-1} = \dfrac{1 - I_k}{k}$, $k = N, N-1, \ldots$

Come si **propaga** l'**errore di arrotondamento** sul dato di input $\epsilon_N = I_N^* - I_N = I_N^*$?

$$\epsilon_{N-1} = \frac{1-I_N^*}{N} - \frac{1-I_N}{N} = -\frac{\epsilon_N}{N}$$

$$\epsilon_{N-2} = \frac{1-I_{N-1}^*}{N-1} - \frac{1-I_{N-1}}{N-1} = \frac{\epsilon_N}{N(N-1)} \qquad \cdots$$

A ogni passo l'errore iniziale viene ridotto

$\Rightarrow$ l'**algoritmo** è **stabile**

$$I_{30} = 0 \longrightarrow \; I_{25}^{(30)} = \underline{0.03708621}625288$$
$$I_{35} = 0 \longrightarrow \; I_{25}^{(35)} = \underline{0.03708621}442374$$

$$I_{30} = 0 \longrightarrow \; I_{26}^{(30)} = \underline{0.0357583}7742504$$
$$I_{35} = 0 \longrightarrow \; I_{26}^{(35)} = \underline{0.0357584}2498278$$

Nota: Si può **stimare** l'**errore di arrotondamento** sul dato di **output** tramite la differenza tra due approssimazioni successive:

$$\epsilon_{25} \simeq I_{25}^{(35)} - I_{25}^{(30)} = -1.83e - 009$$

$$\epsilon_{26} \simeq I_{26}^{(35)} - I_{26}^{(30)} = 4.76e - 008$$