Calcolo Numerico
con elementi di programmazione
(A.A. 2014-2015)

Introduzione

14 Ottobre 2014

Info

Docente: Annalisa Pascarella

Studio: Via dei Taurini,19 (IAC-CNR)

Ricevimento: su appuntamento

Homepage: http://www.iac.cnr.it/∼pasca/

Telefono: +39 (06) 49270946

Email: **a.pascarella@iac.cnr.it**

Organizzazione del corso

Orario delle lezioni: ma h. 17.30-19.00 (**aula 38**); gi h. 12.00-13.30 (**aula 17**)
Orario delle esercitazioni: me h. 8.30-11.45 (**laboratorio**, pc?)

Periodo: 30 settembre 2014 20 dicembre 2014 (**lezioni da recuperare!**)

Pagina Web del corso: http://www.iac.cnr.it/∼pasca/html/did.html
Sulla pagina Web del corso saranno disponibili gli appunti delle lezioni e il materiale delle lezioni svolte in laboratorio

Prerequisiti

Prerequisiti

Calcolo differenziale (in particolare, studio di funzioni, funzioni elementari, integrazione e derivazione di funzioni di una variabile); sistemi lineari (in particolare, matrici e loro propriet, determinanti); equazioni differenziali ordinarie.

Testo di teoria L. Gori - Calcolo Numerico, Ed. Kappa, V edizione, 2006.

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

Programma del corso

Le lezioni avranno lo scopo di illustrare vari **metodi numerici** utili in differenti campi della matematica applicata sia da un punto di vista teorico che applicativo.

Programma sintetico

- concetti di condizionamento e stabilità
- equazioni non lineari
- soluzione numerica di sistemi lineari
- approssimazione di dati e funzioni

- formule di quadratura
- soluzione numerica di equazioni differenziali
- elementi di programmazione in **MATLAB**

Esami

Prova di esame: prova scritta
Appelli: 2 appelli nella sessione invernale (gennaio-febbraio), 3 appelli di recupero (giugno, luglio, settembre)
Modalità di prenotazione: INFOSTUD Presentarsi alle prove con la ricevuta di prenotazione e un documento di riconoscimento.

Organizzazione: la prova scritta avrà lo scopo di valutare la conoscenza dei metodi numerici illustrati durante le lezioni tramite lo svolgimento di semplici esercizi di calcolo numerico e di programmazione.
Durata: 3 ore. Ci si può ritirare entro un quarto d'ora dalla fine della prova.
Risultati: i risultati saranno resi noti una settimana dopo la prova (sulla pagina WEB del corso).

Appelli

Sessione invernale: 22 gennaio 2015, 17 febbraio 2015

Sessione estiva: 8 giugno 2015, 10 luglio 2015

Sessione settembre: 17 settembre 2015

Domande?

Cosa è il **CALCOLO NUMERICO** ?

Computer graphics

per l'intrattenimento

facebook

twitter

WhatsApp

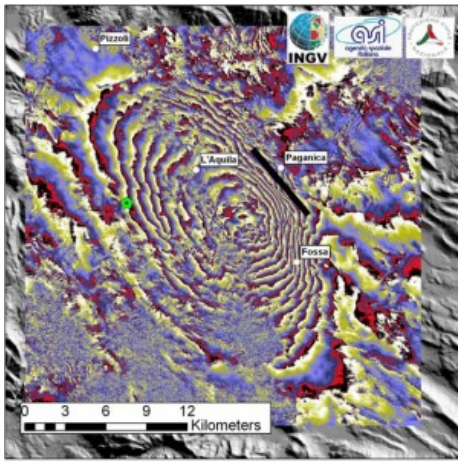
Social network

Insta

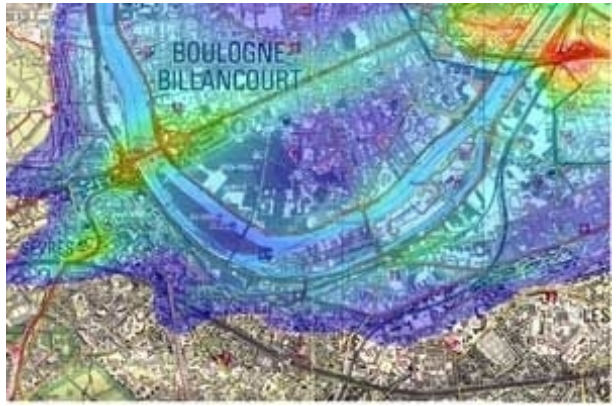
Google™

motori di ricerca

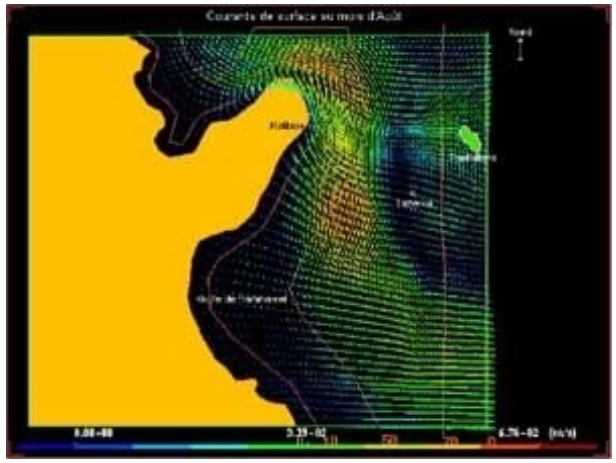
Previsioni meteo



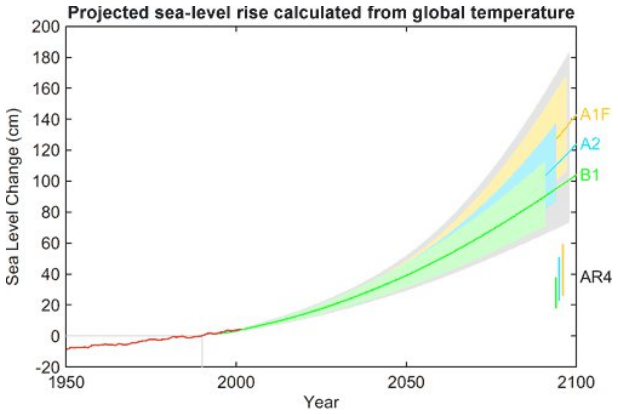
Elaborazione di dati da satellite

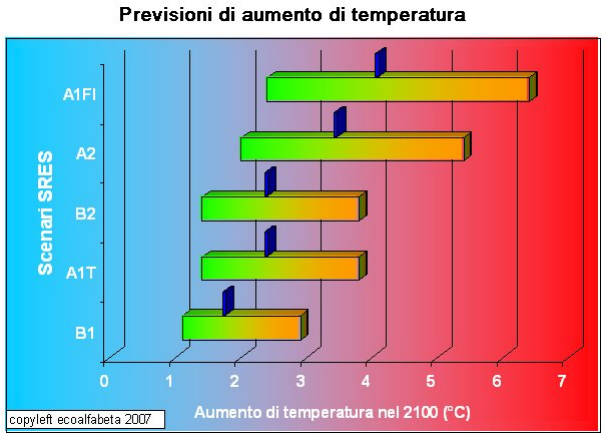


Software di simulazione:

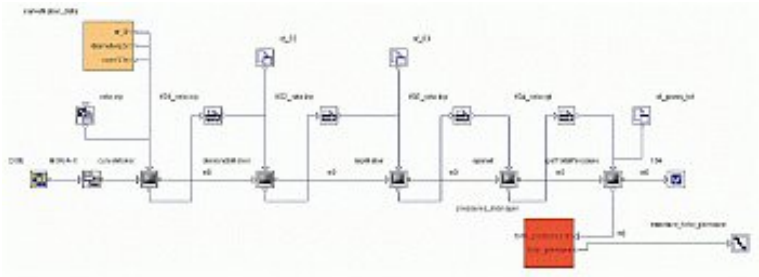


dispersione di inquinanti

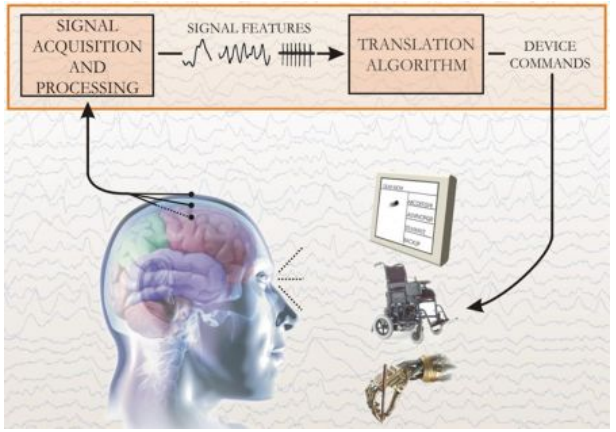




Previsione dell'innalzamento della temperatura e dell'innalzamento del livello del mare



Progettazione e gestione di reti di servizi: acquedotti, ferrovie, strade, ...



Ricostruzione di sorgenti neurali per Brain Computer Interface

Progettazione di protesi e attrezzature per disabili

Cercare sul web immagini relative a **'calcolo numerico'**

Cosa è il **CALCOLO NUMERICO** ?

È 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**, **medicina** ...

Problema da risolvere

Esempio: Calcolare la temperatura di un gas noti la **pressione**, il **volume occupato** e il **numero di moli**

↓

Schematizzazione sulla base di ipotesi esemplificative → **errori inerenti**

↓

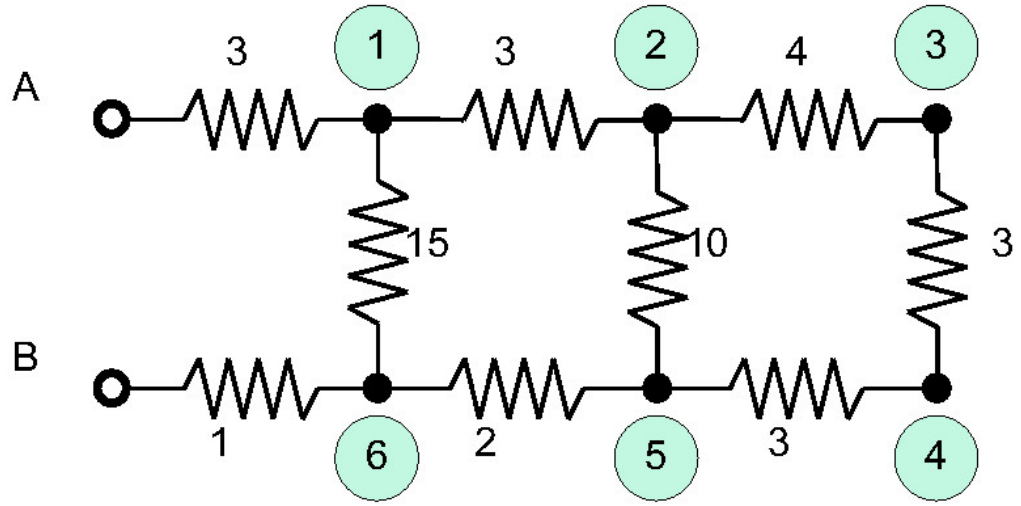
Modello matematico relazione in termini logico-matematici tra le variabili caratteristiche del problema. Esempi di modelli matematici sono i **sistemi di equazioni non lineari** (es., equilibri chimici, ottimizzazione), gli **integrali** (es., aree, volumi, energia), i **sistemi di equazioni differenziali** (es., sistemi dinamici).

Esempio: **Legge dei gas ideali**: $PV = NRT$

P: pressione, V: volume, T: temperatura R: costante

Esempio: circuito elettrico

Calcolare i potenziali v_1, v_2, ..., v_6 nei nodi del circuito



(i valori delle resistenze sono date in *Ohm*) quando tra A e B è applicata una differenza di potenziale di 100 *Volt*.

Esempio: circuito elettrico

Nodo 1: $I_{A1} + I_{21} + I_{61} = \frac{100-v_1}{3} + \frac{v_2-v_1}{3} + \frac{v_6-v_1}{15} = 0$

Applicando la Legge di Kirchoff a ciascun nodo si ottiene il **sistema lineare**:

$$\begin{cases} 11v_1 & -5v_2 & & & & -v_6 & = & 500 \\ -20v_1 & +41v_2 & -15v_3 & & -6v_5 & & = & 0 \\ & -3v_2 & +7v_3 & -4v_4 & & & = & 0 \\ & & -v_3 & +2v_4 & -v_5 & & = & 0 \\ & & & -10v_4 & +28v_5 & -15v_6 & = & 0 \\ -2v_1 & & & & -15v_5 & +47v_6 & = & 0 \end{cases}$$

↓

Metodo numerico → **errori di troncamento**

(la scelta è un *arte*)

Esempio: per i sistemi lineari
metodo di Cramer,
metodo di eliminazione di Gauss
metodo di Jacobi

↓

Algoritmo descrizione completa e non ambigua di un numero finito di operazioni logiche e aritmetiche → **stabilità**

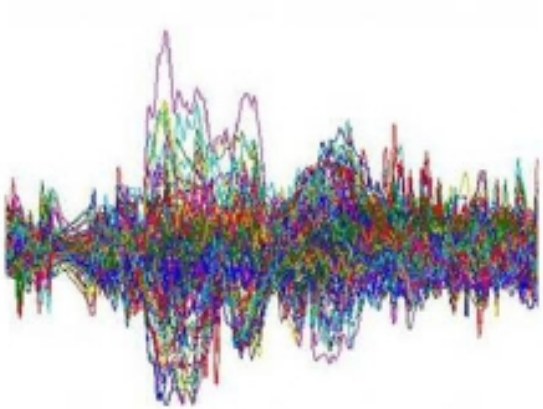
Esempio: algoritmo per il
metodo di Newton-Raphson
per la soluzione di equazioni non lineari
scelta del punto iniziale e dei criteri di arresto

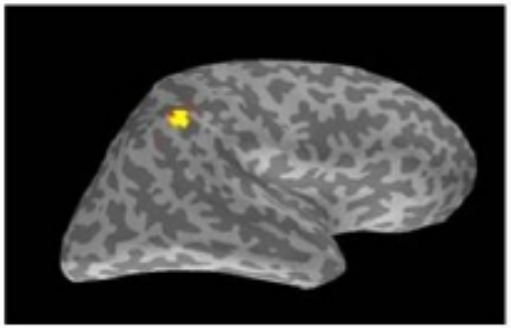
↓

Soluzione numerica → **errori di arrotondamento**

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

MEG

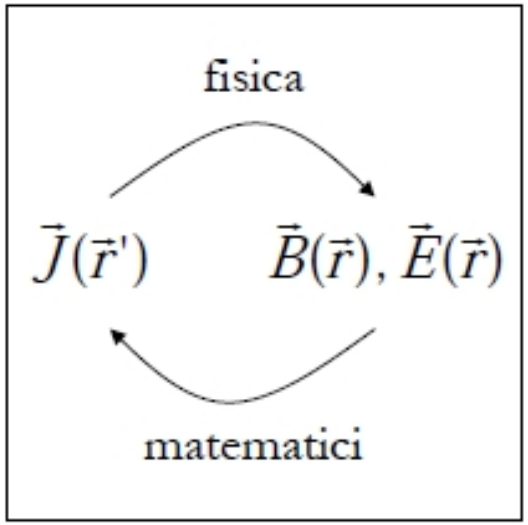


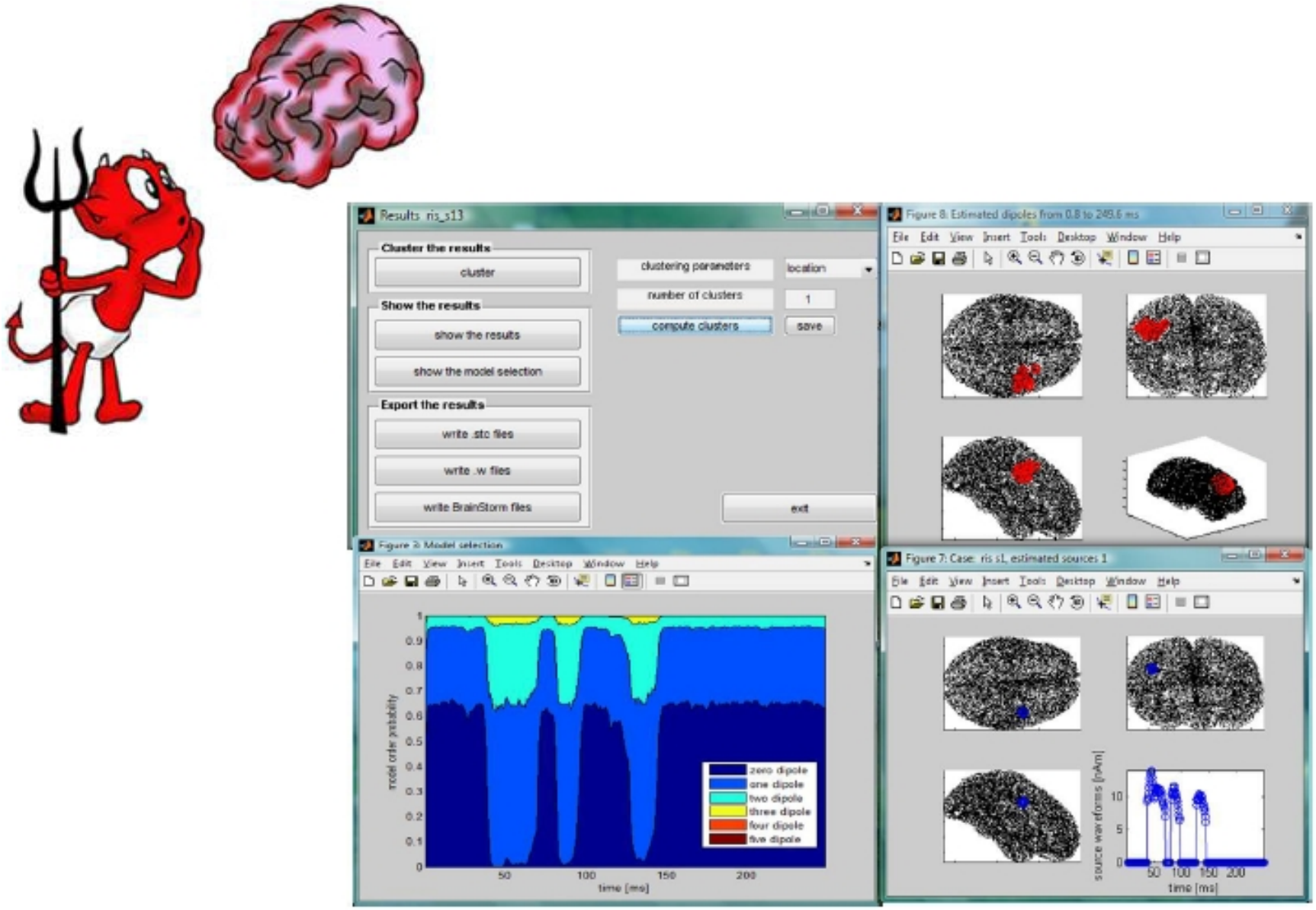


$$\mathbf{g} = \mathrm{A}\mathbf{f}$$

g -> magnetic field

f -> neural currents




Results: ris_s13
Cluster the results
cluster
Show the results
show the results
show the model selection
Export the results
write .stc files
write .w files
write BrainStorm files
clustering parameters
location
number of clusters
1
compute clusters
save
exit
Figure 8: Estimated dipoles from 0.8 to 249.6 ms
File Edit View Insert Tools Desktop Window Help
Figure 3: Model selection
model order probability
time (ms)
zero dipole
one dipole
two dipole
three dipole
four dipole
five dipole
Figure 7: Case: ris s1, estimated sources 1
source waveforms (nAm)
time (ms)

Errori di arrotondamento - 1

Il **sistema di numeri** disponibile su un **calcolatore** è un **sistema finito** di **numeri di lunghezza finita**, mentre l'**analisi matematica**, l'**algebra** o la **geometria** trattano **numeri infiniti** di **lunghezza infinita**.

Analisi Matematica
Geometria, Algebra

↓

$\mathbb{R}$: *Numeri reali*

$\rightarrow$ **Errori di arrotondamento**

Analisi Numerica

↓

F: *Numeri macchina*

Uno dei compiti dell'analisi numerica è quello di *stabilire quantitativamente l'impatto di questa approssimazione a un numero finito di cifre sull'accuratezza della soluzione approssimata* calcolata attraverso gli algoritmi.

Errori di arrotondamento - 2

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

$\sqrt{2} = 1.41421356237309504880168 87...$

Analisi Numerica:

```
atan(1)*4  →  3.141592653589 79
sqrt(2)    →  1.414213562373 10
```

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...
...
3.751294196 6...
...
3.751294196 7...
...
3.751294196 8...
...
3.751294196 9...
...
3.751294197 0...
...
3.751294197 1...
...
3.751294197 2...
...
3.751294197 3...
...
3.751294197 4...

$\overbrace{\Longrightarrow}^{\text{Arrotondamento}}$

Numeri macchina

3.751294197

Errori di arrotondamento - 3

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

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

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

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

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

Si diche che il numero macchina ha D **decimali esatti** e le **cifre** che precedono il $(D+1)$-esimo decimale esclusi gli zeri subito dopo il punto decimale sono dette **significative**.

Nell'esempio di prima il numero macchina 3.751294197 ha 9 decimali esatti e 9 cifre significative.

Errori di arrotondamento: esempi

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

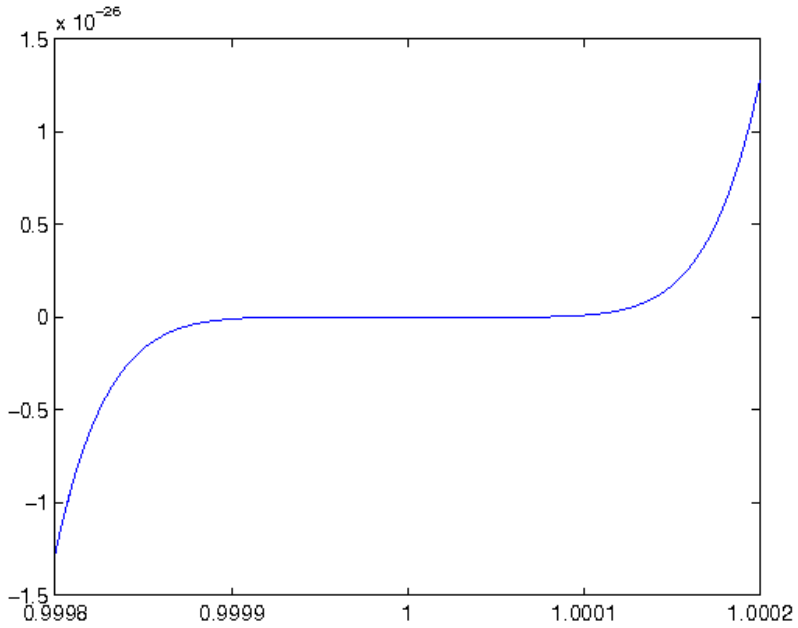
Dal punto di vista dell'**algebra** le quantità $q_1(x)$ e $q_2(x)$ sono **identiche**.

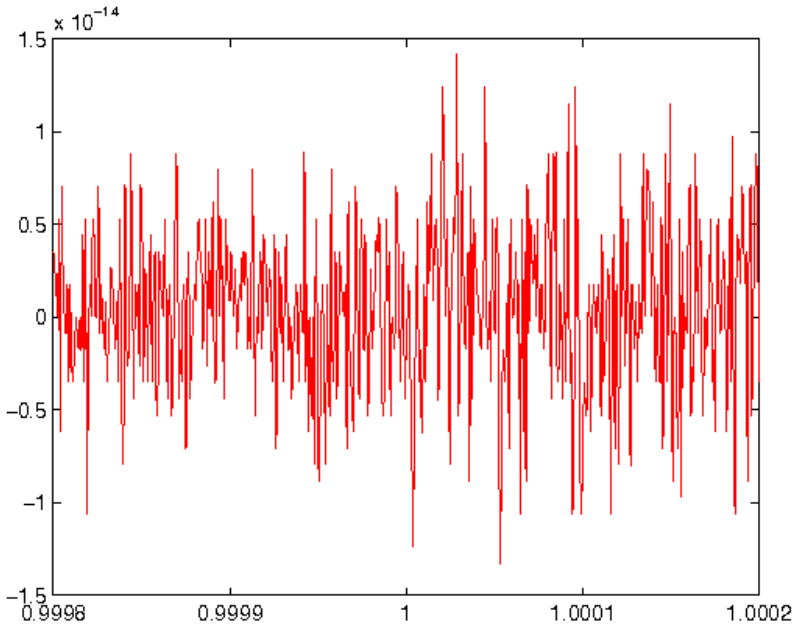
Calcoliamo $q_1(x)$ e $q_2(x)$ **numericamente** nell'intervallo $[0.9998, 1.0002]$ utilizzando una **calcolatrice** che lavora con **10 cifre significative**.

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 (Matlab)

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 sempre con **15 cifre significative**.

Rappresentazione dei numeri

Un numero reale x è rappresentato nel calcolatore come un numero macchina (**numero floating-point**)

$$fl(x) = (-1)^s \cdot \beta^e \sum_{i=1}^{t} \gamma_i \beta^{-i} = (-1)^s \cdot m \cdot \beta^e \qquad \gamma_1 \neq 0 \quad 0 \leq \gamma_i \leq \beta - 1$$

s	$= 0, 1$:	**segno**
β	(intero ≥ 2):	**base**
m	(intero di lunghezza t):	**mantissa**
e	(intero, $-N \leq e \leq M$):	**esponente**

Per rappresentare in aritmetica finita un numero reale è necessario fornire **tre numeri naturali** t, N, M e un set di t numeri naturali $\gamma_1, \ldots, \gamma_t$.

In **MATLAB**: $\beta = 2$, $t = 53$, $-1021 \leq e \leq 1024$.

Nota. 53 cifre significative in base 2 corrispondono a **15 cifre significative** in base 10.

Esempio (Matlab)

$153/7 = 21.857142857142857142857142857...$

```
>> format short      21.8571
>> format short e    2.1857e+001
>> format short g    21.857
>> format long       21.857142857142857
>> format long e     2.185714285714286e+001
>> format long g     21.8571428571429
```

Nota. Nel formato **short** vengono mostrate solo 6 cifre significative mentre nel formato **long** vengono mostrate tutte le cifre significative. I calcoli vengono comunque fatti utilizzando **tutte** le cifre significative

Underflow e overflow (Matlab)

Poiché $-1021 \leq e \leq 1024$, non si possono rappresentare numeri con valore assoluto **inferiore** a $x_{min} = \beta^{-1022}$

```
>> realmin        2.2250738585072O1e-308
```

e **superiore** a $x_{max} = \beta^{1024}(1 - \beta^{-t})$

```
>> realmax        1.7976931348623l6e+308
```

Nota. Un numero **più piccolo** di x_{min} viene trattato come 0 (**underflow**). Un numero **più grande** di x_{max} produce un messaggio di **overflow** e viene memorizzato in una variabile .

Precisione macchina

Errore relativo:

$$\frac{|x - fl(x)|}{|x|} \leq \varepsilon$$

Il numero ε è detto **precisione macchina** e dipende solo da t e β. E' il più piccolo numero macchina positivo tale che

$$fl(1 + \varepsilon) > 1$$

In **MATLAB**

$$\varepsilon = \beta^{1-t} = 2^{-52}$$

```
>> eps = 2.22044604925031e-016
```

Errore di cancellazione

Consideriamo due numeri reali x_1, x_2 e le loro relative rappresentazioni floating-point $fl(x_1), fl(x_2)$. Sia $u = x_1 + x_2$, per definizione di rappresentazione floating point si ha

$$fl(u) = fl(x_1) + fl(x_2)$$

. L'**errore relativo** sarà quindi

$$\frac{|u - fl(u)|}{|u|} \leq \frac{|x_1 - fl(x_1)| + |x_2 - fl(x_2)|}{|x_1 + x_2|} \leq \frac{\Delta x_1 + \Delta x_2}{|x_1 + x_2|}$$

Se x_1 e x_2 hanno segno opposto e sono vicini in valore assoluto l'errore relativo su u può diventare estremamente grande. Si parla in questo caso di **errore di cancellazione**.

Cancellazione numerica: esempio

Consideriamo l'**equazione di secondo grado**

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

Dall'**algebra** sappiamo che se $\Delta = b^2 - 4ac > 0$, l'equazione ha 2 **soluzioni reali** distinte:

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

Se $b >> c, \quad b > 0, \quad a = 1$ si ha

$$x_2 = \frac{-b + \sqrt{b^2 - 4c}}{2} \simeq \frac{-b + b}{2}$$

Bisogna quindi calcolare la **differenza** tra due numeri molto vicini $\rightarrow$ **cancellazione numerica**. La cancellazione può essere evitata calcolando x_2 mediante x_1

$$x_1 \cdot x_2 = c \quad \Rightarrow \quad x_2 = \frac{2c}{-b - \sqrt{\Delta}}$$

Calcoliamo x_1 e x_2 **numericamente**

a, b, c	x_1	x_2	$ax_1^2 + bx_1 + c$	$ax_2^2 + bx_2 + c$	$x_1 \cdot x_2$
1 4 3	-3	-1	0	0	3
1 206.5 0.01021	206.49995	$-4.9443111109\mathbf{3} \cdot 10^{-5}$	$-3.702 \cdot 10^{-13}$	$5.453 \cdot 10^{-13}$	0.0102099

Calcoliamo ora le soluzioni con le formule

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

a, b, c	x_1	x_2	$ax_1^2 + bx_1 + c$	$ax_2^2 + bx_2 + c$	$x_1 \cdot x_2$
1 4 3	-3	-1	0	0	3
1 206.5 0.01021	206.49995	$-4.9443111111\mathbf{9}7 \cdot 10^{-5}$	$-3.702 \cdot 10^{-13}$	$1.735 \cdot 10^{-18}$	0.01021

Cosa è successo? Per calcolare x_2 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**.

Algoritmo

L'**algoritmo** è una successione di **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 **linguaggio di programmazione**.

Le **istruzioni** sono **operazioni logiche** o **operazioni aritmetiche** date seguendo la **sintassi** del linguaggio di programmazione scelto.

Stabilità di un algoritmo

Anche se l'**errore di arrotondamento** è **"piccolo"** , la sua **propagazione** attraverso i calcoli può avere effetti **disastrosi**.

Gli errori di arrotondamento possono venire **amplificati** durante i calcoli così da rendere la soluzione numerica del tutto **inaffidabile**.

Un'altra fonte di errore dipende quindi dall'algoritmo usato per sviluppare un metodo numerico.

Si introduce il concetto di **stabilità numerica** di un algoritmo: si dice che l'**algoritmo** è **stabile** (o **instabile**) se gli errori di arrotondamento **non** sono (o sono) **amplificati** durante i calcoli.

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

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

e continuando ...

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

↑ **Algoritmo**

dove

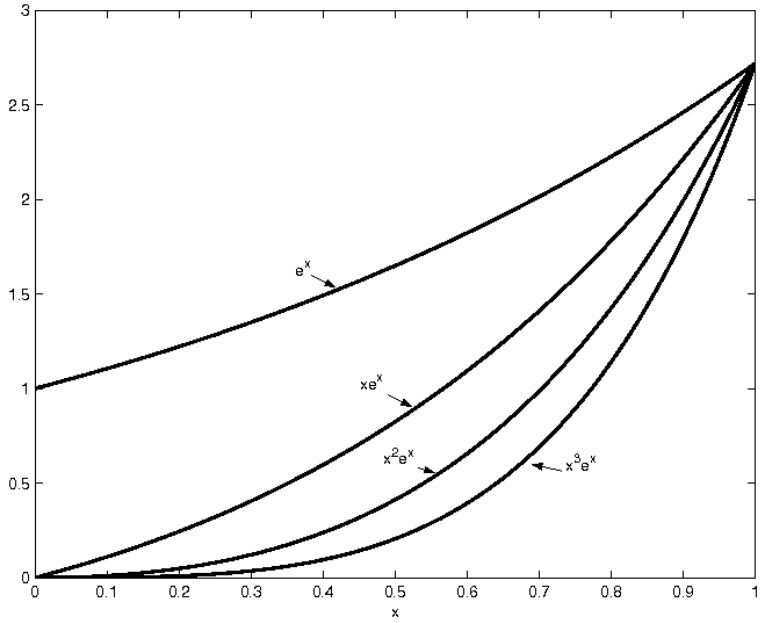
$I_0 = \frac{1}{e}\int_0^1 e^x \; dx = 1 - \frac{1}{e}$ si può assumere come dato

$I_0 = 0.632120558828558 \rightarrow$ Numero macchina (15 cifre significative)

$I_1 = 1 - I_0 = 0.36787944117144$

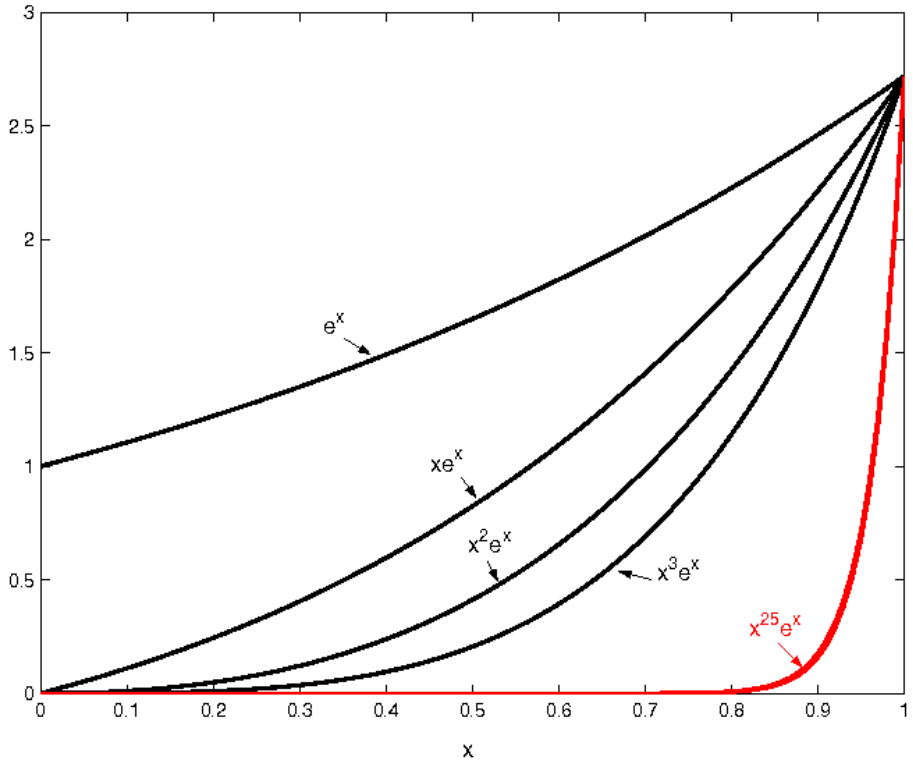
$I_2 = 1 - 2 + 2!I_0 = -2 + 2I_0 = 0.264241117657115$

$I_3 = 1 - 3 + 3 \cdot 2 - 3!I_0 = 4 - 6I_0 = 0.207276647028654$



$I_{24} = 0$
$I_{25} = 2.147483648000000e + 09$
$I_{26} = -3.435973836800000e + 10$

Non è possibile!!



L'algoritmo è **instabile**: l'errore sui dati si propaga in modo da distruggere il risultato anche per n non molto grandi.

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** $I_0^* = 0.63212055882856...$ 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: $\epsilon_n = I_n^* - I_n = f(I_0^*) - f(I_0) = \underbrace{(-1)^n n!}\epsilon_0$

L'algoritmo produce un'amplificazione di $|\epsilon_0|$ che cresce molto rapidamente con n; ad esempio se $\epsilon_0 = 10^{-10}$
$\epsilon_{10} = 10!\epsilon_0 = 3.6 \cdot 10^{-4}$
$\epsilon_{15} = -15!\epsilon_0 = -1.3 \cdot 10^{2}$

$\Rightarrow$ L'**algoritmo non è stabile** $\uparrow$ **Coeff. di amplificazione**

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

$$\epsilon_N = I_N^* - I_N = I_N^*$$

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

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

Con questo nuovo algoritmo, ad ogni passo l'errore iniziale viene ridotto

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

$$I_{30} = 0 \longrightarrow \quad I_{25}^{(30)} = \underline{0.037086216}252883$$
$$I_{35} = 0 \longrightarrow \quad I_{25}^{(35)} = \underline{0.037086214}423739$$

$$I_{30} = 0 \longrightarrow \quad I_{26}^{(30)} = \underline{0.035758377}425044$$
$$I_{35} = 0 \longrightarrow \quad I_{26}^{(35)} = \underline{0.035758424}982781$$

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

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

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

Mal posizione

Un problema è detto **mal posto** nel senso di Hadamard se la sua soluzione **non esiste**, oppure **non è unica** oppure dipende in modo **non continuo** dai dati.

Un problema in spazi a dimensione finita è tipicamente ben posto. Tuttavia, la sua soluzione numerica può essere **instabile**. Questa instabilità può essere conseguenza della scelta sbagliata del tipo di algoritmo. Oppure può essere una **instabilità intrinseca**, che permane qualunque sia l'algoritmo che si applica per la risoluzione.
Tipicamente, un problema numerico (formulato in dimensione finita) che nasce dalla discretizzazione di un problema mal posto è numericamente instabile. Questa patologia nel discreto, figlia di una patologia nel continuo, è detta **cattivo condizionamento**.

Il **numero di condizionamento** di un problema discreto misura l'instabilità numerica intrinseca e il modo con cui questo numero viene calcolato dipende dal tipo di problema.

Condizionamento di un problema

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

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

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

Sviluppo in serie di Taylor:

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

Errore relativo:

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

Numero di condizionamento del problema:

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

Osservazioni sul condizionamento

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**.

- 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.

Condizionamento: esempi

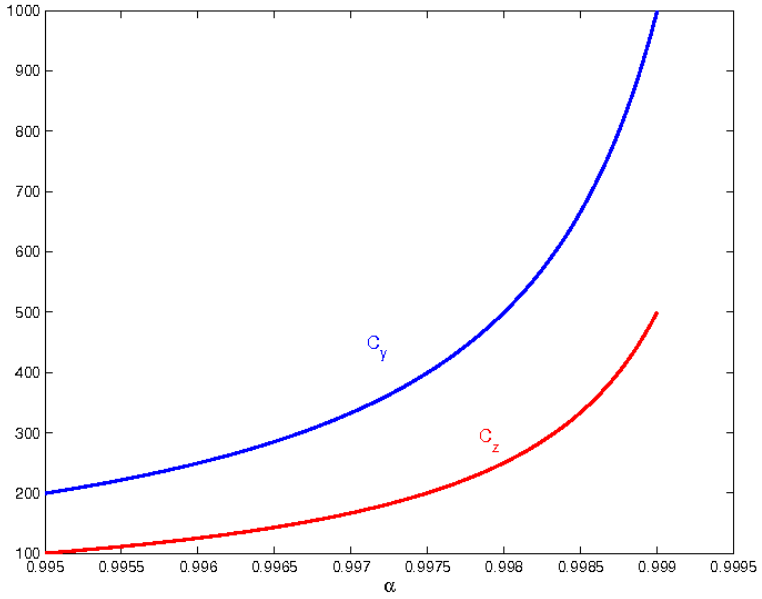
La soluzione del **sistema lineare** $\begin{cases} y + \alpha z = 1 \\ \alpha y + z = 0 \end{cases}$

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

$(\alpha^2 \neq 1)$

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

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



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

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

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

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

Esercizio.

Graficare (con gnuplot o con Matlab) C_y e C_z in funzione di α in diversi intervalli. Cosa succede se l'intervallo contiene il valore $\alpha = 1$?

Costo computazionale di un algoritmo

Prima di implementare un algoritmo bisogna stimare il suo **costo computazionale**, cioè il numero di **operazioni pesanti** (**moltiplicazioni** o **divisioni**) necessarie per calcolare **numericamente** la soluzione.

Costo computazionale:

$$\mathbf{C_c} \approx \text{numero di moltiplicazioni o divisioni}$$

Il **tempo di calcolo** è il prodotto tra il costo computazionale e il tempo con cui viene eseguita una singola operazione.

Esempio: supponendo che una singola operazione venga svolta in 10^-9 secondi, se si risolve un sistema lineare di ordine 15 con il metodo di Cramer il tempo di calcolo sarà di circa 6 ore! Per lo stesso ordine, usando il metodo di Gauss, si impiegano $1.35 \cdot 10^{-6}$ secondi.

Riferimenti bibliografici

L. Gori, *Calcolo Numerico*:
Cap. 1, Par. 1.1, 1.3 (fino errore relativo), Esempio 1.4.2, 1.5 (escluso caso bidimensionale e condizionamento del calcolo di una radice), 1.6 (concetto di stabilità ed esempio 1.6.1)

Per consultazione:
A. Quarteroni, F. Saleri, *Calcolo scientifico*, Springer, 2008