Errare non è solo umano!

prof.ssa Rossana Vermiglio, dott. Dimitri Breda
Dipartimento di Matematica e Informatica
Università degli Studi di Udine
rossana.vermiglio,dimitri.breda@dimi.uniud.it
http://www.dimi.uniud.it/rossana,dbreda

Indice

1 Appunti sulla teoria degli errori

1.1 Introduzione

L'**analisi numerica** si occupa dello sviluppo e dell'analisi di algoritmi per la risoluzione di problemi matematici che sono suggeriti dalle scienze computazionali e dall'ingegneria. Per questo motivo più recentemente è stata anche indicata con il nome **calcolo scientifico**.

In particolare tratta problemi matematici che coinvolgono variabili reali e complesse (**problemi di matematica del continuo**), ricerca delle tecniche di approssimazione che convergano rapidamente e ne studia l'accuratezza. Gli algoritmi sviluppati sono implementati su un calcolatore. Pertanto anche l'architettura dello strumento di calcolo gioca un ruolo non trascurabile nella risoluzione del problema e nell'accuratezza della risposta finale.

In molti problemi della matematica del continuo, l'approssimazione è inevitabile, perchè le quantità coinvolte non sono calcolabili dal punto di vista analitico. L'obiettivo del calcolo scientifico è pertanto quello di sviluppare ed analizzare degli algoritmi efficienti, che forniscano una soluzione ad alcuni problemi della matematica del continuo nel minimo tempo e con la massima accuratezza mediante il calcolatore.

Sono diversi gli "ingredienti" importanti nell'analisi numerica, ma il suo cuore sta negli **algoritmi**.

L'analisi teorica di uno schema di approssimazione richiede non solo lo studio della sua convergenza, ma anche della sua **complessità computazionale**. Tale parametro misura il numero delle operazioni aritmetiche richieste in funzione della dimensione del problema considerato e ci fornisce una stima del tempo di esecuzione dell'algoritmo.

Per essere eseguito l'algoritmo deve essere codificato in un programma e altri fattori devono essere considerati per misurare la qualità del software sviluppato: affidabilità, efficienza, flessibilità e altri (ulteriori dettagli nella sezione 1.8).

La strategia generale per affrontare il problema da approssimare consiste nel sostituire il problema **difficile** con uno più **semplice**, che ha la stessa soluzione del problema originale o almeno ha una soluzione "vicina" secondo lo schema:

- complicato $\rightarrow$ semplice
- nonlineare $\rightarrow$ lineare
- processo infinito $\rightarrow$ processo finito
- funzioni complicate $\rightarrow$ funzioni semplici (per esempio polinomi)
- matrici generali $\rightarrow$ matrici con semplice struttura.

In generale la soluzione ottenuta è solo un'approssimazione di quella originale e pertanto diventa essenziale **analizzare gli errori** introdotti. Il procedimento risolutivo richiede diverse fasi, ognuna delle quali introduce delle semplificazioni

e quindi degli errori. Pertanto per valutare la bontà della risposta finale (**attendibilità del risultato**), bisogna analizzare e controllare tutte le "sorgenti d'errore" o "fonti d'incertezza".

1.2 Fonti d'incertezza

Nell'elencare le fonti di incertezza e i relativi errori da analizzare, possiamo distinguere due momenti:

1. prima dell'implementazione:
   - formulazione del modello matematico (i.e. insieme di leggi di natura matematica che descrivono il fenomeno da analizzare) → **adeguatezza del modello**;
   - formulazione del modello numerico (i.e. algoritmo per approssimare il modello matematico) → **errore analitico o di troncamento**;
   - misurazioni sperimentali o computazioni precedenti → **errore nei dati di input**;
2. durante la computazione:
   - strumento di calcolo (i.e. errori di rappresentazione sul calcolatore dei numeri ed errori nelle operazioni aritmetiche) → **errori di arrontondamento**.

L'accuratezza del risultato finale risente della combinazione delle perturbazioni introdotte nei vari passi e che possono essere amplificate dalla natura del problema trattato e/o dall'algoritmo scelto.

Esempio 1 *Il calcolo della superficie della terra mediante la formula* $A = 4\pi r^2$ *coinvolge diverse approssimazioni e quindi sorgenti d'errore:*

- *la terra viene modellata come una sfera, che idealizza la sua vera forma;*
- *il valore del raggio terrestre* r *si basa su misurazioni empiriche e su altre computazioni;*
- *il valore di* π *richiede una tecnica di approssimazione ed il troncamento di un processo infinito;*
- *i valori di input e i risultati delle operazioni aritmetiche sono arrotondati sul computer;*

(vedi svolgimento Esercizio 11).

Come misurare gli errori?

1.3 Errore assoluto e relativo

Sia $\tilde{x}$ un'aprossimazione del numero reale x. Si definisce **errore assoluto**

$$e_x = |\tilde{x} - x|$$

mentre, se $x \neq 0$, si definisce **errore relativo** la quantità

$$\epsilon_x = \frac{e_x}{|x|}.$$

Una definizione equivalente di errore relativo è la seguente

$$\tilde{x} = x(1 + \delta), \quad \epsilon_x = |\delta|.$$

Nelle approssimazioni scientifiche, dove le quantità in gioco possono variare molto in grandezza, è più opportuno misurare l'errore relativo, perchè non dipende dallo "scaling". Sostituendo, infatti, x e $\tilde{x}$ rispettivamente con mx e $m\tilde{x}$, m costante moltiplicativa, l'errore relativo non varia.

Quando x e $\tilde{x}$ sono vettori, i.e. $x = (x_1, \ldots, x_n)$ e $\tilde{x} = (\tilde{x}_1, \ldots, \tilde{x}_n)$, gli errori assoluti e relativi si definiscono usando le norme vettoriali $\|\cdot\|$ come segue

$$e_x = \|\tilde{x} - x\|$$

e, se $x \neq 0$,

$$\epsilon_x = \frac{e_x}{\|x\|}.$$

In alcuni contesti può essere utile considerare l'errore relativo "componentwise", valutando $\max_{i=1,\ldots,n} \epsilon_{x_i}$.

Solitamente il valore esatto x è sconosciuto, allora si cerca una stima o una limitazione superiore dell'errore. Per la stessa ragione, l'errore relativo è spesso valutato rispetto al valore approssimato $\tilde{x}$ invece di quello esatto x.

Spesso nella valutazione di accuratezza di un risultato, si parla di **cifre significative esatte**. Vediamo attraverso alcune semplici considerazioni, come sia problematico fornire una definizione rigorosa di tale concetto abbastanza intuitivo. Le **cifre significative** di un numero x sono le prime cifre diverse da zero e tutte le successive (es. $0.0\mathbf{12302}, \mathbf{121.000567}$). La nozione di **cifre significative esatte** è legata all'errore relativo.

Consideriamo, per esempio, le seguenti coppie di numeri reali e l'errore relativo corrispondente

$$x = 1.00000, \tilde{x} = 1.00345, \epsilon_x = 3.45 \times 10^{-3};$$

$$y = 9.0000, \tilde{y} = 8.99899, \epsilon_y = 1.12 \times 10^{-4}.$$

Secondo qualsiasi ragionevole definizione di cifre significative esatte, tali numeri reali x, y concordano fino a tre cifre significative con le loro rispettive approssimazioni $\tilde{x}, \tilde{y}$. L'errore relativo è però diverso nei due casi. Possiamo così

concludere che l'errore relativo fornisce una stima più attendibile e fine dell'accuratezza dell'approssimazione rispetto all'informazione sul numero delle cifre significative esatte.

Vediamo ora alcuni esempi che hanno lo scopo di sottolineare il ruolo non trascurabile dello strumento di calcolo nell'accuratezza della risposta numerica finale.

Esempio 2 *Ci proponiamo di trovare un' approssimazione del numero π. Prendiamo a tale scopo l'algoritmo di Archimede che approssima π mediante la lunghezza del semiperimetro p_i dei poligoni regolari con 2^{i+1} lati inscritti in una circonfernza di raggio unitario, per $i = 1, 2, \ldots$.*

Dopo semplici computazioni, si ottengono le seguenti relazioni

$$\begin{aligned} l_1 &= \sqrt{2}, \\ l_{i+1} &= \sqrt{2 - \sqrt{4 - l_i^2}}, \quad i = 1, 2, \ldots, \\ p_i &= l_i \times 2^i, \quad i = 1, 2, \ldots. \end{aligned} \tag{1}$$

Implementando tale algoritmo in MATLAB con $eps = 2.22 \times 10^{-16}$, si ottengono i risultati in Tabella 1, che indicano che l'errore dopo un certo numero di iterazioni inizia a crescere, fornendo stime di π sempre meno accurate. Pur lavorando con una precisione di circa 16 cifre decimali, otteniamo che la migliore approssimazione di π ha solo 9 cifre decimali esatte ($i = 14$).

Esempio 3 *Per calcolare la funzione esponenziale $e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$, si può considerare il troncamento della serie $g_n(x) = \sum_{k=0}^{n} \frac{x^k}{k!}$ con n determinato in modo che l'ulteriore addendo non migliora l'approssimazione ottenuta. Implementando tale algoritmo in MATLAB con $eps = 2.22 \times 10^{-16}$, si ottengono i risultati in Tabella 2. È evidente l'inaccuratezza della risposta per valori $x < 0$.*

Esempio 4 *Data una funzione $f(x)$, si vuole costruire un'approssimazione della sua derivata in un punto x fissato. Poichè vale $\lim_{h\to 0} \frac{f(x+h)-f(x)}{f(x)} = f'(x)$, possiamo sperare di ottenere una buona approssimazione di $f'(x)$, considerando il rapporto incrementale $r(x, h) = \frac{f(x+h)-f(x)}{f(x)}$ per h sufficientemente piccolo. In Tabella 3 sono riportati i valori ottenuti per $f(x) = e^x$ e $x = 1$. Anche in questo caso si verifica un aumento dell'errore quando h diminuisce sotto un opportuno valore (vedi svolgimento Esercizi 12 e 21).*

Esempio 5 *Vale il seguente risultato $\lim_{x\to\infty} (1 + \frac{1}{x})^x = e$. Si vuole approssimare tale limite calcolando $f_k = (1 + \frac{1}{x_k})^{x_k}$ per $x_k = 10^k, k = 1, 2, \ldots, 20$. I risultati ottenuti con un programma MATLAB sono riportati in Tabella 4. Anche in questo caso l'errore dopo un andamento descrescente, riprende a crescere e*

i	p_i	err_{ass}	err_{rel}
1	2.828427124746190	3.13e-01	9.97e-02
2	3.061467458920719	8.01e-02	2.55e-02
3	3.121445152258053	2.01e-02	6.41e-03
4	3.136548490545941	5.04e-03	1.60e-03
5	3.140331156954739	1.26e-03	4.01e-04
6	3.141277250932757	3.15e-04	1.00e-04
7	3.141513801144145	7.88e-05	2.51e-05
8	3.141572940367883	1.97e-05	6.27e-06
9	3.141587725279961	4.92e-06	1.57e-06
10	3.141591421504635	1.23e-06	3.92e-07
11	3.141592345611077	3.07e-07	9.80e-08
12	3.141592576545004	7.70e-08	2.45e-08
13	3.141592633463248	2.01e-08	6.40e-09
14	**3.14159265**4807589	1.22e-09	3.88e-10
15	3.141592645321215	8.27e-09	2.63e-09
16	3.141592607375720	4.62e-08	1.47e-08
17	3.141592910939673	2.57e-07	8.19e-08
18	3.141594125195191	1.47e-06	4.68e-07
19	3.141596553704820	3.90e-06	1.24e-06
20	3.141596553704820	3.90e-06	1.24e-06
21	3.141674265021758	8.16e-05	2.60e-05
22	3.141829681889202	2.37e-04	7.54e-05
23	3.142451272494134	8.59e-04	2.73e-04
24	3.142451272494134	8.59e-04	2.73e-04
25	3.162277660168380	2.07e-02	6.58e-03
26	3.162277660168380	2.07e-02	6.58e-03
27	3.464101615137754	3.23e-01	1.02e-01
28	4.000000000000000	8.58e-01	2.73e-01

Tabella 1: Approssimazione di $\pi = 3.141592653589793\ldots$ con l'algoritmo di Archimede (1).

x	n	$g_n(x)$	err_{rel}
0.5	16	1.648721270700128	2.69e-16
1	19	2.718281828459046	1.63e-16
20	68	4.851651954097902e+08	1.22e-16
40	103	2.353852668370200e+17	1.35e-16
100	192	2.688117141816133e+43	9.21e-16
-0.5	16	6.065306597126333e-01	1.83e-16
-1	20	3.678794411714424e-01	3.01e-16
-20	96	5.621884472130418e-09	1.72e+00
-40	138	-3.165731894063124e+00	7.45e+17
-100	246	-2.913755646891533e+25	7.83e+68

Tabella 2: Approssimazione di e^x con il troncamento della serie.

h	err_{rel}
1.e-01	5.1709e-02
1.e-02	5.0167e-03
1.e-03	5.0017e-04
1.e-04	5.0002e-05
1.e-05	5.0000e-06
1.e-06	4.9994e-07
1.e-07	5.1484e-08
1.e-08	2.4290e-09
1.e-09	7.9257e-08
1.e-10	5.6937e-07
1.e-11	1.2005e-05
1.e-12	1.5904e-04
1.e-13	1.6770e-04
1.e-14	3.4351e-03
1.e-15	1.4360e-01
1.e-16	1.0000e+00

Tabella 3: Approssimazione di $f'(1) = e = 2.71828182845905\ldots$ con il rapporto incrementale.

l'approssimazione migliore di e ha un errore relativo pari a 1.1077×10^{-8}*. Nella Tabella 5 sono riportati i risultati ottenuti scegliendo* $x_k = 2^k$*,* $k = 1, 2, \ldots, 20$*. In questo caso si riesce a raggiungere un'approssimazione dell'ordine della precisione di macchina, ma gli ultimi valori non sono accurati (vedi svolgimento Esercizio 17).*

Esercizio 1 *Scrivere un programma MATLAB che approssima* $\lim_{x\to 0} \frac{e^x-1}{x} = 1$ *valutando* $\frac{e^x-1}{x}$ *in* $x_k = 10^{-k}$*,* $k = 1, 2, \ldots, 20$*, stampa i valori apprrossimati e l'errore con eventuale grafico.*

Per capire la causa dell'inaccuratezza dei risultati ottenuti, dobbiano analizzare meglio la rappresentazione dei numeri reali sul calcolatore e l'aritmetica di macchina.

1.4 Numeri di macchina o floating-point

Un sistema di numeri **di macchina** o **floating-point** è un insieme $\mathbb{F} \subset \mathbb{R}$ caratterizzato dai seguenti parametri

- B base di rappresentazione,
- t numero cifre della mantissa,
- $-p_{min}$ e p_{max}, $p_{min}, p_{max} > 0$, limitazioni inferiore e superiore per l'esponente.

k	f_k	err_{rel}
1	2.593742460100002	4.5815e-02
2	2.704813829421528	4.9546e-03
3	2.716923932235594	4.9954e-04
4	2.718145926824926	4.9995e-05
5	2.718268237192297	4.9999e-06
6	2.718280469095753	5.0008e-07
7	2.718281694132082	4.9416e-08
8	**2.718281**798347358	1.1077e-08
9	2.718282052011560	8.2240e-08
10	2.718282053234788	8.2690e-08
11	2.718282053357110	8.2735e-08
12	2.718523496037238	8.8905e-05
13	2.716110034086901	7.9896e-04
14	2.716110034087023	7.9896e-04
15	3.035035206549262	1.1653e-01
16	1.000000000000000	6.3212e-01
17	1.000000000000000	6.3212e-01
18	1.000000000000000	6.3212e-01
19	1.000000000000000	6.3212e-01
20	1.000000000000000	0 6.3212e-01

Tabella 4: Approssimazione di $e = 2.71828182845905\ldots$ valutando l'espressione in $x_k = 10^k$, $k = 1, 2, \ldots, 20$.

k	f_k	err_{rel}
1	2.565784513950348	5.6101e-02
2	2.697344952565099	7.7022e-03
3	2.715632000168991	9.7482e-04
4	2.717950081189666	1.2204e-04
5	2.718240351930294	1.5258e-05
6	2.718276643766046	1.9073e-06
7	2.718281180370437	2.3842e-07
8	2.718281747447938	2.9802e-08
9	2.718281818332656	3.7253e-09
10	2.718281827193247	4.6566e-10
11	2.718281828300821	5.8208e-11
12	2.718281828439267	7.2759e-12
13	2.718281828456573	9.0949e-13
14	2.718281828458736	1.1371e-13
15	2.718281828459006	1.4213e-14
16	2.718281828459040	1.7971e-15
17	2.718281828459045	1.6337e-16
18	1.000000000000000	6.3212e-01
19	1.000000000000000	6.3212e-01
20	1.000000000000000	6.3212e-01

Tabella 5: Approssimazione di $e = 2.71828182845905\ldots$ valutando l'espressione in $x_k = 2^k$, $k = 1, 2, \ldots, 20$.

$\mathbb{F}$ contiene, oltre $x = 0$, tutti i numeri reali x rappresentabili come

$$x = \pm(d_1 B^{-1} + d_2 B^{-2} + \ldots + d_t B^{-t}) B^p,$$

dove $0 \leq d_i \leq (B-1)$, $i = 1, ..., t$, $d_1 \neq 0$, $-p_{min} \leq p \leq p_{max}$.

Ricordiamo che $(d_1 d_2 ... d_t)$ è la **mantissa**, p è l'**esponente** del numero, mentre $c = p + p_{min} + 1$ è la **caratteristica**. Tale rappresentazione si dice **normalizzata**, perchè la cifra più significativa $d_1 \neq 0$, per $x \neq 0$. Nei sistemi binari, i.e. $B = 2$, risulta $d_1 = 1$ e pertanto non è necessario memorizzarla.

Le proprietà dell'insieme $\mathbb{F}$ sono:

- l'insieme $\mathbb{F}$ è finito e la sua cardinalità è data da $1 + 2(B-1)B^{t-1}(p_{max} + p_{min} + 1)$;
- il più piccolo numero positivo normalizzato di $\mathbb{F}$ è $realmin = B^{-p_{min}-1}$;
- il più grande numero positivo normalizzato di $\mathbb{F}$ è $realmax = B^{p_{max}}(1 - B^{-t})$;
- i numeri di $\mathbb{F}$ non sono uniformemente distribuiti, sono equispaziati solo tra due potenze successive di B.

L'insieme dei numeri di macchina $\mathbb{F}$ può essere esteso includendo anche i **numeri denormalizzati**, che sono i numeri reali con la cifra significativa principale $d_1 = 0$ e con l' esponente $p = -p_{min}$. Tali numeri sono distribuiti tra $B^{-p_{min}-1}$ e lo zero e sono equispaziati con spaziatura pari a $B^{-p_{min}-t}$. Tali numeri permettono un underflow graduale (vedi la sezione 1.5).

Esercizio 2 *Descrivi l'insieme dei numeri di macchina $\mathbb{F}$ caratterizati da $B = 2$, $t = 3$, $p_{min} = 2$, $p_{max} = 1$, includendo anche i numeri denormalizzati.*

1.5 Approssimazione dei numeri reali e precisione di macchina

Solitamente i calcolatori lavorano in una base $B \neq 10$ e pertanto anche un numero che è rappresentabile con un numero finito di cifre decimali, può richiedere un numero infinito di cifre in una base B diversa

Esempio 6 $x = (0.1)_{10} = (0.0001100110011...)_2$.

Esercizio 3 *Scrivere un programma MATLAB che determini la rappresentazione di un numero decimale in una base $B \neq 10$.*

Preso $x \in \mathbb{R}$ si cercherà un numero floating point $\text{fl}(x) \in \mathbb{F}$ che lo approssimi, definendo così un'applicazione $\text{fl} : \mathbb{R} \to \mathbb{F}$.

Se $x \in \mathbb{R}$ è tale che $|x| > \max\{|y|, y \in \mathbb{F}\}$, l'applicazione fl darà errore di **overflow**. Analogamente si parlerà di errore di **underflow** quando $|x| < \min\{|y|, y \in \mathbb{F}, y \neq 0\}$. In alcuni sistemi, in questo caso, viene definito $\text{fl}(x) = 0$. Se $x \in \mathbb{F}$, allora $\text{fl}(x) = x$, mentre se x non è un numero di macchina, ma è tale che $\min\{|y|, y \in \mathbb{F}, y \neq 0\} < |x| < \max\{|y|, y \in \mathbb{F}\}$, allora $\text{fl}(x) \in \mathbb{F}$ viene scelto secondo una delle seguenti strategie:

- **Troncamento**: la mantissa di x viene troncata alla t-esima cifra.

  Esempio: $B = 10$, $t = 3$: $\mathrm{fl}(0.1234567) = 0.123$; $\mathrm{fl}(0.9876543) = 0.987$, $\mathrm{fl}(0.1235) = 0.123$.

- **Arrotondamento (“rounding”)**: il numero fl(x) è il numero di macchina più vicino a x.

  Esempio: $B = 10$, $t = 3$: $\mathrm{fl}(0.1234567) = 0.123$ e $\mathrm{fl}(0.9876543) = 0.988$. Quando x è equidistante da due numeri di macchina, $\mathrm{fl}(x)$ può essere scelto in modi diversi. In particolare ricordiamo **“round away from zero”** dove $\mathrm{fl}(x)$ è il numero più grande; **“round to even”** dove $\mathrm{fl}(x)$ è il numero di macchina con la cifra d_t pari.

  Esempio: $B = 10$, $t = 3$: $\mathrm{fl}(0.1245) = 0.125$ (away from zero); $\mathrm{fl}(0.1245) = 0.124$ (to even).

Ogni numero reale x nel range di $\mathbb{F}$ sarà approssimato da $\mathrm{fl}(x)$ con un errore relativo (**errore di rappresentazione di** x) maggiorato dalla precisione di macchina u

$$\epsilon_x = \frac{|\mathrm{fl}(x) - x|}{|x|} \leq u.$$

Per la **precisione di macchina** vale $u := B^{1-t}$ nel caso di troncamento e $u := \frac{B^{1-t}}{2}$ nel caso dell’arrotondamento.

Esempio 7 *Consideriamo lo standard IEEE doppia precisione. Esso è definito dai parametri* $B = 2$, $t = 53$, $p_{min} = 1021$, $p_{max} = 1024$. *L’esponente può assumere anche i valori* $p = -p_{min} - 1 = -1022$ *e* $p = p_{max} + 1 = 1025$ *per usi speciali (rappresentazione dello zero, numeri denormalizzati, overflow e NaN). La caratteristica c soddisfa alle limitazioni* $0 \leq c \leq 2047 = 2^{11} - 1$. *Lavorando in base* $B = 2$, *solo le* 52 *cifre* d_i, $i = 2, \ldots, 53$, *indicate in seguito con* $(d_2 \ldots d_t)$, *devono essere memorizzate. Tale standard usa l’arrotondamento alla cifra pari. Il MATLAB esegue tutti i suoi calcoli secondo tale standard. La funzione logica ‘isieee’ ritorna il valore* 1 *‘vero’ se sta usando l’aritmetica IEEE e 0 ‘falso’ in caso contrario. La funzione ‘computer’ ci segnala su quale tipo di macchima il MATLAB sta lavorando.*

```
>> isieee
ans = 1
>> computer
ans = MAC2
```

La funzione ‘eps’ ci fornisce la distanza tra $x = 1.0$ *e il successivo numero floating point ed è uguale al doppio della precisione di macchina* $u = 2^{-53} \approx 1.11 \times 10^{-16}$.

```
>>eps
eps = 2.2204e-16
```

Le funzioni MATLAB ‘realmax’ e ‘realmin’ forniscono rispettivamente i numeri (normalizzati) realmax e realmin:

```
>>realmax
ans =1.7977e+308
>>realmin
ans=2.2251e-308
```

Se il risultato di una computazione è più grande di realmax, si ha overflow segnalato dall' infinito ($p = p_{max} + 1$ e mantissa $(d_2 \dots d_t)$ nulla)

```
>>realmax*2
ans = Inf
```

Se il risultato di una computazione è più piccolo di realmin si ottiene in risposta un numero denormalizzato o il valore zero se è più piccolo di eps∗realmin (i.e. il più piccolo numero denormalizzato). Un numero denormalizzato è rappresentato con la mantissa $(d_2 \dots d_t)$ diversa da zero e $p = -p_{min} - 1$.

```
>>realmin*eps
ans =4.9407e-324
>>realmin*eps/2
ans = 0
```

I risultati di operazioni non matematicamente definite producono un NaN ($p = p_{max} + 1$, mantissa $(d_2 \dots d_t)$ non nulla):

```
>>0/0
Warning: Divide by zero ans = NaN
>>Inf/Inf
ans = NaN
>>Inf-Inf
ans = NaN
>>0*NaN
ans = NaN
```

Con il comando 'format hex ', i numeri binari floating point sono rappresentati nel formato esadecimale:

```
>>format hex
>>eps
eps = 3cb0000000000000
>>realmax
ans =7fefffffffffffff
>>realmin
ans = 0010000000000000
>>2*realmax
ans =7ff0000000000000
```

1.6 Aritmetica di macchina

Le operazioni elementari (op $= +, /, *, -$) che operano su numeri di macchina non è detto diano come risultato esatto un numero di macchina. Di qui la necessità di definire delle **operazioni di macchina**, che indicheremo con fl(op).

Esempio 8 *Sia* $B = 10$, $t = 3$. *Presi* $x_1 = 0.123 \times 10^1$, $x_2 = 0.123 \times 10^{-1} = 0.00123 \times 10^1$, *in aritmetica esatta si ottiene* $x_1 + x_2 = 0.12423 \times 10^1$. *Il risultato non è un numero di macchina e deve essere pertanto approssimato: rimane così definita un'operazione approssimata,* $\mathrm{fl}(+)$, *tale che* $x_1 \mathrm{fl}(+) x_2 = 0.124 \times 10^1$. *Osserva che, in questo caso, le ultime due cifre di* x_2 *non hanno effetto sul risultato.*

Per analizzare gli errori di arrotondamento in un algoritmo, dobbiamo fare delle assunzioni sull'accuratezza nelle operazioni di base. Prendiamo come modello di **aritmetica di macchina** il seguente: tutte le operazioni aritmetiche elementari ($\mathrm{op} = +, /, *, -$) sono calcolate in modo da soddisfare per ogni $x_1, x_2 \in \mathbb{F}$

$$err_{\mathrm{op}} = \frac{|(x_1 \mathrm{fl}(\mathrm{op}) x_2) - (x_1 \mathrm{op}\ x_2)|}{|x_1 \mathrm{op}\ x_2|} \leq u, \quad \mathrm{op} = +, /, *, - \tag{2}$$

È normale considerare (2) valida anche per $\mathrm{op} = \sqrt{\cdot}$. Il modello asserisce che il valore calcolato è "buono" come il valore ottenuto dall'arrotondamento del valore esatto, cioè idealmente si assume

$$x_1 \mathrm{fl}(\mathrm{op})\ x_2 = \mathrm{fl}(x_1 \mathrm{op}\ x_2).$$

Per lo standard IEEE il modello è valido.

L'aritmetica di macchina non soddisfa tutte le proprietà dell'aritmetica esatta. NON sono valide, per esempio, la proprietà associativa della somma e del prodotto, la proprietà distributiva della somma rispetto al prodotto, la legge di cancellazione.

Esempio 9 *Sia* $B = 10, t = 3$. *Presi* $x_1 = 0.123 \times 10^2, x_2 = 0.123 \times 10^{-1} = 0.000123 \times 10^2$, *vale* $x_1 + x_2 = 0.123123 \times 10^2$, *ma* $x_1 \mathrm{fl}(+)\ x_2 = 0.123 \times 10^2$. *Osserva che* $x_2 > 0$ *non ha effetto sul risultato in aritmetica approssimata.*

Esempio 10 *Sia* $B = 10, t = 3$. *Presi* $x_1 = 0.559, x_2 = 0.555, x_3 = 0.4 \times 10^{-2}$, *si ottiene* $(x_1 \mathrm{fl}(+) x_2) \mathrm{fl}(+) x_3 = 0.111 \times 10^1 \mathrm{fl}(+)\ 0.0004 \times 10^1 = 0.111 \times 10^1$, *mentre* $x_1 \mathrm{fl}(+)(x_2 \mathrm{fl}(+) x_3) = 0.559 \mathrm{fl}(+) 0.559 = 0.112 \times 10^1$ *che è il risultato esatto arrotondato.*

L'uso dell'aritmetica di macchina comporta che **algoritmi matematicamente equivalenti non lo siano numericamente**.

Esempio 11 *Vale* $(1 \mathrm{fl}(+) x)\, \mathrm{fl}(+) x = 1$ *ma* $1 \mathrm{fl}(+)\, (x \mathrm{fl}(+) x) > 1$, *dove* x *è un numero di macchina positivo leggermente più piccolo della precisione* u.

Esercizio 4 *Verificare che non vale la proprietà associativa del prodotto nel seguente caso:* $B = 10$, $t = 5$, $x_1 = 0.80001 \times 10^1$, $x_2 = 0.12508 \times 10^1$, $x_3 = .80008 \times 10^1$.

Esercizio 5 *Verificare che non vale la proprietà distributiva del prodotto rispetto alla somma nel seguente caso:* $B = 10$, $t = 5$, $x_1 = -0.6 \times 10^1$, $x_2 = 0.60003 \times 10^1$, $x_3 = 0.2 \times 10^2$.

1.7 Analisi dell'errore

Per analizzare ed introdurre i principali concetti, prendiamo in esame il problema del calcolo del valore $y = f(x)$ dove $f : \mathbb{R} \to \mathbb{R}$ è una funzione assegnata non razionale. Osserviamo che $x \in \mathbb{R}$ rappresenta il dato di input e $y \in \mathbb{R}$ rappresenta il dato di output.

Osservazione: tutte le definizioni di errore possono essere estese al caso più generale di $y = F(x)$ con $F : \mathbb{R}^n \to \mathbb{R}^m, n, m, \geq 1$ usando le norme.

Sia g la funzione che rappresenta l'algoritmo scelto per approssimare la funzione f in aritmetica esatta e sia infine $\tilde{g}$ la funzione effettivamente calcolata in aritmetica di macchina. A causa degli errori di misurazione e di rappresentazione del numero reale x sul calcolatore, sarà usato in input un valore approssimato $\tilde{x}$. In prima approssimazione risulta

$$err_{totale} = \frac{f(x) - \tilde{g}(\tilde{x})}{f(x)} = err_{inerente} + err_{analitico} + err_{algoritmico}$$

dove

$$err_{inerente} = \frac{f(x) - f(\tilde{x})}{f(x)}, \tag{3}$$

$$err_{analitico} = \frac{f(\tilde{x}) - g(\tilde{x})}{f(\tilde{x})}, \tag{4}$$

$$err_{algoritmico} = \frac{g(\tilde{x}) - \tilde{g}(\tilde{x})}{g(\tilde{x})}. \tag{5}$$

L'accuratezza del risultato finale viene misurata dall'errore totale err_{totale} e dipende da tutte le componenti d'errore $err_{inerente}$, $err_{analitico}$, $err_{algoritmico}$.

1.7.1 Errore inerente

L'errore inerente (3) è legato al problema assegnato e alla sua sensibilità rispetto alle perturbazioni introdotte nei dati di input. La scelta dell'algoritmo non ha effetto su tale errore.

Il problema si dice **ben condizionato** se non amplifica gli errori nei dati, mentre sarà **mal condizionato** se un errore anche piccolo nei dati di ingresso darà origine ad un grande errore nella risposta.

Come "misurare" il condizionamento del problema? Sia

$$\mathrm{cond_f}(x) := \frac{|\mathrm{errore_{risposta}}|}{|\mathrm{errore_{dato}}|} = \frac{|f(x) - f(\tilde{x})||x|}{|f(x)||x - \tilde{x}|},$$

allora se $\mathrm{cond}(x) \gg 1$ il problema sarà mal condizionato. Osserviamo subito che il condizionamento dipende non solo da f ma anche dal dato x. Se supponiamo che la funzione f sia sviluppabile con Taylor, otteniamo, in prima approssimazione,

$$\mathrm{cond_f}(x) = \frac{|x||f'(x)|}{|f(x)|}.$$

Esempio 12 *Il condizionamento della funzione* $f(x) = \cos x$ *è dato da*

$$\mathrm{cond_f}(x) = |x \tan x|,$$

il problema è dunque mal condizionato per $x \approx \frac{\pi}{2}$. *Infatti* $\cos(1.57079) = 6.32679489e-06$, $\cos(1.57078) = 1.632679489e-05$, $\epsilon_x = 6.36 \times 10^{-6} \rightarrow \epsilon_{f(x)} = 1.58$.

Esercizio 6 *Stima il condizionamento del calcolo delle funzioni* $\sqrt{x}$, e^x, $\tan x$.

Consideriamo ora il problema del condizionamento del calcolo di una funzione $F : \mathbb{R}^n \rightarrow \mathbb{R}$. Usando anche in questo caso la formula di Taylor si trova, in prima approssimazione,

$$\frac{F(\tilde{x}) - F(x)}{F(x)} = \sum_{i=1}^{n} \left(\frac{x_i}{F(x)} \frac{\partial F(x)}{\partial x_i} \right) \frac{(\tilde{x}_i - x_i)}{x_i},$$

dove $x = (x_1, \ldots, x_n)$, $\tilde{x} = (\tilde{x}_1, \ldots, \tilde{x}_n) \in \mathbb{R}^n$. Passando ai moduli si ottiene

$$|err_{inerente}| \leq \sum_{i=1}^{n} |c_i(x)| \epsilon_{x_i}, \tag{6}$$

dove ϵ_{x_i} è l'errore relativo al dato i-esimo e $c_i(x) := \frac{\partial F(x)}{\partial x_i} \frac{x_i}{F(x)}$, $i = 1, \ldots, n$, è il **coefficiente di amplificazione** ad esso relativo. I coefficienti di amplificazione forniscono una misura del condizionamento del problema.

Esempio 13 *Vogliamo valutare il condizionamento della somma di* n *numeri, i.e.* $F(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i$. *I coefficienti di amplificazione sono*

$$c_i(x) = \frac{x_i}{x_1 + \ldots + x_n}, \quad i = 1, 2, \ldots, n,$$

e, applicando la formula (6), si ottiene

$$|err_{inerente}| \leq \max_{i=1,\ldots,n} (\epsilon_{x_i}) \frac{\sum_{i=1}^{n} |x_i|}{|\sum_{i=1}^{n} x_i|}. \tag{7}$$

Pertanto se $\frac{\sum_{i=1}^{n} |x_i|}{|\sum_{i=1}^{n} x_i|} >> 1$ *il problema sarà fortemente mal condizionato (***cancellazione***). Nel caso di somma di numeri* x_i *di segno concorde invece risulta* $\frac{\sum_{i=1}^{n} |x_i|}{|\sum_{i=1}^{n} x_i|} = 1$ *ed il problema è ben condizionato.*

Esempio 14 *Vogliamo valutare il condizionamento nel prodotto di n numeri, i.e.* $F(x_1, \ldots, x_n) = \prod_{i=1}^{n} x_i$. *I coefficienti di amplificazione sono*

$$c_i(x) = 1, \quad i = 1, 2, \ldots, n,$$

e pertanto il problema è sempre ben condizionato.

Esercizio 7 *Calcolare il condizionamento del quoziente di due numeri, i.e.* $F(x_1, x_2) = \frac{x_1}{x_2}$.

Esercizio 8 *Per calcolare* $y = \left(\frac{\sqrt{2}-1}{\sqrt{2}+1}\right)^3$, *si possono considerare diverse espressioni matematicamnte equivalenti:* $y = (\sqrt{2}-1)^6 = (3-2\sqrt{2})^3 = (5\sqrt{2}-7)^2 = \frac{1}{(\sqrt{2}+1)^6} = \frac{1}{99+70\sqrt{2}} = 99-70\sqrt{2}$. *Indicare quale delle formulazioni risulta essere quella con il migliore e peggiore numero di condizionamento.*

Esempio 15 Sistema lineare $Ax = b$. *Consideriamo il sistema lineare definito dai dati*

$$A = \begin{pmatrix} 0.7800 & 0.5630 \\ 0.4570 & 0.3300 \end{pmatrix}, b = \begin{pmatrix} 0.2170 \\ 0.1270 \end{pmatrix}.$$

La soluzione esatta è $x = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$. *Perturbiamo l'elemento* $a_{1,1} = 0.7800$ *della matrice* A *con* $\tilde{a}_{1,1} = 0.7810$. *L'errore relativo nel dato perturbato è* $\epsilon_{a_{1,1}} = 1.2821 \times 10^{-3}$. *La soluzione del sistema perturbato risulta* $\tilde{x} = \begin{pmatrix} 0.2483 \\ 0.0410 \end{pmatrix}$ *con un errore misurato in norma infinito pari a* $\|\tilde{x} - x\| = 1.0410$.

Esercizio 9 Zeri di polinomio. *Vogliamo studiare come variano gli zeri del polinomio* $p(x) = (x-1)(x-2)\cdots(x-20) = x^{20} - 210x^{19} + \ldots$ *perturbando il coefficiente* $a = 210$ *del monomio* x^{19} *come segue:* $\tilde{a} = 210 + 2^{-23}$ *con un errore relativo pari* $\epsilon_a = 5.677 \times 10^{-10}$ *(vedi svolgimento Esercizio 13).*

1.7.2 Errore analitico o di troncamento → $\frac{f(x)-g(x)}{f(x)}$

L'errore (4) misura la differenza tra il risultato esatto e quello fornito dall'algoritmo che usa gli stessi dati di input e opera in aritmetica esatta. Ci fornisce quindi un'informazione sulla bontà della tecnica di approssimazione (i.e. modello numerico) usata. Tale errore è dovuto, per esempio, all'approssimazione di una serie infinita con una serie troncata.

Esempio 16 *Vogliamo valutare l'errore di approssimazione in Esempio 3 di* $f(x) = e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$ *con troncamento della serie* $g_n(x) = \sum_{k=0}^{n} \frac{x^k}{k!}$. *Per* $x = 1$ *si ottiene una stima del numero di Nepero* e *con* $g_n(1)$ *con un errore* $|e - g_n(1)| = \frac{e^a}{(n+1)!} < \frac{e}{(n+1)!}$ *dove* $0 < a < 1$.

Nasce dall'interruzione di un processo iterativo prima della convergenza dopo un numero finito di passi.

Esempio 17 *L'approssimazione del valore della derivata* $f'(x)$ *con il rapporto incrementale* $r_{x,h} = \frac{f(x+h)-f(x)}{h}$ *in Esempio 4 porta all'errore analitico* $|f'(x) - r_h(x)| \leq \frac{|f''(\xi_x)h|}{2}$ *dove* $x < \xi_x < x + h$. *Per* $f(x) = e^x$ *e* $x = 1$ *si ottiene in* $x = 1$ $|e - r_h(1)| \leq \frac{e^{\xi_x} h}{2} \leq \frac{e^2 h}{2}$ *dove* $1 < \xi_x < 1 + h, 0 < h \leq 1$. *L'errore analitico tende a zero per* h *che tende a zero e pertanto non può essere il responsabile degli errori in Tabella 3 (vedi svolgimento Esercizi 12 e 21).*

1.7.3 Errore algoritmico → $\frac{g(\tilde{x}) - \tilde{g}(\tilde{x})}{g(\tilde{x})}$

L'errore algoritmico (5) nasce dalla differenza tra il risultato fornito da un algoritmo che usa l'aritmetica esatta e il risultato prodotto dallo stesso algoritmo in aritmetica di macchina. L'algoritmo scelto per la valutazione di $g(\tilde{x}), \tilde{x} \in \mathbb{F}$ può essere descritto da una sequenza finita di operazioni elementari (i.e. $+, *, /, \sqrt{\cdot}$) indicate dalle funzioni $g_i : \mathbb{R}^{n_i} \to \mathbb{R}, n_i \leq 1, i = 1, 2, \ldots, N$, i.e.

$$g = g_N \circ g_{N-1} \circ \ldots \circ g_1.$$

Se ad ogni operazione in aritmetica esatta $g_i, i = 1, \ldots, N$ viene sostituita l'operazione $\tilde{g}_i : \mathbb{F}^{n_i} \to \mathbb{F}, i = 1, \ldots, N$ in aritmetica di macchina, si ottiene la descrizione dello stesso algoritmo in aritmetica di macchina, i.e.

$$\tilde{g} = \tilde{g}_N \circ \tilde{g}_{N-1} \ldots \circ \tilde{g}_1.$$

Esempio 18 *I passi dell' algoritmo per la valutazione della funzione* $y_N = g(\tilde{x}) = \tilde{x}^N, N \geq 2$ *possono essere descritti da*

$$\begin{aligned} y_2 &= g_1(\tilde{x}, \tilde{x}) \\ y_i &= g_{i-1}(\tilde{x}, y_{i-1}), \quad i = 3, \ldots, N \end{aligned} \tag{8}$$

dove $g_i = *, i = 1, ..., N - 1$.

Il risultato $\tilde{y}_N = \tilde{g}(\tilde{x})$ *dell'algoritmo eseguito sul calcolatore sarà descritto da*

$$\begin{aligned} \tilde{y}_2 &= \tilde{g}_1(\tilde{x}, \tilde{x}) \\ \tilde{y}_i &= \tilde{g}_{i-1}(\tilde{x}, \tilde{y}_{i-1}), \quad i = 3, \ldots, N \end{aligned} \tag{9}$$

dove $\tilde{g}_i = \text{fl}(*), i = 1, ..., N - 1$

A questo errore è legato il concetto di **stabilità di un algoritmo**. Un algoritmo è **stabile** se è insensibile alle perturbazioni dovute alle approssimazioni introdotte dalle operazioni di macchina durante le computazioni, mentre si parlerà di algoritmo **instabile** in caso contrario.

In particolare si dirà che l'algoritmo è **stabile in avanti** se $|err_{algoritmico}|$ sarà un piccolo multiplo della precisione di macchina u.

Osservazione: la stabilità in avanti dell'algoritmo non garantisce l'accuratezza del risultato finale. L'accuratezza della risposta finale dipende dall' errore

totale e pertanto sia dal condizionamento del problema sia dalla stabilità dell'algoritmo. Un risultato non accurato può risultare sia dall'applicazione di un algoritmo instabile ad un problema ben condizionato sia dall'applicazione di un algoritmo stabile ad un problema mal condizionato. Le ricette per migliorare sono diverse: scegliere un algoritmo più stabile nel primo caso, riformulare matematicamente il problema nel secondo caso.

Per analizzare l'errore algoritmico abbiamo a disposizione i **grafi computazionali** nei casi semplici. Un altro approccio consiste nell'**analisi all' indietro** introdotta da J.H. Wilkinson (1919-1986). L'idea di tale tecnica sta nell' interpretare la soluzione $\tilde{g}(\tilde{x})$ fornita dall'algoritmo in aritmetica di macchina con dato $\tilde{x} \in \mathbb{F}$ come la soluzione ottenuta in aritmetica esatta con un dato perturbato, i.e.

$$\tilde{g}(\tilde{x}) = g(\tilde{x} + \epsilon).$$

L'algoritmo si dirà **stabile all'indietro** se, $\forall \tilde{x}$ il risultato sarà la soluzione in aritmetica esatta su un dato "vicino" all'originale, i.e. $\frac{|\epsilon|}{|\tilde{x}|}$ è un piccolo multiplo della precisione di macchina u.

$$\begin{array}{ccc} \tilde{x} & \rightarrow & g(\tilde{x}) \\ \text{errore} & & \text{errore} \\ \text{all}'\text{indietro} & & \text{in avanti} \\ \tilde{x} + \epsilon & \rightarrow & \tilde{g}(\tilde{x}) = g(\tilde{x} + \epsilon) \end{array}$$

Osserva che

$$|err_{algoritmico}| \approx \text{cond}_{\text{g}}(\tilde{x}) \frac{|\epsilon|}{|\tilde{x}|},$$

dove cond$(\tilde{x})$ è il numero di condizionamento relativo alla funzione g.

Esempio 19 *La somma di due numeri $x_1, x_2 \in \mathbb{F}$ con l'operazione di macchina* fl$(+)$ *può essere riscritta come segue*

$$x_1 \text{fl}(+)\ x_2 = (x_1 + x_2)(1 + e) = x_1(1 + e) + x_2(1 + e) = \tilde{x}_1 + \tilde{x}_2$$

dove $|e| \leq u$. L'algoritmo di somma risulta "stabile all'indietro".

In generale l'analisi all'indietro viene applicata al problema $y = f(x)$

$$\begin{array}{ccc} x & \longrightarrow & y = f(x) \\ \text{errore} & \searrow & \text{errore} \\ \text{all}'\text{indietro} & & \text{in avanti} \\ x + \epsilon & \longrightarrow & \tilde{y} = \tilde{g}(\tilde{x}) = f(\tilde{x} + \epsilon) \end{array} \tag{10}$$

La soluzione $\tilde{y} = \tilde{g}(\tilde{x})$ fornita dall'algoritmo in aritmetica di macchina con dato $\tilde{x} \in \mathbb{F}$ viene vista come la soluzione del problema su un dato perturbato $\tilde{g}(\tilde{x}) = f(x + \epsilon)$

Un metodo per calcolare $y = f(x)$ si dirà **stabile all'indietro** se, $\forall x$, il risultato $\tilde{y}$ sarà soluzione del problema con un dato "vicino" all'originale, i.e. $\frac{|\epsilon|}{|x|}$.

Vale

$$|\text{errorre in avanti}| \approx \text{cond}_\text{f}(x)|\text{errorre all'indietro}|,$$

dove $\text{cond}_\text{f}(x)$ è il numero di condizionamento relativo alla funzione f.

Esempio 20 *Per la funzione esponenziale* $f(x) = e^x$ *otteniamo*

$$\tilde{y} = e^{x+\epsilon} \Rightarrow x + \epsilon = log(\tilde{y})$$

Per esempio $x = 1$ *e* $n = 3$ *abbiamo*

$$\begin{array}{l} e^x = e = 2.7182818284590..., \tilde{y} = 2.66666666666667 \\ x + \epsilon = log(\tilde{y}) = 0.98082925301173 \\ errore\ in\ avanti = \tilde{y} - e^x = -0.05161516179238 \\ errore\ all'indietro = \epsilon = -0.01917074698827 \end{array} \tag{11}$$

1.7.4 Analisi dell'errore di tipo misto : in avanti e all'indietro

Diverse *routines* per calcolare delle funzioni $y = f(x)$ soddisfano una condizione più debole (**errore misto**)

$$\tilde{y} + \delta = f(\tilde{x} + \epsilon), |\delta| \leq D|y|, |\epsilon| \leq E|x|,$$

con D, E sufficientemente piccoli .

$$\begin{array}{ccc} \tilde{x} & \longrightarrow & y = f(x) \\ & \searrow & \\ & & \tilde{y} = \tilde{g}(\tilde{x}) \\ & & \updownarrow \\ x + \epsilon & \longrightarrow & \tilde{y} + \delta = f(x + \epsilon) \end{array} \tag{12}$$

In generale un algoritmo viene detto **numericamente stabile** se è stabile rispetto ad un analisi dell'errore di tipo misto.

1.7.5 Cancellazione

In base alla formula (7), la sottrazione di due numeri approssimati (a causa di errori di arrotondamento o di altri errori attribuibili a computazioni precedenti) aventi lo stesso segno e modulo quasi uguale può essere causa di inaccuratezza nel risultato. Tale fenomeno è indicato come **cancellazione**.

Esempio 21 **Radici di equazioni di secondo grado**. *Le soluzioni dell'equazione di secondo grado* $ax^2 + bx + c = 0$ *sono fornite dalla ben nota formula*

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Ma, mentre il problema matematico è banale, dal punto di vista numerico la formula richiede più attenzione in quanto può dare risposte inaccurate o essere non valutatabile in corripsondenza di particolari scelte dei dati a, b *e* c.

- *Una prima situazione numericamente "delicata" si potrà presentare quando* $b^2 \gg |4ac|$ *da cui segue che* $\sqrt{b^2 - 4ac} \approx |b|$ *ed il fenomeno della cancellazione si può presentare nel calcolo di una delle due soluzioni. Il termine* $\sqrt{b^2 - 4ac}$ *sarà affetto da errori di approssimazione e pertanto secondo la formula (7), la sottrazione amplificherà tale errore. La cancellazione in questo caso può essere evitata facilmente calcolando* $x_1 = \frac{-(b+\text{sign}(b)\sqrt{b^2-4ac})}{2a}$ *e, successivamente,* $x_2 = \frac{c}{ax_1}$.

- *Una cancellazione ben più pericolosa numericamente si può presentare nella valutazione dell'espressione* $b^2 - 4ac$. *Nel caso di radici quasi coincidenti (i.e.* $b^2 \approx 4ac$*), ci può essere una significativa perdita di accuratezza nel risultato, che non può essere risolta con una trasformazione algebrica. L'unica possibilità consiste nel valutare tale espressione con una precisione maggiore.*

- *Un'altra difficoltà numerica può essere dovuta all'overflow, che, per esempio, si può incontrare nel calcolo di* b^2 *e* $4ac$ *con* a, b *e* c *con esponente* $p = (p_{max} + 1)/2$. *In alcuni casi l'overflow si può risovere con uno "scaling" dei dati.*

Esempio 22 Deviazione standard *La media* m_n *di* n *numeri* $x_1, \ldots, x_n$, *è data da*

$$m_n = \frac{\sum_{i=1}^{n} x_i}{n}$$

e la deviazione standard σ_n *è data da*

$$\sigma_n = \frac{\sum_{i=1}^{n} (x_i - m)^2}{n-1}$$

Per calcolare la deviazione σ_n, *spesso, onde evitare due passaggi di dati, viene proposta la seguente formula matematicamente equivalente*

$$\sigma_n = \frac{\sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2}{n-1}.$$

In presenza di errori di arrotondamento, questa seconda formulazione che calcola la deviazione come differenza di due numeri positivi, può fornire una risposta non accurata per il fenomeno della cancellazione. Un algoritmo numericamente stabile per calcolare la devizione standard è il seguente: definite le quantità

$$m_k = \frac{\sum_{i=1}^{k} x_i}{k}, \quad k = 1, \ldots, n,$$

e

$$q_k = \sum_{i=1}^{k} (x_i - m_k)^2, \quad k = 1, \ldots, n,$$

valgono le relazioni ricorsive

$$m_1 = x_1, \quad m_k = m_{k-1} + \frac{x_k - m_{k-1}}{k}, \quad k = 2, \ldots, n,$$

$$q_1 = 0, \quad q_k = q_{k-1} + \frac{(x_k - m_{k-1})^2 (k-1)}{k}, \quad k = 2, \ldots, n,$$

che mi permettono di ricavare la deviazione standard da $\sigma_n = q_n/(n-1)$.

L'inacuratezza dei risultati ottenuti dall'algoritmo di Archimede in Esempio 2, dall' approssimazione dell'esponenziale per $x < 0$ in Esempio 3, dall'approssimazione della derivata con il rapporto incrementale in Esempio 4 sono tutti imputabili alla cancellazione.

Esempio 23 *L'algoritmo di Archimede può essere riscritto, per evitare la cancellazione, nel seguente modo*

$$l_1 = \sqrt{2},$$
$$l_{i+1} = \frac{l_i}{\sqrt{2+\sqrt{4-l_i^2}}}, \quad i = 1, 2, \ldots,$$
$$p_i = l_i \times 2^i, \quad i = 1, 2, \ldots.$$

I risultati ottenuti in MATLAB con $eps = 2.22 \times 10^{-16}$ *sono riportati in Tabella 6 e confermano che tale algoritmo è numericamente stabile.*

Esempio 24 *L'instabilità dell'algoritmo per il calcolo di* e^x *per* $x < 0$ *in Esempio 3 può essere risolta riscrivendo* $e^x = 1/e^{-x}$ *per* $x < 0$. *I risultati in Tabella 7 confermano la stabilità della nuova versione dell'algoritmo.*

Esempio 25 Cancellazione errori di arrotondamento *La valutazione della funzione* $f(x) = (e^x - 1)/x$ *presenta il fenomeno della cancellazione per valori di* $|x| \ll 1$. *Nella Tabella 8 si possono confrontare i risultati calcolati con i due algoritmi seguenti*

- **Algoritmo 1:**

```
if x=0 then
    f=1
else
    f=(exp(x)-1)/x
end
```

- **Algoritmo 2:**

```
y=log(x)
if y=1 then
    g=1
else
    g=(y-1)/log(y)
end
```

i	p_i	err_{ass}	err_{rel}
1	2.828427124746190	3.13e-01	9.97e-02
2	3.061467458920718	8.01e-02	2.55e-02
4	3.136548490545939	5.04e-03	1.60e-03
5	3.140331156954753	1.26e-03	4.02e-04
6	3.141277250932773	3.15e-04	1.01e-04
7	3.141513801144301	7.89e-05	2.51e-05
8	3.141572940367092	1.97e-05	6.27e-06
9	3.141587725277160	4.93e-06	1.57e-06
10	3.141591421511200	1.23e-06	3.92e-07
11	3.141592345570118	3.08e-07	9.80e-08
12	3.141592576584873	7.70e-08	2.45e-08
13	3.141592634338564	1.92e-08	6.13e-09
14	3.141592648776986	4.81e-09	1.53e-09
15	3.141592652386592	1.20e-09	3.83e-10
16	3.141592653288993	3.01e-10	9.57e-11
17	3.141592653514594	7.52e-11	2.39e-11
18	3.141592653570994	1.88e-11	5.98e-12
19	3.141592653585094	4.70e-12	1.49e-12
20	3.141592653588619	1.17e-12	3.73e-13
21	3.141592653589501	2.92e-13	9.30e-14
22	3.141592653589721	7.19e-14	2.29e-14
23	3.141592653589776	1.69e-14	5.37e-15
24	3.141592653589790	3.11e-15	9.89e-16
25	3.141592653589794	4.44e-16	1.41e-16

Tabella 6: Approssimazione di π con l'algoritmo di Archimede stabile.

x	n	$g_n(x)$	err_{rel}
-0.5	16	6.065306597126335e-01	1.83e-16
-1	20	3.678794411714423e-01	1.50e-16
-20	68	2.061153622438558e-09	2.01e-16
-40	103	-4.248354255291590e-18	1.81e-16
-100	192	3.720075976020839e-44	8.03e-16

Tabella 7: Approssimazione di e^x con il troncamento della serie stabile per $x < 0$.

L'algoritmo 1 fornisce risultati inaccurati causa la cancellazione.

L'algoritmo 2 fornisce risultati accurati anche per $x \approx 0$ perchè la divisione cancella gli errori di arrotondamento del numeratore e denominatore. Conclusione gli errori si cancellano nella divisione!!

x	f	g
1.e-05	1.000005000006965e+00	1.000005000016667e+00
1.e-06	1.000000499962183e+00	1.000000500000167e+00
1.e-07	1.000000049433680e+00	1.000000050000002e+00
1.e-08	9.999999939225288e-01	1.000000005000000e+00
1.e-09	1.000000082740371e+00	1.000000000500000e+00
1.e-10	1.000000082740371e+00	1.000000000050000e+00
1.e-11	1.000000082740371e+00	1.000000000005000e+00
1.e-12	1.000088900582341e+00	1.000000000000500e+00
1.e-13	9.992007221626408e-01	1.000000000000050e+00
1.e-14	9.992007221626408e-01	1.000000000000005e+00
1.e-15	1.110223024625157e+00	1.000000000000000e+00
1.e-16	0	1

Tabella 8: Confronto tra gli algoritmi 1 e 2 per il calcolo di $f(x) = (e^x - 1)/x$.

Per l'approssimazione della derivata si devono determinare degli approcci alternativi e c'è un capitolo dell'analisi numerica chiamato "derivazione numerica" che si occupa di tale problema. Tra i vari approcci possibili ricordiamo la tecnica di **Estrapolazione di Richardson**.

Esercizio 10 Il computer sa contare fino a sei? *Commenta i seguenti risultati forniti dal MATLAB.*

```
>>2-1
ans =
    1
>>(1/cos(100*pi+pi/4))^2
ans =
    2.00000000000011
>>3*cos(acos(10000))/10000
ans =
    2.99999997414701
>>s=4;
  for i=1:53
    s=sqrt(s);
  end
  for i=1:53
    s=s^2;
  end
>>s
s =
    1
```

```
>>5*((1+exp(-100))-1)/((1+exp(-100))-1)
ans =
    NaN
>>log(exp(6000))/1000
ans =
    Inf
```

1.7.6 Somma di n numeri

Vogliamo analizzare diversi approcci algoritmici per calcolare la somma s_n di n numeri $x_i \in \mathbb{F}, i = 1, \ldots, n$. Poichè sappiamo che non vale la proprietà associativa per la somma in aritmetica di macchina e che l'errore di arrotondamento di un singolo passo dipende dagli addendi da sommare, il valore finale effettivemente calcolato dall'algoritmo della **somma ricorsiva** dipenderà dall'ordinamento dei dati. Analizzando l'errore nell'algoritmo

```
s(1)=x(1)
for i=2:n
    s(i)=s(i-1)+x(i);
end
```

si ottiene in prima approssimazione

$$err_{algoritmo} = \frac{1}{s_n} \sum_{i=2}^{n} s_i \delta_{i-1}$$

dove δ_i è l'errore nell'i-esima operazione di somma. Da $|\delta_i| \leq u, i = 1, \ldots, n-1$, si ottiene

$$|err_{algoritmo}| \leq \frac{u}{|s_n|} \sum_{i=2}^{n} |s_i|, \tag{13}$$

e, dalla maggiorazione $|s_i| \leq \sum_{i=1}^{n} |x_i|, i = 2, \ldots, n$, si ottiene

$$|err_{algoritmo}| \leq (n-1)u \frac{\sum_{i=1}^{n} |x_i|}{|s_n|} \tag{14}$$

In particolare, se i dati x_i, $i = 1, \ldots, n$, hanno lo stesso segno, (14) fornisce la limitazione

$$|err_{algoritmo}| \leq (n-1)u.$$

Da (13) possimo dedurre che l'accuratezza migliore si otterrà scegliendo l'ordinamento che minimizza $|s_i|, i = 2, \ldots, n$. Questo problema di ottimizzazione combinatoria è troppo costoso da risolvere in generale. Nel caso particolare di x_i, $i = 1, \ldots, n$, non negativi, la soluzione ottimale è data dall'ordinamento crescente che fornisce la stima

$$|err_{algoritmo}| \leq \frac{(n+1)u}{2}.$$

In generale si possono proporre soluzioni di compromesso quali l'ordinamento per modulo crescente oppure l'ordinamento determinato sequenzialmente minimizzando passo dopo passo $|s_1| = |x_1|, |s_2|, \ldots$ (**algoritmo PSUM**). In generale, l'ordinamento decrescente non è consigliabile nella somma di numeri positivi, perchè la limitazione in (13) è potenzialmente più grande e perchè in una somma con termini che variano molto in ordine di grandezza non permette ai termini più piccoli di contribuire al risultato. Ciò nonostante l'ordine decrescente può fornire una risposta più accurata sia di Psum che dell'ordinamento crescente in presenza di forte cancellazione, i.e. $|s_n| \ll \sum_{i=1}^{n} |x_i|$, come questo esempio dimostra.

Esempio 26 *Sia* $M \in \mathbb{F}$ *tale che* $\mathrm{fl}(1 + M) = M$ *e supponiamo di dover sommare* 1, M, $2M$ *e* $-3M$. *I risultati che si ottengono sono*

- *ordinamento crescente:* $\mathrm{fl}(1 + M + 2M - 3M) = 0$*;*
- *ordinamento PSUM:* $\mathrm{fl}(1 + M - 3M + 2M) = 0$*;*
- *ordinamento decrescente:* $\mathrm{fl}(-3M + 2M + M + 1) = 1$.

Ricordiamo infine **l'algoritmo dell'inserzione**: i dati x_i vengono ordinati secondo il modulo crescente ed ogni somma parziale s_i calcolata per $i = 2, \ldots, n$ viene inserita nella lista $x_{i+1}, \ldots, x_n$ mantenendo l'ordinamento crescente. L'**algoritmo della somma binaria** per $n = 2^k$, somma al primo passo gli addendi a coppie $y_i = x_{2*i-1} + x_{2*i}$, $i = 1, \ldots, n/2$, e ripete il procedimento ricorsivamente a partire da y_i. Il risultato si raggiunge in $\log_2 n$ passi.

Particolarmente interessante è l'**algoritmo di somma compensata** studiato da Kahan (1972) per la somma di n numeri. Si basa sulla seguente osservazione: dati due numeri $a, b \in \mathbb{F}$, $|a| > |b|$, una stima dell'errore commesso, i.e. $(a + b) - \tilde{s}$, dove $\tilde{s} = (a\mathrm{fl}(+)\ b)$, viene fornita da

$$e = -[((a\mathrm{fl}(+)\ b)\mathrm{fl}(-)\ a)\mathrm{fl}(-)\ b] = (a\mathrm{fl}(-)\ \tilde{s})\mathrm{fl}(+)\ b$$

L'algoritmo di Kahan applica la correzione fornita da e ad ogni passo della somma ricorsiva, i.e.

```
s=0
e=0
for i=1:n
    t=s
    y=x(i)+e
    s=t+y
    e=(t-s)+y
end
s=s+e
```

E' stato dimostrato inoltre che vale la maggiorazione

$$|err_{algoritmo}| \leq \left(2u + O(nu^2)\right) \frac{\sum_{i=1}^{n} |x_i|}{|s_n|}.$$

Finchè $nu \leq 1$ la costante di questa limitazione è indipendente da n e rappresenta un miglioramento rispetto (14). In caso di forte cancellazione anche l'algoritmo di somma compensata non è in grado di garantire un errore algoritmico piccolo.

1.8 Complessità computazionale e parametri di qualità del software matematico

Nella valutazione teorica di un algoritmo un parametro importante è la sua **complessità computazionale**. Esso misura il numero delle operazioni aritmetiche necessarie alla sua esecuzione e ci fornisce una stima del tempo di esecuzione.

In generale il costo computazionale viene espresso in funzione alla dimensione n del problema da risolvere. L'agoritmo avrà **complessità costante**, indicata con $\mathcal{O}(1)$, se il numero di operazioni è indipendente dalla dimensione n. La complessità sarà **lineare** o **polinomiale** se richiede un numero di operazioni pari a rispettivamente $\mathcal{O}(n)$ e $\mathcal{O}(n^k)$, con k intero positivo. Alcuni algoritmi possono avere complessità **esponenziale**, i.e. $\mathcal{O}(c^n)$ operazioni, o **fattoriale**, i.e. $\mathcal{O}(n!)$ operazioni.

Il numero di operazioni floating-point eseguite in un secondo dall'elaboratore, **flops**, rappresenta un indicatore della velocità di un calcolatore. Esso ci permette di stimare il tempo di esecuzione di un algoritmo di cui conosciamo il costo computazionale.

Esempio 27 *Il calcolo della soluzione di un sistema lineare di dimensione n mediante la regola di Cramer ([BBCM92, pag.13]) ha una complessità fattoriale* $\mathcal{O}(n!)$. *Su un calcolatore in grado di eseguire 1 mega-flops* (10^6) *servirebbero* $1,6 * 10^6$ *anni per risolvere un sistema di dimensione* $n = 20!$ *Anche con un calcolatore più avanzato con velocità pari a* 10^{15} *flops, sarebbero necessari* 20 *anni per risolvere un sistema di dimensione* $n = 24$. *Anche aumentando la potenza di calcolo l'algoritmo di Cramer è inadeguato alla risoluzione di un sistema lineare e pertanto tale problema matematico va affrontato sviluppando altri algoritmi.*

Quando l'agoritmo viene codificato in un programma altri parametri intervengono nella sua valutazione. Per esempio è importante la gestione della memoria che, assieme al tempo di esecuzione, ne definisce l'**efficienza**. Sia che si utilizzi o che si sviluppi del software matematico, è bene ricordare alcune delle caratteristiche che ne determinano la qualità ([Hea02]):

- **affidabilità** il codice svolge in maniera soddisfacente le funzioni richieste;
- **accuratezza** il codice produce risultati accurati in relazione al problema ed ai dati di ingresso e fornisce una stima dell'accuratezza;
- **efficienza** il codice gestisce al meglio e senza sprechi le risorse di memoria e minimizza il tempo di esecuzione;
- **flessibilità-applicabilità** il codice permette la risoluzione di un'ampia classe di problemi;

- **mantenibilità** il codice e' facilmente modificabile;
- **portabilità** il codice si addatta facilmente a diversi ambienti di calcolo.
- **robustezza** il codice tratta problemi di difficile soluzione, verificando se i dati in ingresso sono consistenti con le specifiche, analizzando e segnalando gli errori in caso di fallimento;
- **testabilità** La correttezza e l'efficienza del codice sono verificabili su esempi test;
- **usabilità** il codice ha un'interfaccia utente di facile comprensione e ben documentata.

Il peso da assegnare ai vari fattori di qualità dipende anche dall'uso del software: i programmi che sono stati sviluppati per essere diffusi e durare come le librerie devono soddisfare ai più alti livelli qualitativi.

1.9 Osservazioni conclusive

L'analisi degli errori di arrotondamento e dell'aritmetica di macchina è stata un importante argomento dell'analisi numerica agli inizi, perchè ha posto l'attenzione e definito importanti concetti quali la stabilità di algoritmi e il condizionamento del problema.

Tuttavia, pur rimanendo fondamentale (visita il sito

`http://www.ima.umn.edu/~arnold/disasters/`

dove sono elencati dei disastri dovuti a computazioni numeriche non eseguite in maniera attenta), riguarda solo una piccola parte dell'analisi numerica perchè la sua anima sta nello sviluppo ed analisi di algoritmi veloci.

2 Esercizi svolti

Esercizio 11 *Utilizzando la formula per la superficie di una sfera*

$$A(r) = 4\pi r^2 \tag{15}$$

e lavorando in aritmetica finita a 4 cifre significative, si calcoli la superficie della terra assumendo $r = 6370$ *km e la variazione di superficie* $\Delta A_1 = A(r+h) - A(r)$ *corrispondente ad un aumento di raggio* $h = 1$ *km. In alternativa, tale variazione si può ottenere con un'approssimazione di primo grado troncando lo sviluppo in serie di Taylor di (15), ovvero*

$$\Delta A_2 = \frac{dA(r)}{dr} \cdot h. \tag{16}$$

Si confrontino i due valori ottenuti e si ripetano i calcoli lavorando in aritmetica finita a 6 cifre significative. Si commentino i risultati analizzando i diversi tipi di errore commessi nel procedimento.

Soluzione. Utilizzando (15) si commette innanzitutto un errore di modellizzazione dato che la terra non è sferica. Per $r = 6370$ km e $\pi = 3.142$ si ottiene

$$A(r) = 4\pi r^2 = 5.100 \times 10^8 \,\mathrm{km}^2$$

dove un ulteriore errore consiste nell'utilizzo di $\pi = 3.142$ come troncamento di una serie infinita. Se il valore del raggio aumenta di $h = 1$ km, la nuova superficie risulta pari a

$$A(r+h) = 4\pi(r+h)^2 = 5.101 \times 10^8 \,\mathrm{km}^2$$

e dunque lavorando con 4 cifre significative la variazione di superficie risulta

$$\Delta A_1 = A(r+h) - A(r) = 1.000 \times 10^5 \,\mathrm{km}^2.$$

Poichè

$$\frac{dA}{dr} = 8\pi r,$$

la variazione di superficie conseguente alla variazione h di raggio è approssimata da

$$\Delta A_2 = 8\pi r h = 1.601 \times 10^5 \,\mathrm{km}^2.$$

Ripetendo il calcolo con un'aritmetica finita a 6 cifre significative si ottengono i seguenti valori ($\pi = 3.14159$):

$$\Delta A_1 = 5.10064 \times 10^8 - 5.09904 \times 10^8 = 1.60000 \times 10^5 \,\mathrm{km}^2$$

$$\Delta A_2 = 1.60095 \times 10^5 \,\mathrm{km}^2$$

per cui si conclude che il secondo algoritmo, seppur accompagnato da un errore analitico (di troncamento della serie di Taylor), risulta più corretto del primo. Ciò è imputabile al fenomeno della cancellazione che interviene nel primo algoritmo, mentre per il secondo si commette un errore analitico pari a

$$\frac{d^2A}{dr^2} \cdot \frac{h^2}{2} = 4\pi h^2 = 1.257 \times 10^1.$$

Svolgendo i calcoli con MATLAB che opera in standard IEEE doppia precisione si ottengono i valori

$$\Delta A_1 = 1.601081279975176 \times 10^5 \,\mathrm{km}^2$$

$$\Delta A_2 = 1.600955616269358 \times 10^5 \,\mathrm{km}^2$$

che confermano la migliore accuratezza del secondo algoritmo. ■

Esercizio 12 *Si analizzi l'errore relativo commesso approssimando la derivata prima di una funzione in un punto usando le differenze finite in avanti*

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}, \quad h > 0.$$

In particolare si fornisca una maggiorazione dell' errore totale in funzione del passo h, assumendo che l'errore nella valutazione di f sia maggiorato dalla precisione di macchina u. Con l'ausilio di MATLAB, si eseguano i calcoli per $f(x) = \tan(x)$ con $x = 1$ e $h = 10^{-1}, 10^{-2}, \ldots, 10^{-16}$.

Soluzione. Si consideri una funzione differenziabile $f : \mathbb{R} \to \mathbb{R}$. La derivata prima in un punto $x \in \mathbb{R}$ fissato è approssimata con le differenze finite (in avanti), ovvero con il rapporto incrementale della funzione:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} = g(x), \quad h > 0.$$

Supponendo $f \in \mathcal{C}^2$ in un intorno di x, arrestando lo sviluppo di Taylor di f al secondo ordine si ottiene

$$f(x+h) = f(x) + f'(x)h + f''(\xi_x)\frac{h^2}{2}$$

per qualche $\xi_x \in (x, x+h)$, ovvero

$$f'(x) - g(x) = -f''(\xi_x)\frac{h}{2}.$$

L'errore (relativo) analitico, definito

$$\varepsilon_{analitico} = \frac{f'(x) - g(x)}{f'(x)}$$

è dunque limitato superiormente da

$$|\varepsilon_{analitico}| \leq \frac{Mh}{2|f'(x)|}$$

dove M è una maggiorazione di $|f''(t)|$ per $t \in [a, b]$,dove $x, x+h \in [a, b]$, per ogni $h \leq h_0$. Risulta pertanto che l'errore analitico tende linearmente a zero per $h \to 0$.

Il rimanente contributo $\varepsilon_{arrotondamento}$ all'errore totale può essere stimato con l'ausilio dei grafi computazionali assumendo un errore relativo nei dati di input $f(x), f(x+h), h$ maggiorato dalla precisione di macchina u. Si ottiene così

$$|\varepsilon_{arrotondamento}| \leq 3u + \frac{u}{\cdot}\frac{|f(x+h)| + |f(x)|}{|f(x+h) - f(x)|} \approx 3u + \frac{2u}{\cdot}\frac{|f(x)|}{h|f'(x)|}.$$

L'andamento dell'errore di arrotondamento rispetto al parametro h è diverso dall'errore analitico: riducendo h tale errore diventa arbitrariamente grande causa la cancellazione. L'errore totale $\varepsilon_{totale} = \varepsilon_{analitico} + \varepsilon_{arrotondamento}$ risulta dunque limitato da

$$|\varepsilon_{totale}| \leq \frac{Mh}{2|f'(x)|} + 3u + \cdot\frac{2|f(x)|u}{h|f'(x)|}$$

e nello scegliere il passo h bisogna 'bilanciare' i due contributi. Il passo h_m che fornisce la scelta migliore si può stimare ponendo $\frac{Mh}{2|f'(x)|} = \cdot\frac{2|f(x)|u}{h|f'(x)|}$ da cui si ottiene

$$h_m \simeq 2\sqrt{\frac{u|f(x)|}{M}}.$$

In corrispondenza di tale valore h_m si ottiene la seguente maggiorazione dell' errore totale

$$|\varepsilon_{totale}| \leq +3u + \cdot \frac{2\sqrt{M|f(x)|u}}{|f'(x)|}$$

Si consideri ora $f(x) = \tan(x)$ per $x = 1$. La derivata prima è $f'(x) = \sec^2(x)$, per cui risulta $f'(1) = 3.425518820814759$, mentre la derivata seconda è $f''(x) = 4\tan(x)/\cos(x)$ per cui si prenda $M \simeq 17.33$ come maggiorazione nell'intervallo $[1, 1.1]$. Considerando la precisione di macchina di MATLAB, pari a $u \simeq 1.11 \times 10^{-16}$, si ottiene $h_m \simeq 6.32 \times 10^{-9}$. I risultati sono riportati in Tabella 9 e in Figura 9. L'ultimo valore del rapporto incrementale risulta zero dato che il passo h è inferiore alla precisione di macchina e dunque $\text{fl}(f(x+h)) = \text{fl}(f(x))$.

h	$\frac{f(x+h)-f(x)}{h}$	errore
10^{-01}	4.073519325937500	6.48e-01
10^{-02}	3.479829956466807	5.43e-02
10^{-03}	3.430863217312341	5.34e-03
10^{-04}	3.426052408281866	5.33e-04
10^{-05}	3.425572171056324	5.34e-05
10^{-06}	3.425524155442616	5.33e-06
10^{-07}	3.425519354838258	5.34e-07
10^{-08}	3.425518846356113	2.55e-08
10^{-09}	3.425518979582876	1.59e-07
10^{-10}	3.425517647315246	1.17e-06
10^{-11}	3.425504324638950	1.45e-05
10^{-12}	3.425704164783383	1.85e-04
10^{-13}	3.421707361894733	3.81e-03
10^{-14}	3.419486915845482	6.03e-03
10^{-15}	3.774758283725532	3.49e-01
10^{-16}	0	3.43e+00

Tabella 9: Errore nell'uso delle differenze finite nel calcolo della derivata prima.

La seguente function MATLAB realizza l'analisi dell'errore di approssimazione della derivata prima per la tangente.

```
function F=findiffor(x,h) F(:,1)=(sec(x))^2*ones(length(h),1);
F(:,2)=(tan(x+h)-tan(x))./h;
F(:,3)=abs(F(:,2)-F(:,1))./abs(F(:,1));
M=max(4*sin(x:.001:x+.1)./(cos(x:.001:x+.1)).^2);
F(:,4)=abs(M*h/2)./abs(F(:,1));
F(:,5)=eps+eps/2*(abs(tan(x+h))+abs(tan(x)))./(h.*abs(F(:,1)));
F(:,6)=F(:,4)+F(:,5); loglog(h,F(:,3),'-b') hold on grid on
loglog(h,F(:,4),'--g') loglog(h,F(:,5),'--c')
loglog(h,F(:,6),'-.r')
```

■

Esercizio 13 *Si vuole studiare come variano gli zeri del polinomio*

$$p(x) = (x-1)(x-2)\cdots(x-20) = x^{20} - 210x^{19} + \cdots$$

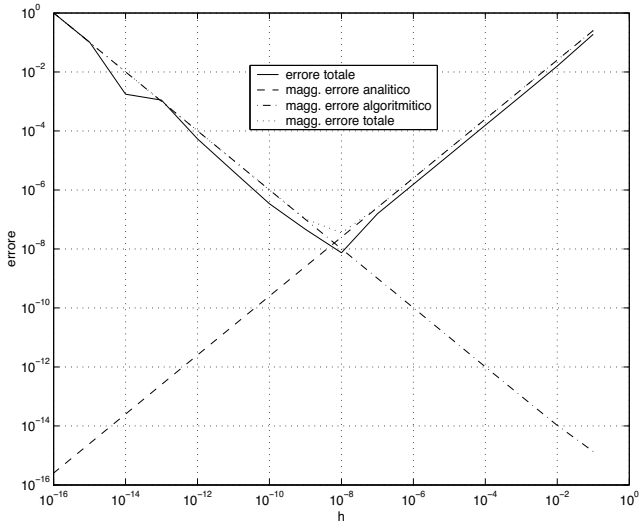

Figura 1: Errore totale nell'approssimazione della derivata con il rapporto incrementale al variare del passo h.

quando si perturbano i suoi coefficienti. Si analizzi il condizionamento del problema del calcolo delle radici di p quando il coefficiente $a = -210$ *di* x^{19} *viene perturbato diventando* $a' = -210 + 2^{-23}$ *commettendo un errore relativo pari a* $\varepsilon_a = 5.677 \times 10^{-10}$.

Soluzione. Il condizionamento dell'i-esima radice è misurato da

$$C_i = a\frac{f_i'(a)}{f_i(a)}$$

dove $f_i(a) = x_i(a)$ è la funzione che fornisce l'i-esima radice del polinomio in corrispondenza del valore a del coefficiente. Dal teorema delle funzioni implicite si ha

$$f_i'(a) = -\frac{\partial p}{\partial a}\Big/\frac{\partial p}{\partial x}\,|_{x=i} = \frac{i^{19}}{\prod_{k=1,\,k\neq i}^{20}(i-k)}.$$

Tramite le seguenti istruzioni MATLAB che implementano i calcoli precedenti

```
function [r,rr,err,dfa]=polycond(N,n,ea)
r=[1:N]';                          %radici
cp=poly(r);                        %coefficienti polinomio
era=abs(ea)/abs(cp(N-n+1));        %errore relativo di a
cp(N-n+1)=cp(N-n+1)+ea;
rr=flipud(roots(cp));              %radici perturbate
```

```
err=abs(rr-r)./r;                    %errore relativo radici
C=ones(N,1)*r'-r*ones(1,N); P=prod(reshape(C(find(C)),N-1,N));
dfa=(r.^(n))./P';                    %f'(a)
```

si ottengono i risultati in Tabella 10 per $N = 20$.

radice	radice perturbata	errore relativo	$f_i'(a)$
1	1.0000e+00	1.8252e-13	-8.2206e-18
2	2.0000e+00	3.6771e-12	8.1890e-11
3	3.0000e+00	2.2747e-10	-1.6338e-06
4	4.0000e+00	2.3822e-09	2.1896e-03
5	5.0000e+00	7.3107e-09	-6.0774e-01
6	6.0000e+00	7.7604e-07	5.8248e+01
7	7.0003e+00	3.8578e-05	-2.5424e+03
8	7.9933e+00	8.3844e-04	5.9696e+04
9	9.1442e+00	1.6025e-02	-8.3933e+05
10	9.5058e+00	4.9416e-02	7.5941e+06
11	1.0893e+01 -1.1488i	1.0489e-01	-4.6445e+07
12	1.0893e+01 +1.1488i	1.3297e-01	1.9850e+08
13	1.2822e+01 -2.1236i	1.6393e-01	-6.0556e+08
14	1.2822e+01 +2.1236i	1.7347e-01	1.3330e+09
15	1.5306e+01 -2.7755i	1.8615e-01	-2.1191e+09
16	1.5306e+01 +2.7755i	1.7881e-01	2.4075e+09
17	1.8181e+01 -2.5490i	1.6526e-01	-1.9044e+09
18	1.8181e+01 +2.5490i	1.4197e-01	9.9559e+08
19	2.0477e+01 -1.0390i	9.5036e-02	-3.0901e+08
20	2.0477e+01 +1.0390i	5.7160e-02	4.3100e+07

Tabella 10: Condizionamento del calcolo degli zeri di polinomio.

NOTA. Difficile può risultare anche la valutazione di un polinomio in prossimità delle sue radici. Si consideri infatti il polinomio $p(x) = (x - x_1)(x - x_2)\cdots(x - x_n)$ che ha n-radici $x_1, x_2, \ldots, x_n$. Il numero di condizionamento risulta infatti

$$C = \sum_{i=1}^{n} \frac{x}{x - x_i}$$

per cui il problema della valutazione di $p(x)$ è malcondizionato per $x \simeq x_i$.

Infine può intervenire anche il fenomeno della cancellazione: si consideri infatti il polinomio $p(x) = (x-1)^6$ che ha radice $x = 1$ con molteplicità $\nu = 6$ e che può essere espresso anche come $p(x) = x^6 - 6x^5 + 15x^4 - 20x^3 + 15x^2 - 6x + 1$. Se si rappresenta con MATLAB $p(x)$ per $x = [0.995 : 0.0001 : 1.005]$ in entrambe le forme si ottengono risultati diversi (Figura 2): la seconda rappresentazione è infatti accompagnata da cancellazione tra i termini di grado più elevato. Entrambi gli algoritmi di calcolo sono instabili in prossimità della radice (vedi grafi dell'errore computazionale), ma il secondo lo è molto più del primo.

■

Esercizio 14 *(vedi anche* $24/03/03 - \sharp 1$*) Calcolando in aritmetica di macchina standard IEEE doppia precisione le seguenti espressioni matematicamente*

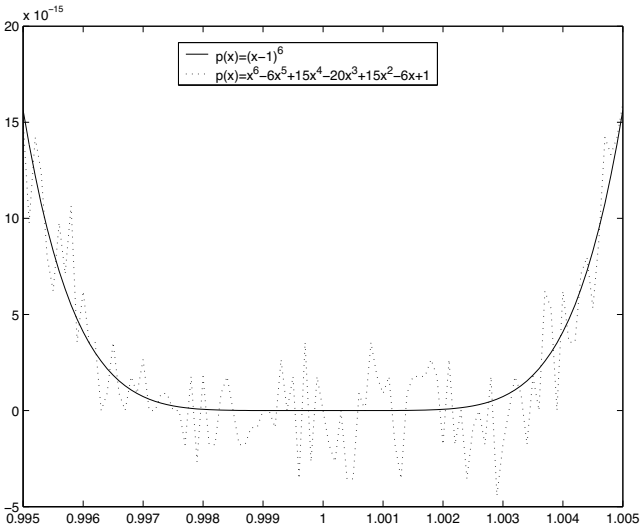

Figura 2: Rappresentazione di $p(x) = (x-1)^6$ e di $p(x) = x^6 - 6x^5 + 15x^4 - 20x^3 + 15x^2 - 6x + 1$.

equivalenti

$$\frac{1 - cos(x)}{x} = \frac{sin^2(x)}{x(1 + cos(x))}$$

per valori di x positivi, $|x| << 1$*, si ottengono i risultati in Tabella 11. Spiega i risultati.*

x	$\frac{1-\cos(x)}{x}$	$\frac{\sin^2(x)}{x(1+\cos(x))}$
0	NaN	NaN
1.2000e-08	9.2519e-09	6.0000e-09
1.6000e-08	6.9389e-09	8.0000e-09
2.0000e-08	1.1102e-08	1.0000e-08
2.4000e-08	1.3878e-08	1.2000e-08
2.8000e-08	1.5860e-08	1.4000e-08
3.2000e-08	1.7347e-08	1.6000e-08
3.6000e-08	1.8504e-08	1.8000e-08
4.0000e-08	1.9429e-08	2.0000e-08
4.4000e-08	2.2709e-08	2.2000e-08
4.8000e-08	2.3130e-08	2.4000e-08
5.0000e-08	2.4425e-08	2.5000e-08

Tabella 11: Valori delle espressioni al variare di x.

Soluzione. Tenendo conto che

$$\lim_{x \to 0} \frac{1 - \cos(x)}{x^2} = \frac{1}{2}$$

e che

$$\lim_{x\to 0} \frac{\sin(x)}{x} = 1,$$

entrambe le espressioni dovrebbero approssimare $x/2$. La prima espressione è però accompagnata dal fenomeno della cancellazione, per cui la seconda risulta più accurata come verificabile dalla Tabella 11. ■

Esercizio 15 *Si vuole implementare l'algoritmo*

$$\begin{aligned} &y_1 = x^2 \\ &y_{i+1} = \sqrt{y_i}, \quad i = 1, \ldots, m \\ &z_1 = y_{m+1} \\ &z_{i+1} = z_i^2, \quad i = 1, \ldots, m-1 \end{aligned}$$

in una function MATLAB del tipo

```
[z,err]=mroot(x,m)
```

dove $z = z_m$ *ed* $err = \frac{|z_m - x|}{|x|}$ *è l'errore relativo tra l'input* x *e l'output* z *che analiticamente coincidono. Eseguendo la function per* $x = 0.25 : 0.25 : 1.5$ *con* $m = 50$ *si ottengono i valori*

```
>>x=.25:.25:1.5
x=
   0.2500    0.5000    0.7500    1.0000    1.2500    1.5000
>>[z,err]=mroot(x,50)
z=
   0.2375    0.4724    0.7316    1.0000    1.1331    1.4550
err=
   0.0499    0.0553    0.0245         0    0.0935    0.0300
```

Si commentino i risultati.

Soluzione. Una possibile implementazione è la seguente:

```
function [z,err]=mroot(x,m) y=x.^2; for i=1:m
    y=sqrt(y);
end z=y; for i=1:m-1
    z=z.^2;
end err=abs(z-x)./abs(x);
```

Per spiegare i risultati numerici analizziamo l'errore algoritmico mediante l'uso dei grafi computazionali in maniera ricorsiva. Ricordando che l'errore in ogni operazione è maggiorato dalla precisione di macchina u e considerando che l'estrazione della radice quadrata presenta un coefficiente di amplificazione pari a $1/2$ mentre per l'elevamento al quadrato tale coefficiente è pari a 2, otteniamo

perl'errore algoritmico la seguente maggiorazione

$$\begin{aligned} |\epsilon_{alg}| &\leq u\left(\sum_{i=0}^{m-2} 2^i + 2^{m-1}\sum_{i=0}^{m} 2^{-i}\right) = \\ &= u\left(2^m + 2^{m-1} - \frac{3}{2}\right) \\ &= u\frac{3}{2}\left(2^m - 1\right), \end{aligned}$$

dove si è utilizzata la formula

$$\sum_{i=0}^{m-1} x^i = \frac{x^m - 1}{x - 1}.$$

Si osservi dunque come l'instabilità dell'algoritmo aumenta con m in conseguenza al ripetuto elevamento al quadrato con coefficiente di amplificazione totale dell'ordine di 2^m. Essendo infine nello standard IEEE doppia precisione $u = 2^{-53}$, con $m = 50$ l'errore algoritmico totale risulta dell'ordine di $3/16 = 0.1875$. ■

Esercizio 16 *Si vuole scrivere una function MATLAB per valutare gli errori assoluto e relativo nell'approssimazione di Stirling per il fattoriale*

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

Si costruisca la function in modo che calcoli gli errori per un input n generico vettore di valori interi e si commentino i risultati.

Soluzione. Un esempio è il seguente:

```
function [erass,errel]=stirling(n)
% Calcola gli errori assoluto e relativo
% nell'approssimazione di Stirling per
% un vettore di interi 'n'.
for i=1:length(n)
    fatt(i)=prod(1:n(i));
end stir=sqrt(2*pi*n).*(n/exp(1)).^n; erass=abs(stir-fatt);
errel=erass./fatt; subplot(2,1,1) semilogy(n,erass,'-b')
subplot(2,1,2) semilogy(n,errel,'-r')
```

Eseguendo la *function* per $n = 1, 2, \ldots, 171$ si ottengono i risultati riportati in Figura 3. Si osservi dunque che l'errore assoluto cresce seguendo l'ordine del fattoriale stesso mentre quello relativo dimiuisce al crescere di n. L'approssimazione di Stirling è dunque tanto migliore quanto maggiore è l'intero n considerato. Un esame più accurato dimostra inoltre che il fattoriale esatto è sempre una maggiorazione di quello approssimato con la formula di Stirling.

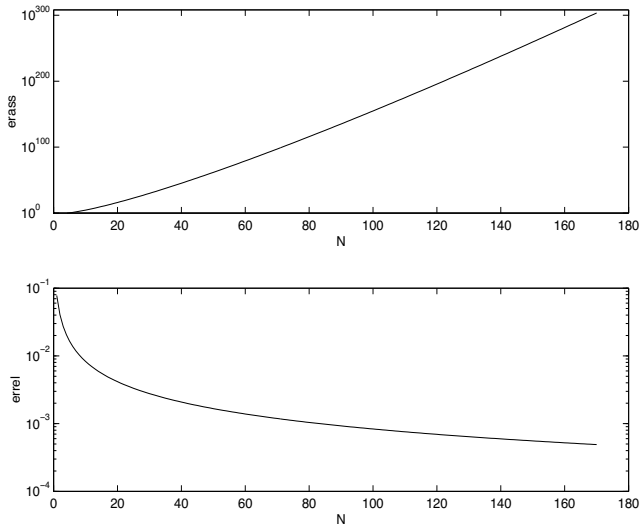

Figura 3: Errori assoluto e relativo nell'approssimazione di $n!$ con la formula di Stirling.

Infine si osservi che per $n = 171$ si ottiene $erass = Inf$ ed $errel = NaN$, cioè il risultato di un'operazione matematicamente indeterminata. Ciò è dovuto alla computazione di $fatt(171)$ che risulta maggiore di $realmax$ e dunque per MATLAB è $fatt(171) = Inf$. Di conseguenza si ha $erass = |3.7842 \times 10^{307} - Inf| = Inf$ ed $errel = Inf/Inf = NaN$. ■

Esercizio 17 *(vedi anche* $10/07/03 - \sharp 1$*) Si vuole approssimare il valore della costante* $e = 2.7182818284590\ldots$ *(numero di Nepero) utilizzando la sua definizione*

$$\lim_{n\to\infty}(1+\frac{1}{n})^n. \tag{17}$$

Calcolando $e_n = (1+\frac{1}{n})^n$ *per* $n = 10^k, k = 1, 2, \ldots, 18$*, in aritmetica di macchina secondo lo standard IEEE doppia precisione si ottengono i risultati in Tabella 12, dove err rappresenta l'errore relativo. Spiega i risultati.*

Soluzione. Approssimando (17) con e_n si commette un errore composto da due contributi. Il primo contributo è costituito dal troncamento nell'approssimare il passaggio al limite. Tale errore analitico diminuisce ovviamente al crescere di n. Il secondo contributo è dovuto alla cancellazione nella somma $n + \frac{1}{n}$ e cresce al crescere di n. L'errore totale raggiunge dunque un minimo (per $k = 8$) che rappresenta il compromesso tra l'errore analitico e la cancellazione, poi aumenta per effetto di quest'ultima fino a stabilizzarsi. Per $k \geq 16$ infatti, $\frac{1}{n}$ scende sotto la precisione di macchina dello standard IEEE doppia precisione, per cui sommandolo ad n (=1) quest'ultimo rimane invariato.

k	e_n	err
1	2.593742460	4.5815e-02
2	2.704813829	4.9546e-03
3	2.716923932	4.9954e-04
4	2.718145926	4.9995e-05
5	2.718268237	4.9999e-06
6	2.718280469	5.0008e-07
7	2.718281694	4.9416e-08
8	2.718281798	1.1077e-08
9	2.718282052	8.2240e-08
10	2.718282053	8.2690e-08
11	2.718282053	8.2735e-08
12	2.718523496	8.8905e-05
13	2.716110034	7.9896e-04
14	2.716110034	7.9896e-04
15	3.035035206	1.1653e-01
16	1	6.3212e-01
17	1	6.3212e-01
18	1	6.3212e-01

Tabella 12: Approssimazione del numero di Nepero.

NOTA. Il fatto che $\frac{1}{n}$ non influisca su n non è dovuto al fenomeno di *underflow*! ■

Esercizio 18 *(vedi anche* $10/07/03 - \sharp 2$*) Dovendo valutare* $log(x) - log(y) = log(\frac{x}{y})$ *per* $x \approx y$*, quale delle due espressioni useresti e perchè?*

Soluzione. La scelta è ininfluente poichè entrambe le espressioni forniscono risultati non accurati, la prima per il fenomeno della cancellazione nella differenza tra i logaritmi, la seconda per il malcondizionamento nel calcolo del logaritmo. Utilizzando i grafi computazionali, supponendo che gli errori di valutazione siano tutti uguali ad η e che tale quantità sia limitata in valore assoluto dalla precisione di macchina u ($|\eta| \leq u$), si ottengono le seguenti limitazioni per l'errore algoritmico:

$$|\varepsilon_{alg1}| \leq u \left(1 + \left|\frac{\log x}{\log x - \log y}\right| + \left|\frac{\log y}{\log x - \log y}\right|\right)$$

$$|\varepsilon_{alg2}| \leq u \left(1 + \left|\frac{1}{\log \frac{x}{y}}\right|\right).$$

Entrambe divergono per $x \approx y$. ■

Esercizio 19 *(vedi anche* $24/07/03 - \sharp 1$*) Sia* $x \in \mathbb{F}(B, t, p_{min}, p_{max})$ *un numero di macchina sufficientemente grande da verificare* $x + 10 = x$*. Sommando in aritmetica di macchina i seguenti numeri* $1, 2, 3, 4, x, -x$ *con diversi ordinamenti (per esempio: 1+2+3+4+x-x, 1+2+x-x+3+4, 2+3+x+1-x+4, ...) che risulati ottieni? Commenta e giustifica la risposta. Nello standard IEEE doppia precisione i parametri che definiscono l'insieme dei numeri floating-point sono*

$B = 2$, $t = 53$, $p_{min} = 1021$, $p_{max} = 1024$. *Sai dare una limitazione inferiore per l'esponente p tale che $x = B^p$ soddisfi alle condizioni sopra?*

Soluzione. Se x è un numero macchina tale che $x + 10 = x$, allora si ottiene:

$$1 + 2 + 3 + 4 + x - x = 0,$$

$$1 + 2 + x - x + 3 + 4 = 7$$

e

$$2 + 3 + x + 1 - x + 4 = 4,$$

ovvero se $y_1, y_2 \leq 10$, allora il risultato finale della somma $y_1 + x + y_2 - x + z$ è z poichè le cifre significative degli addendi y_1 e y_2 non influiscono su quelle di x.

Per dare una limitazione inferiore all'esponente p di $x = B^p$ che soddisfi la condizione $x + 10 = x$ si considera la rappresentazione secondo lo standard IEEE del numero 10, ovvero $10 = (.101)_2 \cdot 2^4$. Si osserva che la prima cifra significativa ha esponente 3, per cui per non influire sulle cifre significative di $x = B^p$ dev'essere $p \geq 53 + 3 + 1 = 57$. Se infatti $p = 57$, allora l'ultima cifra significativa di x ha esponente $4 > 3$ e non viene influenzata dalla prima cifra significativa del numero 10. ■

Esercizio 20 *(vedi anche HW*01 2002/03$-\sharp 1$*) Si considerino le due espressioni matematicamente equivalenti*

$$r_1 = \frac{ay + b}{ay + c}, \quad r_2 = \frac{a + b/y}{a + c/y}$$

e si calcolino i loro valori per

$$a = b = \beta^{-p_{min}-t}, \quad c = 2(\beta - 1)\beta^{-p_{min}-t}, \quad y = (\beta - 1)\beta^{-1}$$

utilizzando MATLAB ovvero secondo lo standard IEEE doppia precisione basato sul sistema floating point $F(B, t, p_{min}, p_{max}) = F(2, 53, 1021, 1024)$ a cui si aggiungono i numeri denormalizzati, cioè con esponente $p = -p_{min} - 1$ e prima cifra nulla. Si dica quale delle due formule fornisce il risultato esatto confrontando i valori calcolati con MATLAB con il risultato teorico ottenuto analiticamente in aritmetica esatta. Si spieghi la differenza tra i valori computati dando una giustificazione analitica per il risultato errato.

Soluzione. Le istruzioni MATLAB sono le seguenti:

```
>>beta=2;t=53;pmin=1021;
>>a=beta^(-pmin-t),b=a,c=2*(beta-1)*beta^(-pmin-t),...
y=(beta-1)*beta^(-1) a=
 4.9407e-324
b=
 4.9407e-324
```

```
c=
 9.8813e-324
y=
   0.5000
>>r1=(a*y+b)/(a*y+c)
r1=
   0.5000
>>r2=(a+b/y)/(a+c/y)
r2=
   0.6000
```

In aritmetica esatta si ha:

$$\begin{aligned} r_2 = r_1 &= \frac{ay+b}{ay+c} = \\ &= \frac{\beta^{-p_{min}-t}(\beta-1)\beta^{-1} + \beta^{-p_{min}-t}}{\beta^{-p_{min}-t}(\beta-1)\beta^{-1} + 2(\beta-1)\beta^{-p_{min}-t}} = \\ &= \frac{2-\frac{1}{\beta}}{2\beta-1-\frac{1}{\beta}} = \\ &= \frac{2\beta-1}{2\beta^2-\beta-1}|_{\beta=2} = \frac{3}{5} = 0.6 \end{aligned}$$

per cui il valore esatto con MATLAB lo si ottiene usando la formula r_2. L'uso della formula r_1 con i dati assegnati non fornisce il valore esatto, ma un valore approssimato con un errore di $\simeq 17\%$. Tale risultato è dovuto all'*underflow*: il valore $a = b = \beta^{-p_{min}-t} = 2^{-1074} \simeq 4.9407 \times 10^{-324}$ (che corrisponde a $realmin * eps$) è infatti il numero minimo rappresentabile con MATLAB per cui la moltiplicazione per $y = 1/2$ fornisce il valore $ay = by = 0$. Risulta perciò in aritmetica di macchina

$$r_1 = \frac{ay \oplus b}{ay \oplus c} = \frac{b}{c} = \frac{1}{2(\beta-1)}|_{\beta=2} = \frac{1}{2} = 0.5.$$

■

Esercizio 21 *(vedi anche HW*01 2002/03−♯2*) Si scriva una function MATLAB per il calcolo del valore approssimato della derivata prima di una funzione in un punto usando le differenze finite centrate*

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}, \quad h > 0.$$

Si testi la function per la funzione esponenziale $f(x) = e^x$ per $x = 1$, valutando l'errore relativo totale (assumendo che x, h, $x+h$ e $x-h$ sono numeri di macchina e che valutando f si commette un'errore di arrotondamento maggiorato dalla precisione di macchina) commesso rispetto al valore esatto di $f'(x)$ per un passo variabile $h = 10^{-1}, 10^{-2}, \ldots, 10^{-16}$. Si rappresentino su scala bilogaritmica l'andamento dell'errore relativo totale e delle maggiorazioni dell'errore analitico e algoritmico in funzione del passo h. La function deve essere del tipo

`[erass,erana,erarr]=nome_function(x,h)`

e deve provvedere alla rappresentazione grafica dei risultati. (Sugg.: per determinare una maggiorazione dell'errore analitico si utilizzi lo sviluppo di Taylor per $f(x+h)$ e per $f(x-h)$ arrestandosi al terz'ordine e si faccia la differenza dei due sviluppi ricordando che esiste $\xi_x \in [x-h, x+h]$ tale che $f'''(\xi_x^+) + f'''(\xi_x^-) = 2f'''(\xi_x)$ per qualche $\xi_x^+ \in (x, x+h)$ e $\xi_x^- \in (x-h, x)$)

Soluzione.

Supponendo che $f \in \mathcal{C}^3[a,b]$, a, b tali che $x+h$, $x-h \in [a,b]$, si può applicare lo sviluppo di Taylor arrestato al terz'ordine ad $f(x+h)$ e ad $f(x-h)$ ottenendo

$$f(x+h) = f(x) + f'(x)h + f''(x)\frac{h^2}{2} + f'''(\xi_x^+)\frac{h^3}{6}$$

$$f(x-h) = f(x) - f'(x)h + f''(x)\frac{h^2}{2} - f'''(\xi_x^-)\frac{h^3}{6}$$

per qualche $\xi_x^+ \in (x, x+h)$ e $\xi_x^- \in (x-h, x)$. Sottraendo la seconda dalla prima si ottiene

$$f(x+h) - f(x-h) = 2f'(x)h + \left(f'''(\xi_x^+) + f'''(\xi_x^-)\right)\frac{h^3}{6}$$

per cui dividendo per $2h$ e considerando che esiste $\xi_x \in [x-h, x+h]$ tale che $f'''(\xi_x^+) + f'''(\xi_x^-) = 2f'''(\xi_x)$, posto

$$g(x) = \frac{f(x+h) - f(x-h)}{2h},$$

si ottiene

$$f'(x) - g(x) = -f'''(\xi_x)\frac{h^2}{6}.$$

L'errore (relativo) analitico, definito

$$\varepsilon_{analitico} = \frac{f'(x) - g(x)}{f'(x)}$$

è dunque limitato superiormente da

$$|\varepsilon_{analitico}| \leq \frac{Mh^2}{6|f'(x)|}$$

dove $M = \max_{t\in[a,b]} |f'''(t)|$. L' errore analitico tende a zero per h che tende a zero in maniera quadratica. La parte rimanente dell' errore totale che chiamiamo errore di arrotondamento può essere stimata con l'ausilio dei grafi computazionali associando ai dati in ingresso $f(x-h), f(x+h), h$ un errore maggiorato dalla precisione di macchina u. L'errore di arrotondamento risulta limitato superiormente da

$$|\varepsilon_{arrotondamento}| \leq 3u \cdot \frac{u(|f(x+h)| + |f(x-h)|)}{2h|f'(x)|}.$$

e quindi aumenta per h che tende a zero. L'errore totale ε_{totale} risulta dunque limitato da

$$|\varepsilon_{totale}| \leq \frac{Mh^2}{6|f'(x)|} + 3u + \frac{u(|f(x+h)| + |f(x-h)|)}{2h|f'(x)|}$$

e nello scegliere il passo h bisogna 'bilanciare' i due contributi. Il passo h_m che fornisce la scelta migliore si può stimare ponendo $\frac{Mh^2}{6|f'(x)|} = \cdot\frac{u(|f(x+h)|+|f(x-h)|)}{2h|f'(x)|}$ da cui si ottiene

$$h_m \approx \sqrt[3]{\frac{6u|f(x)|}{M}}.$$

In corrispondenza di tale valore h_m si ottiene la seguente maggiorazione dell' errore totale

$$|\varepsilon_{totale}| \leq 3u + \cdot\frac{2\sqrt[3]{M|f(x)|^2u^2}}{|f'(x)|}$$

Si consideri ora $f(x) = e^x$ per $x = 1$. La derivata prima è $f'(x) = e^x$, per cui risulta $f'(1) \simeq 2.718281828459046$; la derivata terza è $f'''(x) = e^x$ per cui si prenda $M = 3.0042 \geq e^{1.1}$ come maggiorazione nell'intervallo $[0.9, 1.1]$. I risultati sono riportati in Tabella 13 e in Figura 4 e confermano quanto previsto.

h	erass	erana	erarr
10^{-01}	4.53e-03	5.00e-03	2.22e-15
10^{-02}	4.53e-05	5.006e-05	2.22e-14
10^{-03}	4.53e-07	5.01e-07	2.22e-13
10^{-04}	4.53e-09	5.01e-09	2.22e-12
10^{-05}	5.85e-11	5.01e-11	2.22e-11
10^{-06}	1.634e-10	5.01e-13	2.22e-10
10^{-07}	5.86e-11	5.01e-15	2.22e-09
10^{-08}	6.60e-09	5.01e-17	2.22e-08
10^{-09}	6.60e-09	5.01e-19	2.22e-07
10^{-10}	6.73e-07	5.01e-21	2.22e-06
10^{-11}	1.04e-05	5.01e-23	2.22e-05
10^{-12}	2.10e-04	5.01e-25	2.22e-04
10^{-13}	4.56e-04	5.01e-27	2.22e-03
10^{-14}	9.34e-03	5.01e-29	2.22e-02
10^{-15}	1.68e-01	5.01e-31	2.22e-01
10^{-16}	4.98e-01	5.001e-33	2.22e+00

Tabella 13: Errore nell'uso delle differenze finite centrate nel calcolo della derivata prima.

La seguente *function* di MATLAB realizza l'analisi dell'errore di approssimazione della derivata prima per $f(x) = e^x$:

```
function [erass,erana,erarr]=diffincen(x,h)

d1f=exp(x)*ones(length(h),1); d1fc=(exp(x+h)-exp(x-h))./(2*h);
erass=abs(d1fc-d1f)./abs(d1f); M=max(exp(x-max(h):.001:x+max(h)));
erana=abs(M*h.^2/6)./abs(d1f);
```

```
erarr=eps+eps/4*(abs(exp(x+h))+abs(exp(x-h)))./(h.*d1f);
ertot=erana+erarr; loglog(h,erass,'-b') hold on grid on
loglog(h,erana,'--g') loglog(h,erarr,'--c') loglog(h,ertot,'-.r')
```

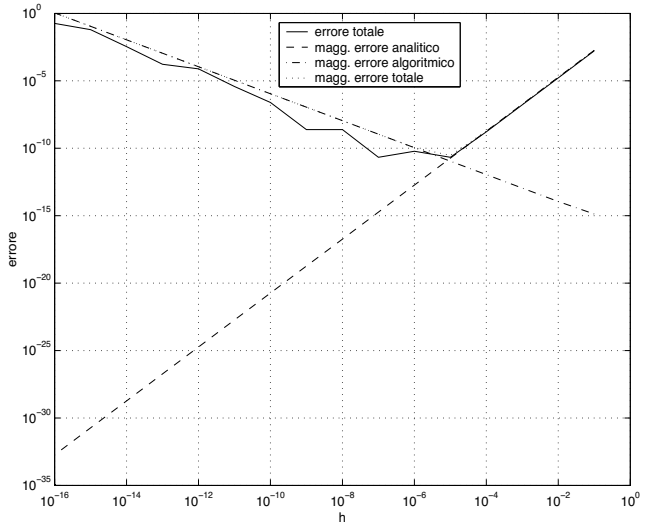

Figura 4: Errori nell'approssimazione della derivata con le differenze finite centrate al variare del passo h.

■

Esercizio 22 *(vedi anche HW*01 2002/03 − ♯3*) Si calcolino le seguenti espressioni matematicamente equivalenti*

$$\sqrt{n+1} - \sqrt{n} = \frac{1}{\sqrt{n+1} + \sqrt{n}}$$

per $n = 10, 10^2, \ldots, 10^{16}$ *con MATLAB ovvero secondo lo standard IEEE doppia precisione. Si spieghino i risultati ottenuti e si analizzi l'errore algoritmico di entrambe le espressioni.*

Soluzione. Le istruzioni MATLAB sono le seguenti

```
>>format long e
>>n=10.^([1:16]');
>>alg1=sqrt(n+1)-sqrt(n);
>>alg2=1./(sqrt(n+1)+sqrt(n));
```

e i risultati sono riportati in Tabella 14.

I risultati attesi non dovrebbero scostarsi di molto dai seguenti valori:

$$\begin{cases} \frac{\sqrt{10}}{2} \times 10^{-k} \simeq 1.5811388 \times 10^{-k} \text{ per } n = 10^{2k-1} \text{ e } k = 1, 2, \ldots, 8 \\ 5 \times 10^{-k-1} \text{ per } n = 10^{2k} \text{ e } k = 1, 2, \ldots, 8 \end{cases}$$

n	$\sqrt{n+1}-\sqrt{n}$	$\frac{1}{\sqrt{n+1}+\sqrt{n}}$
10^{01}	1.543471301870203e-01	1.543471301870205e-01
10^{02}	4.987562112088995e-02	4.987562112089027e-02
10^{03}	1.580743742895763e-02	1.580743742895582e-02
10^{04}	4.999875006248544e-03	4.999875006249609e-03
10^{05}	1.581134877255863e-03	1.581134877256878e-03
10^{06}	4.999998750463419e-04	4.999998750000625e-04
10^{07}	1.581138790243131e-04	1.581138790555721e-04
10^{08}	5.000000055588316e-05	4.999999987500000e-05
10^{09}	1.581139076733962e-05	1.581138829688905e-05
10^{10}	4.999994416721165e-06	4.999999999875000e-06
10^{11}	1.581152901053429e-06	1.581138830080237e-06
10^{12}	5.000038072466850e-07	4.999999999998749e-07
10^{13}	1.578591763973236e-07	1.581138830084150e-07
10^{14}	5.029141902923584e-08	4.999999999999987e-08
10^{15}	1.862645149230957e-08	1.581138830084189e-08
10^{16}	0	5.000000000000000e-09

Tabella 14: Valori di $\sqrt{n+1}-\sqrt{n}$ e di $\frac{1}{\sqrt{n+1}+\sqrt{n}}$ ottenuti con MATLAB per $n = 10, 10^2, \ldots, 10^{16}$.

ottenuti con la seconda formula approssimata da $\frac{1}{2\sqrt{n}}$, per cui confrontando i valori in Tabella proprio la seconda formula è quella che fornisce i valori più accurati. In particolare, per $n = 10^{16}$ la prima formula ritorna il valore zero poichè, lavorando con precisione di macchina $u \simeq 1.11 \times 10^{-16}$, risulta $n+1 = n$ dato che l'aggiunta di 1 non influisce sulle cifre significative di n.

Per l'analisi dell'errore algoritmico si trascuri l'errore sul dato iniziale n supponendo (com'è il caso) che sia esattamente rappresentabile secondo lo standard IEEE doppia precisione. Indicati con $\varepsilon^{(i)}$ e $\eta^{(i)}$ gli errori locali generati dalle i-esime operazioni rispettivamente nel primo e nel secondo algoritmo si ha:

$$\begin{cases} \varepsilon_{alg1} = \varepsilon^{(1)} + \frac{\sqrt{n+1}}{\sqrt{n+1}-\sqrt{n}}\varepsilon^{(2)} + \frac{1}{2}\frac{\sqrt{n+1}}{\sqrt{n+1}-\sqrt{n}}\varepsilon^{(3)} - \frac{\sqrt{n}}{\sqrt{n+1}-\sqrt{n}}\varepsilon^{(4)} \\ \varepsilon_{alg2} = \eta^{(1)} - \eta^{(2)} - \frac{\sqrt{n+1}}{\sqrt{n+1}+\sqrt{n}}\eta^{(3)} - \frac{1}{2}\frac{\sqrt{n+1}}{\sqrt{n+1}+\sqrt{n}}\eta^{(4)} - \frac{\sqrt{n}}{\sqrt{n+1}+\sqrt{n}}\eta^{(5)} \end{cases}$$

Ipotizzando poi $|\varepsilon^{(i)}| \leq u$ e $|\eta^{(i)}| \leq u$ per ogni i si ottengono le maggiorazioni

$$\begin{cases} |\varepsilon_{alg1}| \leq u\left(1 + \frac{\frac{3}{2}\sqrt{n+1}+\sqrt{n}}{|\sqrt{n+1}-\sqrt{n}|}\right) \\ |\varepsilon_{alg2}| \leq u\left(2 + \frac{\frac{3}{2}\sqrt{n+1}+\sqrt{n}}{\sqrt{n+1}+\sqrt{n}}\right) \end{cases}$$

per cui il primo algoritmo risulta instabile per $\sqrt{n+1} \simeq \sqrt{n}$ ovvero per $n \to \infty$.
■

3 Esercizi da svolgere

Esercizio 23 *Considera la seguente espressione:*

$$\frac{1}{1-x} - \frac{1}{1+x},$$

con $x \neq \pm 1$.

- *Definisci in generale l'errore inerente. Per quali valori di* x*, la valutazione dell'espressione risulta un problema malcondizionato?*
- *Definisci in generale l'errore algoritmico. Per quali valori di* x *numero di macchina, la valutazione dell'espressione in artimetica di macchina può fornire risultati non accurati?*
- *Proponi un'espressione equivalente che possa fornire risultati accurati nel caso in cui* x *numero di macchina assuma i valori trovati al punto precedente.*

Esercizio 24 *Considera l'insieme* $\mathbb{F}$ *dei numeri floating-point con* $B = 2$, $t = 4$, $p_{min} = 2$, $p_{max} = 3$, *che contiene anche i numeri denormalizzati per un underflow graduale. Calcola*

- *il più piccolo numero floating point positivo in* $\mathbb{F}$ *normalizzato;*
- *il più grande numero floating point positivo in* $\mathbb{F}$ *normalizzato;*
- *il più piccolo numero floating point positivo in* $\mathbb{F}$ *denormalizzato;*
- *la precisione di macchina* u *per l'arrotondamento.*

Dati $x_1, x_2 \in \mathbb{F}$ *positivi, quali sono la minima e la massima distanza possibile tra* x_1 *e* x_2*?*

Esercizio 25 *Definisci l'errore inerente ed algoritmico nel calcolo di una funzione razionale* $y = f(x), x, y \in \mathbb{R}$, *spiegando il significato di condizionamento di un problema e di stabilità in avanti di un algoritmo. Calcola tali errori nel caso della somma di due numeri, i.e.* $y = f(x_1, x_2) = x_1 + x_2$, *parlando del fenomeno della cancellazione. Definisci la precisione di macchina* u *e spiega perchè risulta* $1 + u = 1$ *in aritmetica di macchina. Per* $a = B^k$, *dove* B *è la base di rappresentazione dei numeri floating-point e* k *è un intero positivo, determina una maggiorazione* β *di* $b > 0$ *per cui vale* $a + b = a$, $0 < b < \beta$, *in aritmetica di macchina. Sulla base delle considerazioni sopra esposte, spiega perchè la serie armonica divergente*

$$\sum_{n=1}^{\infty} \frac{1}{n}$$

può avere un limite finito in aritmetica di macchina.

Esercizio 26 *Considera un sistema di numeri di macchina con* $B = 10$, $t = 6$ *e* $x_1 = 1.23456$ *e* $x_2 = 1.23459$.

- *Quante cifre significative contiene* $x_2 - x_1$*?*
- *In un sistema normalizzato, indica il minimo valore degli esponenti* p_{min} *e* p_{max} *per cui* $x_1, x_2, x_2 - x_1$ *sono rappresentati esattamente.*
- *Per tale sistema calcola* $realmax$, $realmin$ *e la precisione di macchina* u.

4 Domande di verifica

1. Il condizionamento del problema dipende dall'algoritmo scelto per la sua risoluzione: vero o falso?
2. Il mal condizionamento di un problema può essere migliorato passando dalla singola alla doppia precisione: vero o falso?
3. Spiega le differenza tra errore relativo e assoluto.
4. Definisci la precisione di macchina.
5. Cosa si intende per "cancellazione"?
6. L'operazione di somma in aritmetica di macchina è associativa ma non commutativa: vero o falso?
7. Elenca tre fonti d'incertezza nel calcolo scientifico.
8. Definisci la stabilità in avanti di un algoritmo.

Riferimenti bibliografici

[QS02] **testo consigliato:** Quarteroni, A., Saleri, F. (2002) *Introduzione al Calcolo Scientifico*, Springer, Milano.

[BBCM92] Bevilacqua, R., Bini, D., Capovani, M. e Menchi, O. (1992) *Metodi Numerici*, Zanichelli, pagg. 1-5, 8-13, 30-34, 36-45, 50-56, 60-61 (senza dimostrazioni).

[Mon98] Monegato, G. (1998) *Fondamenti di Calcolo Numerico*, Clut, pagg. 1-17.

[NPR01] Naldi, G., Pareschi, L. e Russo, G. (2001) *Introduzione al Calcolo Scientifico*, Mc Graw Hill Ed., pagg. 40-60.

[Hea02] Heath, M.T. (2002) *Scientific Computing*, Mc Graw Hill Ed., pagg. 1-33 (senza dimostrazioni).

[Hi93] Higham, N. J. (1993) *The accuracy of floating point summation*, SIAM J. Sci. Comput. 14, 783-799.

[Ste96] Stewart, G.W. (1996) *Afternotes on Numerical Analysis*, SIAM Ed., pagg. 43-66.