Laboratorio di Algebra Lineare e Analisi Numerica
Laboratorio 1: Aritmetica di macchina e stabilità numerica

Gruppo:
- Mafodda Edoardo S5302507
- Toscano Mattia S5288636
- Trioli Davide S5316731

Esercizio 1

Essendo $c = -b \Leftrightarrow b = -c$, il risultato teorico della somma $a + (b + c) = (a + b) + c$ dovrebbe eguagliare a.

Nei casi $a + (b + c)$ ossia in cui viene eseguita per prima la somma tra questi due ci si trova ad eseguire per prima una cancellazione, riducendo quindi l'errore relativo rispetto alle situazioni $(a + b) + c$; in questi casi il risultato coincide con quello che ci si attende.

```
 - - - Risultato per i = 0 - - -
a = 8; b = 1e+20; c = -1e+20
(a+b)+c : 0
a+(b+c) : 8
errore relativo: -1
 - - - Risultato per i = 1 - - -
a = 80; b = 1e+20; c = -1e+20
(a+b)+c : 0
a+(b+c) : 80
errore relativo: -1
 - - - Risultato per i = 2 - - -
a = 800; b = 1e+20; c = -1e+20
(a+b)+c : 0
a+(b+c) : 800
errore relativo: -1
 - - - Risultato per i = 3 - - -
a = 8000; b = 1e+20; c = -1e+20
(a+b)+c : 0
a+(b+c) : 8000
errore relativo: -1
 - - - Risultato per i = 4 - - -
a = 80000; b = 1e+20; c = -1e+20
(a+b)+c : 81920
a+(b+c) : 80000
errore relativo: 0.024
 - - - Risultato per i = 5 - - -
a = 800000; b = 1e+20; c = -1e+20
(a+b)+c : 802816
a+(b+c) : 800000
errore relativo: 0.00352
 - - - Risultato per i = 6 - - -
a = 8e+06; b = 1e+20; c = -1e+20
(a+b)+c : 7.99539e+06
a+(b+c) : 8e+06
errore relativo: -0.000576
```

Nei casi $a + (b + c)$ invece, siccome la cancellazione viene eseguita per ultima, viene introdotta una quantità di errore relativo nel risultato, dipendente dalla grandezza di a: nelle prime quattro iterazioni dell'esercizio, ossia quando $0 \leq i \leq 3$, a risulta essere così ridotta da non essere nemmeno sentita dalla somma con b, somma che risulta essere di fatto uguale a b a causa dell'aritmetica di macchina e che porta la somma finale con c uguale a 0; nelle ultime due iterazioni a assume valori più compatibili a quelli di b (ma comunque lontani): in questo modo viene "sentita" dalla somma con b, ma porta sempre con sé una certa quantità di errore.

Esercizio 2

Algoritmo 1:

La funzione proposta come approssimante della funzione esponenziale

```
## ALGORITMO 1 ##

:: Funzione corretta e^0.5: 1.64872
Per x = 0.5 e N = 3 la funzione viene approssimata a 1.64583
Errore assoluto: -0.00288794
Errore relativo: -0.00175162

Per x = 0.5 e N = 10 la funzione viene approssimata a 1.64872
Errore assoluto: -1.27627e-11
Errore relativo: -7.74096e-12

Per x = 0.5 e N = 50 la funzione viene approssimata a 1.64872
Errore assoluto: -4.44089e-16
Errore relativo: -2.69354e-16

Per x = 0.5 e N = 100 la funzione viene approssimata a 1.64872
Errore assoluto: -4.44089e-16
Errore relativo: -2.69354e-16

Per x = 0.5 e N = 150 la funzione viene approssimata a 1.64872
Errore assoluto: -4.44089e-16
Errore relativo: -2.69354e-16

:: Funzione corretta e^30: 1.06865e+13
Per x = 30 e N = 3 la funzione viene approssimata a 4981
Errore assoluto: -1.06865e+13
Errore relativo: -1

Per x = 30 e N = 10 la funzione viene approssimata a 2.3883e+08
Errore assoluto: -1.06862e+13
Errore relativo: -0.999978

Per x = 30 e N = 50 la funzione viene approssimata a 1.06833e+13
Errore assoluto: -3.18471e+09
Errore relativo: -0.000298013

Per x = 30 e N = 100 la funzione viene approssimata a 1.06865e+13
Errore assoluto: 0.00390625
Errore relativo: 3.65532e-16

Per x = 30 e N = 150 la funzione viene approssimata a 1.06865e+13
Errore assoluto: 0.00390625
Errore relativo: 3.65532e-16
```

$f(x) = e^x$

effettivamente lavora come previsto: all'aumentare del numero delle iterazioni della sommatoria N la funzione approssima sempre meglio; inoltre da come si nota nei risultati prodotti, l'approssimazione è più fedele per valori di partenza x vicini allo zero (nell'esempio si può vedere come il valore risultante sia più fedele all'"originale" nei casi di $x = \pm 0.5$ e meno nei casi $x = \pm 30$).

Sopra e a destra: output relativi all'algoritmo 1 implementato.

```
:: Funzione corretta e^-0.5: 0.606531
Per x = -0.5 e N = 3 la funzione viene approssimata a 0.604167
Errore assoluto: -0.00236399
Errore relativo: -0.00389757

Per x = -0.5 e N = 10 la funzione viene approssimata a 0.606531
Errore assoluto: 1.17416e-11
Errore relativo: 1.93586e-11

Per x = -0.5 e N = 50 la funzione viene approssimata a 0.606531
Errore assoluto: -1.11022e-16
Errore relativo: -1.83045e-16

Per x = -0.5 e N = 100 la funzione viene approssimata a 0.606531
Errore assoluto: -1.11022e-16
Errore relativo: -1.83045e-16

Per x = -0.5 e N = 150 la funzione viene approssimata a 0.606531
Errore assoluto: -1.11022e-16
Errore relativo: -1.83045e-16

:: Funzione corretta e^-30: 9.35762e-14
Per x = -30 e N = 3 la funzione viene approssimata a -4079
Errore assoluto: -4079
Errore relativo: -4.35901e+16

Per x = -30 e N = 10 la funzione viene approssimata a 1.21255e+08
Errore assoluto: 1.21255e+08
Errore relativo: 1.29579e+21

Per x = -30 e N = 50 la funzione viene approssimata a 8.78229e+08
Errore assoluto: 8.78229e+08
Errore relativo: 9.38517e+21

Per x = -30 e N = 100 la funzione viene approssimata a -4.82085e-06
Errore assoluto: -4.82085e-06
Errore relativo: -5.15179e+07

Per x = -30 e N = 150 la funzione viene approssimata a -4.82086e-06
Errore assoluto: -4.82086e-06
Errore relativo: -5.1518e+07
```

Algoritmo 2:

Notando che $f(-x) = e^{-x} = \frac{1}{e^{x}}$

si può applicare questo metodo di calcolo anche tramite la funzione approssimante. I risultati sono conciliabili con quelli precedentemente ottenuti, ossia la funzione approssima meglio per valori di x vicini allo zero e per numero di iterazioni sempre crescente.

```
## ALGORITMO 2 ##

:: Funzione corretta e^-0.5: 0.606531
Per x = -0.5 e N = 3 la funzione viene approssimata a 0.607595
Errore assoluto: 0.00106428
Errore relativo: 0.0017547

Per x = -0.5 e N = 10 la funzione viene approssimata a 0.606531
Errore assoluto: 4.69513e-12
Errore relativo: 7.74097e-12

Per x = -0.5 e N = 50 la funzione viene approssimata a 0.606531
Errore assoluto: 1.11022e-16
Errore relativo: 1.83045e-16

Per x = -0.5 e N = 100 la funzione viene approssimata a 0.606531
Errore assoluto: 1.11022e-16
Errore relativo: 1.83045e-16

Per x = -0.5 e N = 150 la funzione viene approssimata a 0.606531
Errore assoluto: 1.11022e-16
Errore relativo: 1.83045e-16

:: Funzione corretta e^-30: 9.35762e-14
Per x = -30 e N = 3 la funzione viene approssimata a 0.000200763
Errore assoluto: 0.000200763
Errore relativo: 2.14545e+09

Per x = -30 e N = 10 la funzione viene approssimata a 4.18709e-09
Errore assoluto: 4.18699e-09
Errore relativo: 44744.2

Per x = -30 e N = 50 la funzione viene approssimata a 9.36041e-14
Errore assoluto: 2.78952e-17
Errore relativo: 0.000298102

Per x = -30 e N = 100 la funzione viene approssimata a 9.35762e-14
Errore assoluto: -3.78653e-29
Errore relativo: -4.04647e-16

Per x = -30 e N = 150 la funzione viene approssimata a 9.35762e-14
Errore assoluto: -3.78653e-29
Errore relativo: -4.04647e-16
```

A sinistra: output relativo all'algoritmo 2 implementato.

Esercizio 3

Per determinare il valore eps, ossia il massimo valore d per cui $1 + 2^{-d} > 1$ in aritmetica di macchina, si utilizza un ciclo in modo da incrementare d in maniera lineare ed eseguire la predetta somma fino a che questa non avrà come risultato uno: in questo modo si riesce a capire quando il valore 2^{-d} è talmente piccolo da non essere più sentito. Il valore quindi minore possibile ancora apprezzabile sarà $2^{-(d-1)}$. Nell'esercizio, d viene calcolato "all'indietro" ma in modo analogo – cioè calcolando $\delta = -d$, e quindi

$eps = 2^{\delta+1} = 2^{-d+1} = 2^{-(d-1)}$;

inoltre il test viene effettuato sia a precisione singola (*float*) che a precisione doppia (*double*) apprezzando i due valori differenti nei rispettivi casi.

```
valore : -1 : 1.5
valore : -2 : 1.25
valore : -3 : 1.125
valore : -4 : 1.0625
valore : -5 : 1.03125
valore : -6 : 1.01562
valore : -7 : 1.00781
valore : -8 : 1.00391
valore : -9 : 1.00195
valore : -10 : 1.00098
valore : -11 : 1.00049
valore : -12 : 1.00024
valore : -13 : 1.00012
valore : -14 : 1.00006
valore : -15 : 1.00003
valore : -16 : 1.00002
valore : -17 : 1.00001
valore : -18 : 1
valore : -19 : 1
valore : -20 : 1
valore : -21 : 1
valore : -22 : 1
valore : -23 : 1
valore : -24 : 1
valore : -25 : 1
valore : -26 : 1
valore : -27 : 1
valore : -28 : 1
valore : -29 : 1
valore : -30 : 1
valore : -31 : 1
valore : -32 : 1
valore : -33 : 1
valore : -34 : 1
valore : -35 : 1
valore : -36 : 1
valore : -37 : 1
valore : -38 : 1
valore : -39 : 1
valore : -40 : 1
valore : -41 : 1
valore : -42 : 1
valore : -43 : 1
valore : -44 : 1
valore : -45 : 1
valore : -46 : 1
valore : -47 : 1
valore : -48 : 1
valore : -49 : 1
valore : -50 : 1
valore : -51 : 1
valore : -52 : 1
valore : -53 : 1
 ########## eps (double) = 2^(-52) = 2.22045e-16
```

A sinistra: output relativo al caso *double*, con $eps = 2^{-52}$.
Sotto, output relativo al caso float, con $eps = 2^{-23}$.

```
valore : -1 : 1.5
valore : -2 : 1.25
valore : -3 : 1.125
valore : -4 : 1.0625
valore : -5 : 1.03125
valore : -6 : 1.01562
valore : -7 : 1.00781
valore : -8 : 1.00391
valore : -9 : 1.00195
valore : -10 : 1.00098
valore : -11 : 1.00049
valore : -12 : 1.00024
valore : -13 : 1.00012
valore : -14 : 1.00006
valore : -15 : 1.00003
valore : -16 : 1.00002
valore : -17 : 1.00001
valore : -18 : 1
valore : -19 : 1
valore : -20 : 1
valore : -21 : 1
valore : -22 : 1
valore : -23 : 1
valore : -24 : 1
 ########## eps (float) = 2^(-23) = 1.19209e-07
```