

# Universidade de São Paulo

## Instituto de Física de São Carlos

### **Lista 1**

Pedro Calligaris Delbem 5255417

Professor: Attilio Cucchieri

Março de 2025

# Sumário

1	Exercício 1	2
2	Exercício 2	2
3	Exercício 3	3
4	Exercício 4	3
5	Exercício 5	4
6	Exercício 6	5

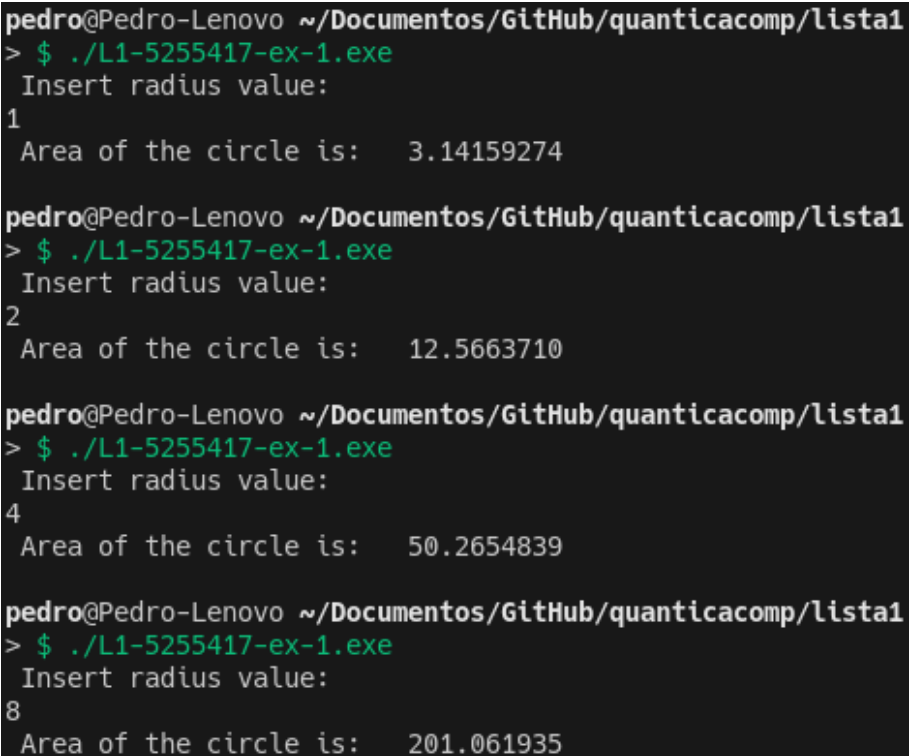
# 1 Exercício 1

Tarefa: Calcular a área de um círculo. Opções: entrada e saída usando teclado e monitor; entrada e saída usando arquivos; calcular a área em uma subrotina.

O código foi compilado com o comando:

```
gfortran L1-5255417-ex-1.f90 -o L1-5255417-ex-1.exe
```

Resultados:



```
pedro@Pedro-Lenovo ~/Documentos/GitHub/quanticacomp/lista1
> $ ./L1-5255417-ex-1.exe
Insert radius value:
1
Area of the circle is: 3.14159274

pedro@Pedro-Lenovo ~/Documentos/GitHub/quanticacomp/lista1
> $ ./L1-5255417-ex-1.exe
Insert radius value:
2
Area of the circle is: 12.5663710

pedro@Pedro-Lenovo ~/Documentos/GitHub/quanticacomp/lista1
> $ ./L1-5255417-ex-1.exe
Insert radius value:
4
Area of the circle is: 50.2654839

pedro@Pedro-Lenovo ~/Documentos/GitHub/quanticacomp/lista1
> $ ./L1-5255417-ex-1.exe
Insert radius value:
8
Area of the circle is: 201.061935
```

Figura 1: Valores de área para os raio de 1, 2, 4 e 8.

A área para o raio = 1 corresponde ao valor de  $\pi$ , de acordo com o esperado. Dobrando o valor do raio espera-se, então, que o valor da área quadruplique e isto foi o que se obteve.

# 2 Exercício 2

Tarefa: Testar overflow e underflow em precisão simples e dupla.

O código foi compilado com o comando:

```
gfortran L1-5255417-ex-2.f90 -o L1-5255417-ex-2.exe
```

Resultados:

```
pedro@Pedro-Lenovo ~/Documentos/GitHub/quanticaomp/lista1
> $ ./L1-5255417-ex-2.exe
Overflow simple real is:      1.000000007E+38
Overflow double real is:     9.999999999999981E+307
Underflow simple real is:    1.40129846E-45
Underflow double real is:    9.8813129168249309E-324
```

Figura 2: Valores de overflow e underflow para precisão simples e dupla.

Nota-se que os valores obtidos são coerentes com o esperado.

### 3 Exercício 3

Tarefa: Achar a precisão do computador, i.e., o maior número positivo tal que  $1 + \epsilon = 1$ , usando precisão simples e dupla.

O código foi compilado com o comando:

```
gfortran L1-5255417-ex-3.f90 -o L1-5255417-ex-3.exe
```

Resultados:

```
pedro@Pedro-Lenovo ~/Documentos/GitHub/quanticaomp/lista1
> $ ./L1-5255417-ex-3.exe
Machine precision for simple precision is:  5.96046448E-08
Machine precision for double precision is:  1.1102230246251565E-016
```

Figura 3: Valores de  $\epsilon$  para precisão simples e dupla.

Nota-se que os valores obtidos correspondem aos valores esperados.

### 4 Exercício 4

Tarefa: Calcular

$$e^{-x} = 1 - x + x^2/2! - x^3/3! + \dots \quad (1)$$

para  $x = 0.1, 1, 10, 100$  e  $1000$  com um erro menor do que  $10^{-8}$ . Problema: quando truncar a série? É preciso calcular o fatorial explicitamente? Comparar o valor obtido usando a série com o resultado exato.

O código foi compilado com o comando:

```
gfortran L1-5255417-ex-4.f90 -o L1-5255417-ex-4.exe
```

Resultados:

```
x = 0.100000001490116119384765625000000000
Computed e^(- 0.100000001490116119384765625000000000 ) = 0.904837416687401464651781741797552441
Real e^(- 0.100000001490116119384765625000000000 ) = 0.904837416687646752130945377714961247
x = 1.000000000000000000000000000000000000
Computed e^(- 1.000000000000000000000000000000000000 ) = 0.367879441160691160691160691160691189
Real e^(- 1.000000000000000000000000000000000000 ) = 0.367879441171442321595523770161460873
x = 10.000000000000000000000000000000000000
Computed e^(- 10.000000000000000000000000000000000000 ) = 4.53999297597373466218554931053598531E-0005
Real e^(- 10.000000000000000000000000000000000000 ) = 4.53999297624848515355915155605506104E-0005
x = 100.000000000000000000000000000000000000
Computed e^(- 100.000000000000000000000000000000000000 ) = 95487007.5342923401240560047228340998
Real e^(- 100.000000000000000000000000000000000000 ) = 3.72007597602083596295969580386311846E-0044
x = 1000.000000000000000000000000000000000000
Computed e^(- 1000.000000000000000000000000000000000000 ) = 1.06968282685368972169321924546937285E+0399
Real e^(- 1000.000000000000000000000000000000000000 ) = 5.07595889754945676529180947957433714E-0435
```

Figura 4: Valores de  $e^{-x}$  para  $x = 0.1, 1, 10, 100$  e  $1000$ .

Por fim, percebe-se que o resultado esperado foi obtido para  $x = 0.1, 1$  e  $10$ . Para  $x = 100$  e  $1000$  o resultado obtido diverge do esperado, o que se deve ao fato de que os termos somados se tornam muito pequenos ao ponto de serem menores que o Machine Precision.

## 5 Exercício 5

Tarefa: Considerar a somatória

$$\Sigma(N) = \sum_{n=1}^{2N} (-1)^n \frac{n}{n+1} = - \sum_{n=1}^N \frac{2n-1}{2n} + \sum_{n=1}^N \frac{2n}{2n+1} = \sum_{n=1}^N \frac{1}{2n(2n+1)} \quad (2)$$

e calcular  $\Sigma(N)$  para  $N = 1, 2, \dots, 10^6$  usando as três fórmulas acima. Comparar os resultados usando precisão simples.

O código foi compilado com o comando:

```
gfortran L1-5255417-ex-5.f90 -o L1-5255417-ex-5.exe
```

Resultados:

```
pedro@Pedro-Lenovo ~/Documentos/GitHub/quanticacomp/lista1
> $ ./L1-5255417-ex-5.exe
      1 -0.333333313      0.166666687      0.166666672
      2  0.883333385      0.216666698      0.216666669
      3 -0.509523690      0.240476370      0.240476191
      4  1.05436516      0.254365206      0.254365087
      5 -0.569877207      0.263456345      0.263455987
      6  1.12700903      0.269866943      0.269866258
      7 -0.600371778      0.274629116      0.274628162
      9 -0.618771255      0.281229973      0.281228602
     10  1.19270074      0.283611298      0.283609569
    100  1.29447055      0.304382324      0.304371417
   1000  1.30560207      0.306640625      0.306603163
  10000  1.30670857      0.306640625      0.306800693
100000  1.30684614      0.304687500      0.306800693
1000000 1.30686092      0.312500000      0.306800693
```

Figura 5: Valores de  $\Sigma(N)$  para  $N = 1, 2, \dots, 10^6$  usando precisão simples.

Nota-se que a primeira série demora é mais instável do que as demais. Além disso, a terceira série se mostra mais estável que a segunda - sendo então a melhor versão.

## 6 Exercício 6

Tarefa: Estudar numericamente o erro da aproximação

$$e^{-x} \approx \sum_{n=0}^N \frac{(-x)^n}{n!} \quad (3)$$

em função de  $N$ , para diferentes valores de  $x$ . Sugestão: faça um gráfico do erro em função de  $N$ . O que acontece quando  $e^{-x}$  é calculado usando a série  $e^x = \sum_{n=0}^N \frac{x^n}{n!}$  e, depois, calculando  $1/e^x$ ?

O código foi compilado com o comando:

```
gfortran L1-5255417-ex-6.f90 -o L1-5255417-ex-6.exe
```

Resultados:

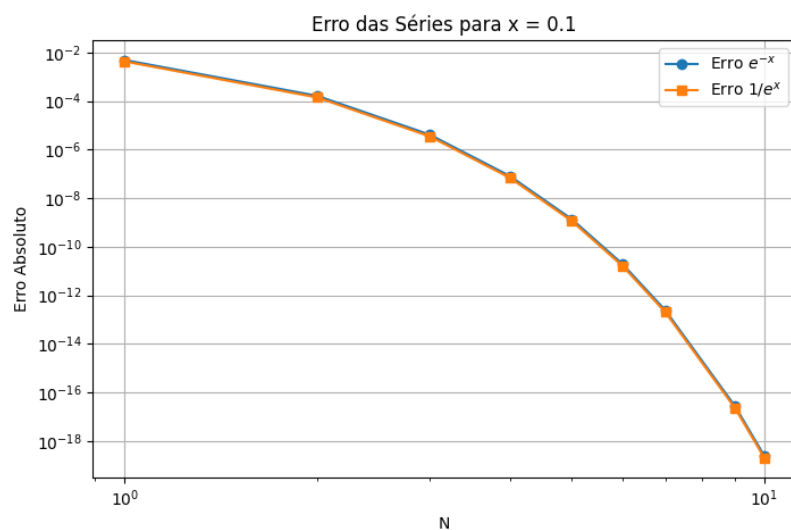


Figura 6: Gráfico do erro em função de N para  $x = 0.100000001$ .

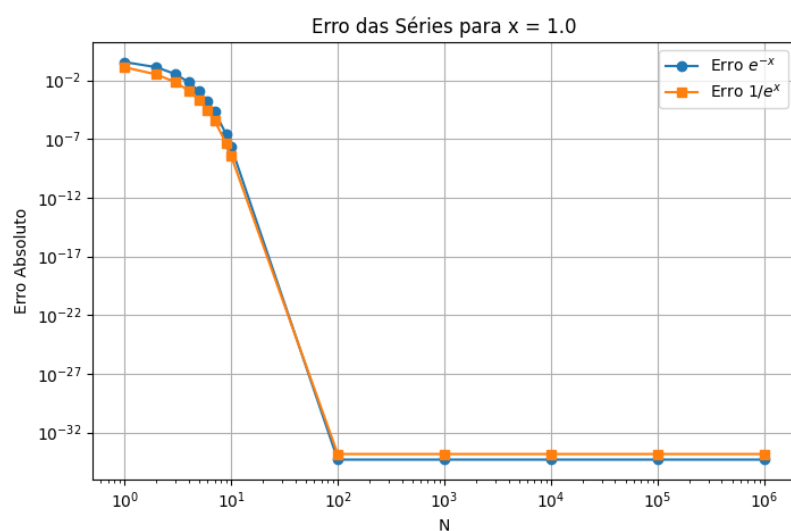


Figura 7: Gráfico do erro em função de N para  $x = 1$ .

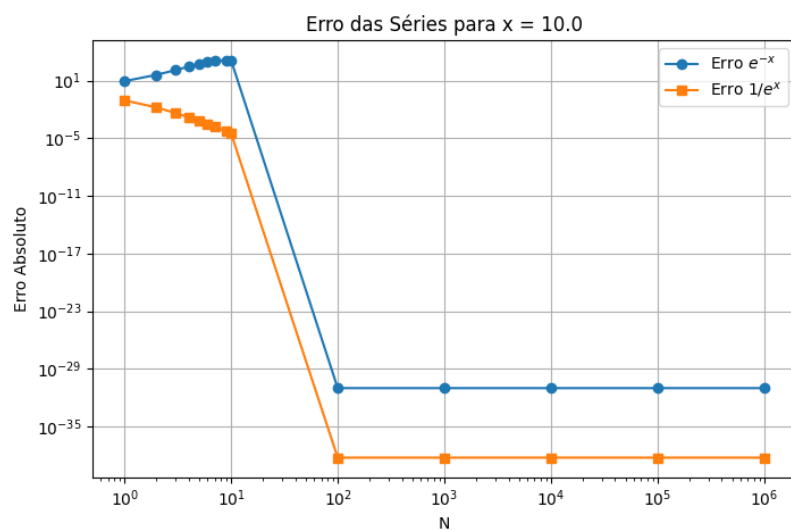


Figura 8: Gráfico do erro em função de N para  $x = 10$ .

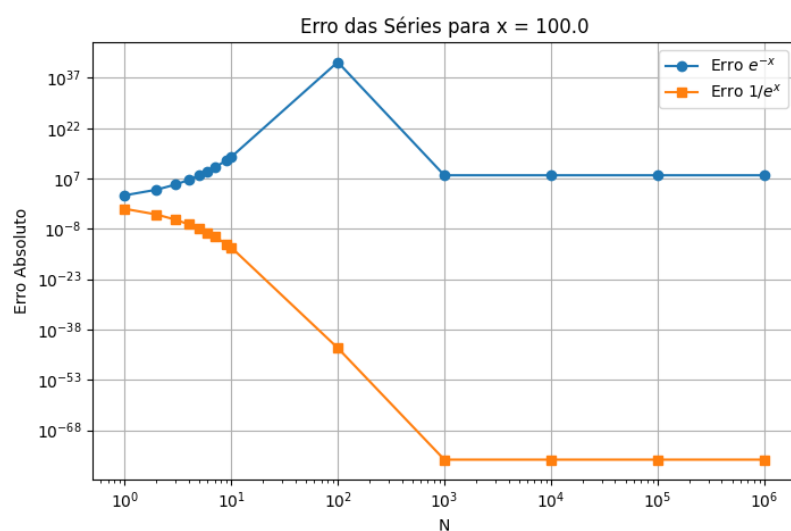


Figura 9: Gráfico do erro em função de N para  $x = 100$ .



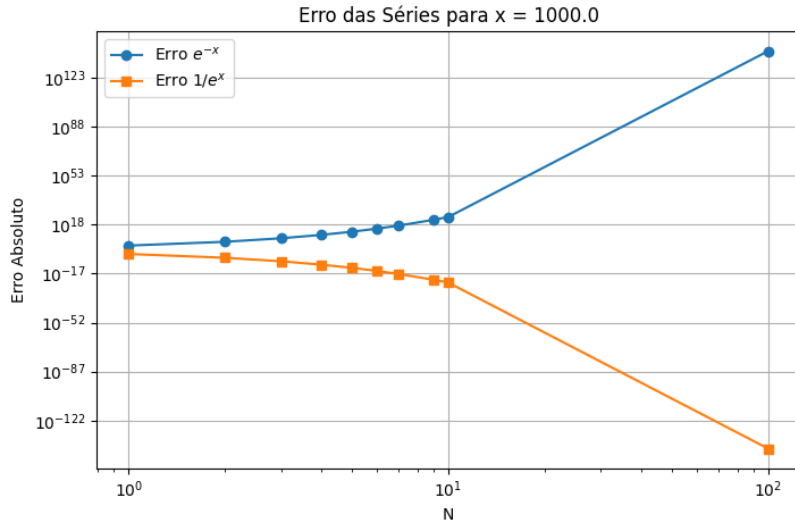


Figura 10: Gráfico do erro em função de N para x = 1000.

Percebe-se - claramente - que calcular a série de  $e^x$  e fazer  $1/e^x$  o resultado é mais preciso do que calcular a série, diretamente, de  $e^{-x}$ . Além disso, é notório que ao calcular a série de  $e^{-x}$  diretamente para valores de N muito grande o resultado apresenta uma divergência muito grande do resultado real o que se deve ao fato de que os termos somados se tornam muito pequenos ao ponto de serem menores que o Machine Precision.