

$$(d) (0.02)_{F!} = \frac{0}{2!} + \frac{2}{3!} = (\frac{2}{6})_{10} = (\frac{1}{3})_{10} = (\mathbf{0.333333\dots})_{10}$$

$$(e) (0.113)_{F!} = \frac{1}{2!} + \frac{1}{3!} + \frac{3}{4!} = (\frac{19}{24})_{10} = (\mathbf{0.791\bar{6}\dots})_{10}$$

$$(f) (321.123)_{F!} = 3 \cdot 3! + 2 \cdot 2! + 1 \cdot 1! + \frac{1}{2!} + \frac{2}{3!} + \frac{3}{4!} = (\frac{575}{24})_{10} = (\mathbf{23.958\bar{3}})_{10}$$

$$4) (a) (10111.1101)_2 = (0001\ 0111.1101)_2 = (17.D)_{16} = 1 \cdot 16^1 + 7 \cdot 16^0 + 13 \cdot 16^{-1} = (\frac{381}{16})_{10} = (\mathbf{23.8125})_{10}$$

$$(b) (BD.0E)_{16} = (1011\ 1101.0000\ 1110)_2 = 1 \cdot 2^7 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^0 + 1 \cdot 2^{-5} + 1 \cdot 2^{-6} + 1 \cdot 2^{-7} = (\frac{24199}{128})_{10} = (\mathbf{189.0546875})_{10}$$

$$(c) (41.1)_{10} = (?)_2 = (?)_{16}$$

$$(41)_{10} = (?)_2 \longrightarrow \begin{array}{r} 41 \left| \begin{array}{l} 2 \\ 20 \end{array} \right. \begin{array}{l} 2 \\ 10 \end{array} \left| \begin{array}{l} 2 \\ 5 \end{array} \right. \begin{array}{l} 2 \\ 2 \end{array} \left| \begin{array}{l} 2 \\ 1 \end{array} \right. \begin{array}{l} 2 \\ 0 \end{array} \\ \text{1} \quad \text{0} \quad \text{0} \quad \text{1} \quad \text{0} \quad \text{1} \quad \text{0} \end{array}$$

←

$$(41)_{10} = (101001)_2 = 1 \cdot 2^5 + 1 \cdot 2^3 + 1 \cdot 2^0$$

$$(0.1)_{10} = (?)_2 \longrightarrow$$

$$\begin{array}{cccccccccccc} 0.1 & 0.2 & 0.4 & 0.8 & 0.6 & 0.2 & 0.4 & 0.8 & 0.6 & 0.2 & \\ \times 2 & \times 2 & \times 2 & \times 2 & \times 2 & \times 2 & \times 2 & \times 2 & \times 2 & \times 2 & \dots \\ \text{0.2} & \text{0.4} & \text{0.8} & \text{1.6} & \text{1.2} & \text{0.4} & \text{0.8} & \text{1.6} & \text{1.2} & \text{0.4} & \end{array}$$

$$(0.1)_{10} = (0.00011001100110011\dots)_2$$

Logo, $(41.1)_{10} = (101001.\overline{00011})_2 = (0010\ 1001.0001\ \overline{1001})_2 = (\mathbf{29.19})_{16}$. Haverá perda de dígitos significativos.

$$5) F(\beta, t, I, S) \longrightarrow F(2, 3, -3, +3) \longrightarrow \boxed{s_1 \mid d_1 \mid d_2 \mid d_3 \mid s_2 \mid e_1 \mid e_2 \mid e_3}$$

$$(a) NM = (\beta - 1) \cdot \beta^{t-1} = (2 - 1) \cdot 2^{3-1} = \mathbf{4}$$

$$(b) NE = S - I + 1 = 3 - (-3) + 1 = \mathbf{7}$$

$$(c) NR = 2 \cdot NM \cdot NE + 1 = \mathbf{57}$$

$$(d) \boxed{0 \mid 1 \mid 0 \mid 0 \mid 1 \mid 1 \mid 0 \mid 0} \longrightarrow m.p. = (0.1)_2 \cdot (2^{-3})_{10} = (2^{-1})_{10} \cdot (2^{-3})_{10} = (0.0625)_{10}$$

Logo, a região de *underflow* é $\{x \in \mathbb{R} \mid -(\mathbf{0.0625})_{10} < x < (\mathbf{0.0625})_{10}\}$.

$$\boxed{0 \mid 1 \mid 1 \mid 1 \mid 0 \mid 1 \mid 0 \mid 0} \longrightarrow M.P. = (0.111)_2 \cdot (2^3)_{10} = (0.875)_{10} \cdot (2^3)_{10} = (7)_{10}$$

Logo, a região de *overflow* é $\{x \in \mathbb{R} \mid x < -(\mathbf{7})_{10} \cup x > (\mathbf{7})_{10}\}$.

$$(e) \text{ A precisão decimal equivalente é aproximadamente } \log_{10}(2^3) = 3 \cdot \log_{10}(2) \approx \mathbf{0.90308998699}.$$

$$(f) F(2, 3, 0, 6) \text{ (polarização } p = +3)$$

$$6) F(\beta, t, I, S) \longrightarrow F(2, 3, 0, +7) \longrightarrow \boxed{s_1 \mid d_1 \mid d_2 \mid d_3 \mid 0 \mid e_1 \mid e_2 \mid e_3}$$

$$(a) NM = (\beta - 1) \cdot \beta^{t-1} = (2 - 1) \cdot 2^{3-1} = \mathbf{4}$$

$$(b) NE = S - I + 1 = 7 - 0 + 1 = \mathbf{8}$$

$$(c) NR = 2 \cdot NM \cdot NE + 1 = \mathbf{65}$$

$$(d) \boxed{0 \mid 1 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0} \longrightarrow m.p. = (0.1)_2 \cdot (2^0)_{10} = (2^{-1})_{10} \cdot (1)_{10} = (\mathbf{0.5})_{10}$$

Logo, a região de *underflow* é $\{x \in \mathbb{R} \mid -(\mathbf{0.5})_{10} < x < (\mathbf{0.5})_{10}\}$.

$$\boxed{0 \mid 1 \mid 1 \mid 1 \mid 0 \mid 1 \mid 1 \mid 1} \longrightarrow m.p. = (0.111)_2 \cdot (2^7)_{10} = (0.875)_{10} \cdot (2^7)_{10} = (\mathbf{112})_{10}$$

Logo, a região de *overflow* é $\{x \in \mathbb{R} \mid x < -(\mathbf{112})_{10} \cup x > (\mathbf{112})_{10}\}$.

$$(e) \text{ A precisão decimal equivalente é aproximadamente } \log_{10}(2^3) = 3 \cdot \log_{10}(2) \approx \mathbf{0.90308998699}.$$

7) O formato de ponto flutuante de precisão dupla tem 1 bit de sinal, 11 bits para expoente e 52 bits explícitos para decimais (o 1º bit é implícito por conta do armazenamento do número de acordo com o padrão IEEE 754).

Também foi decidido neste padrão **facilitar** o armazenamento do expoente. Assim, um conceito chamado *exponent bias*, calculado por $2^{k-1} - 1$ onde k é o número de bits do expoente, foi criado com esta função. Este *bias* é essencial para a codificação do expoente como um valor **unsigned**. Para obter o expoente verdadeiro, subtrai-se o *bias* do expoente representado.

Assim sendo, $bias = 2^{11-1} - 1 = 1023$. $e_{min} = 1 - 1023 = -1022$ e $e_{max} = 2046 - 1023 = 1023$. (Os expoentes 0 e 2047 são reservados.)

Portanto, o menor número passível de ser representado com precisão dupla é $2^{-1022} \approx \mathbf{2.225 \cdot 10^{-308}}$, e o maior número é 2^{1023} com todos os bits da mantissa ativados, ou seja, $2^{1023} \cdot (2 - 2^{-52}) \approx \mathbf{1.797 \cdot 10^{308}}$. Às custas da perda de precisão, existem números que vão abaixo da fronteira de *underflow*, chamados de **subnormais**.

- O código acima simula os cálculos com precisão dupla nos dois tipos de equação quadrática, além de imprimir o resultado arredondado para o operador aritmético $F(10, 4, -99, +99)$ e os resultados reais.
- Observa-se que, onde não existem subtrações internas para calcular o resultado final, o erro relativo mostra-se muito inferior, assim gerando resultado mais confiáveis. Utilizando precisão dupla, os erros tornam-se irrelevantes entre si – a diferença entre métodos é da ordem de $1 \cdot 10^{-14}$.
- Saída do programa:

11) (a) • Erro absoluto: $|VA - VE| = |1102.345 - 1100.9| = \mathbf{1.445}$
 • Erro relativo: $\left| \frac{VA-VE}{VE} \right| = \left| \frac{1.445}{1100.9} \right| \approx \mathbf{0.00131}$
 • Percentual: $0.00131 \cdot 100 = \mathbf{0.131\%}$

(b) • Erro absoluto: $|VA - VE| = |0.01245 - 0.0119| = \mathbf{0.00055}$
 • Erro relativo: $\left| \frac{VA-VE}{VE} \right| = \left| \frac{0.00055}{0.0119} \right| \approx \mathbf{0.04622}$
 • Percentual: $0.04622 \cdot 100 = \mathbf{4.622\%}$

$$\begin{array}{cccccccccc} 0.05 & 0.1 & 0.2 & 0.4 & 0.8 & 0.6 & 0.2 & 0.4 & 0.8 & 0.6 \\ \times 2 & \times 2 & \times 2 & \times 2 & \times 2 & \times 2 & \times 2 & \times 2 & \times 2 & \times 2 & \dots \\ \color{red}{0.1} & \color{red}{0.2} & \color{red}{0.4} & \color{red}{0.8} & \color{red}{1.6} & \color{red}{1.2} & \color{red}{0.4} & \color{red}{0.8} & \color{red}{1.6} & \color{red}{1.2} & \dots \end{array}$$

$$-(0.05)_{10} = -(0.0000\overline{11})_2 = 1 \cdot 2^{-5} \cdot -(0.\overline{1001})_2$$

$$(-10.05)_{10} = (1010.0000\overline{1100})_2 = (1.0100000\overline{1100})_2 \cdot 2^3$$

(b) Convertendo novamente o número para decimal, tem-se $VA = (-10.050000190734863)_{10}$.
O percentual de erro é igual a $\left| \frac{VA - VE}{VE} \right| \cdot 100 \approx \mathbf{1.897856 \cdot 10^{-6}\%}$.

$$e = 0, f \neq 0 \longrightarrow v = (-1)^s \cdot \left(\sum_{i=1}^{23} b_{23-i} 2^{-i} \right) \cdot 2^{e-126} = (-1)^1 \cdot (2^{-1} + 2^{-23}) \cdot 2^{-126} \approx -5.877473 \cdot 10^{-39}$$

4

$f = (01101011100001001110111)_2 + (1)_2 = (01101011100001001111000)_2$ (primeiros 23 bits + arredondamento)

Resultado final:

| | | |
|---|----------|-------------------------|
| 1 | 11111010 | 01101011100001001111000 |
|---|----------|-------------------------|

- (b) Convertendo novamente o número para decimal, tem-se $VA = (-1.51000004197986229847484292417978368 \cdot 10^{37})_{10}$.

O percentual de erro é igual a $\left| \frac{VA-VE}{VE} \right| \cdot 100 \approx \mathbf{2.7801 \cdot 10^{-6}\%}$.

- 15) (a) $(-1.1 \cdot 10^{-41})_{10} = (-1)^1 \cdot 2^{-126} \cdot (0.f)_2$ com $e \leq 0$, pois $\log_2(1.1 \cdot 10^{-41}) \approx -136$, o que implica em $e = -136 + 127 = -9 < 0 \rightarrow$ número subnormal)

Então, para descobrir a mantissa f , calcula-se $\frac{1.1 \cdot 10^{-41}}{2^{-126}}$ e multiplica-se o decimal para encontrar seu valor binário. Por fim, $(-1.1 \cdot 10^{-41})_{10} = (-1.111010101001110111001100000 \dots)_2 \cdot 2^{-137}$.

É necessário mover o decimal para que o número possa ser representado com $e = 0$. Então, $(-1.111010101001110111001100000 \dots)_2 \cdot 2^{-137} = (-0.00000000001111010101001 \dots)_2 \cdot 2^{-126}$.

Resultado final:

| | | |
|---|----------|-------------------------|
| 1 | 00000000 | 00000000001111010101010 |
|---|----------|-------------------------|

- (b) Convertendo novamente o número para decimal, tem-se $VA = -1.1000192944949814 \cdot 10^{-41}$

O percentual de erro é igual a $\left| \frac{VA-VE}{VE} \right| \cdot 100 = \mathbf{1.754044998309 \cdot 10^{-3}\%}$.

- 16) Caso seja viável calcular o resultado desejado à mão, ou utilizando ferramentas de maior precisão, é possível verificar o erro relativo sem maiores dificuldades. Porém, na maioria dos casos isso não acontece, e é necessário observar como o computador se comporta nestes casos. Esta aritmética de ponto flutuante é uma grande área de estudo do cálculo numérico.

17, 18)

```
from functools import reduce
from math import factorial as fact
from numpy import float32 as single
```

```
n = 4
x = -.111
desired = 0.894938748929031
```

```
for y in [single(x), x]:
    obtained = reduce(lambda w, i: w + (y**i)/fact(i), range(n), 0)
    print("{:.65f}%".format(abs((desired - obtained) / desired) * 100))
```

- O decimal desejado foi retirado do [Wolfram Alpha](#), para comparação com os resultados calculados pelo programa em variáveis de precisão simples e dupla, respectivamente. O erro da variável de precisão simples é maior pela quantidade reduzida de bits disponíveis para armazenar a parte decimal do resultado.
- Novamente, a função *reduce* é utilizada para simular o cálculo da série de Maclaurin com os argumentos fornecidos.
- Saída do programa:

```
0.00069152325516942810771509053680006218201015144586563110351562500%
0.00069138016857893662848316695956896182906348258256912231445312500%
```