

# MAC0210 - Laboratório de Métodos Numéricos

## Exercício-programa 1

Vítor Kei Taira Tamada - 8516250

André Ferrari Moukarzel - 9298166

### 1 Parte 1: Aritmética de ponto flutuante

**Questão 1 (3.11): Dado o sistema de ponto flutuante com base 2:**

$$\begin{aligned}x &= \pm S \times 2^E \\ \text{sendo que} \\ S &= (0.1b_2b_3b_4\dots) \\ -128 < E < 127\end{aligned}$$

**a) Qual é o maior número de ponto flutuante desse sistema?**

R) O maior número de ponto flutuante que esse sistema é capaz de representar é:

$$+(0.11111111111111111111111111111111) \times 2^{126}$$

que é equivalente a

$$+11111111111111111111111111111111 \times 2^{102}$$

Como 11111111111111111111111111111111 equivale a 16777215 em base 10, temos que o maior número que esse sistema consegue representar é:

$$16777215 \times 2^{102}$$

**b) Qual é o menor número de ponto flutuante positivo desse sistema?**

R) O menor número de ponto flutuante positivo que esse sistema é capaz de representar é:

$$+(0.10000000000000000000000000000000) \times 2^{-127}$$

Como 0.10000000000000000000000000000000 equivale a 0.5 em base 10, temos que o menor número positivo que esse sistema consegue representar é:

$$0.5 \times 2^{-127}$$

**c) Qual é o menor inteiro positivo que não é exatamente representável nesse sistema?**

R) O menor número positivo não representável no sistema é o número com o menor S dentro do intervalo e o menor E acima do intervalo. Ou seja,

$$\begin{aligned}S &= 0.10000000000000000000000000000000 \\ E &= 127\end{aligned}$$

Logo, o menor inteiro não representável nesse sistema é o

$$x = 0.5 \times 2^{127}$$

■

**Questão 2 (5.1): Qual é a representação do número  $\frac{1}{10}$  no formato IEEE single para cada um dos quatro modos de arredondamento? E para os números  $1 + 2^{-25}$  e  $2^{130}$ ?**

R) As representações no formato IEEE single em cada um dos quatro modos de arredondamento são as seguintes:

[Para  $\frac{1}{10}$ ]

- Aproximação para  $+\infty$ :  
 $(0.00011001100110011001101)_2 \times 2^0$
- Aproximação para  $-\infty$ :  
 $(0.00011001100110011001100)_2 \times 2^0$
- Aproximação para 0:  
 $(0.00011001100110011001100)_2 \times 2^0$
- Aproximação para o mais próximo:  
 $(0.00011001100110011001101)_2 \times 2^0$

[Para  $1 + 2^{-25}$ , sabendo que a mantissa tem 23 bits]

- Aproximação para  $+\infty$ :  
 $(0.100000000000000000000001)_2 \times 2^1$
- Aproximação para  $-\infty$ :  
 $(0.100000000000000000000000)_2 \times 2^1$
- Aproximação para 0:  
 $(0.100000000000000000000000)_2 \times 2^1$
- Aproximação para o mais próximo:  
 $(0.100000000000000000000000)_2 \times 2^1$

[Para  $2^{130}$ ]

*Uma vez que o formato IEEE single não consegue representar números maior do que  $2^{128}$  (sendo que esse limite é reservado para representar  $\infty$ ), o número  $2^{130}$  não possui aproximação nesse sistema*

■

**Questão 3 (6.4): Qual é o maior número de ponto flutuante  $x$  tal que  $1 \oplus x$  é exatamente 1, assumindo que o formato usado é IEEE single e modo de arredondamento para o mais próximo? E se o formato for IEEE double?**

R) Dada uma mantissa binária com  $t$  bits de precisão, um número  $x \leq 2^{-(t+2)}$  é arredondado para baixo se utilizar arredondamento para mais próximo. Portanto, o número  $x$  é:

- Para IEEE single,  $2^{-25}$
- Para IEEE double,  $2^{-54}$

■

**Questão 4 (6.8): Em aritmética exata, a soma é um operador comutativo e associativo. O operador de soma de ponto flutuante é comutativo? E associativo? Explique.**

R)

- Associatividade:

Prova por contradição: Supondo que o operador de soma de ponto flutuante é associativo:

$$\begin{aligned} a &= 1 + 2^{-25} \\ b &= 1 \end{aligned}$$

Com o arredondamento para o mais próximo, temos que:

$$((a + a) + b = 3 + 2^{-23}) \neq ((a + b) + a = 3)$$

Por contradição, conclui-se que a soma de ponto flutuante não é associativa.

- Comutatividade:

Seja  $fl(x) = x + x \times e_x$  o valor de  $x$  em ponto flutuante - ou seja,  $e_x$  é o erro causado pela limitação do computador;

$$\begin{aligned} fl(a) + fl(b) &= fl(b) + fl(a) \\ a + a \times e_a + b + b \times e_b &= b + b \times e_b + a + a \times e_a \end{aligned}$$

Seja  $e_{ab} = a \times e_a + b \times e_b$ , temos então que:

$$a + b + e_{ab} = b + a + e_{ab}$$

Logo, conclui-se que a soma de ponto flutuante é comutativa

■

## 2 Parte 2: Método de Newton

**Método de Newton:** O método de Newton foi implementado na função `newton()` segundo o indicado pelo livro com as seguintes checagens:

- Uma iteração verifica se o valor absoluto de  $f(x)$  atual é menor do que a tolerância pré determinada. Em caso positivo, retorna  $|f(x_0)|$ ;

```
root = newton(f, x0)
err = 0.000001;
valor = polyval(f, x0);
if (abs(valor) < err)
    root = x0;
    return;
endif
...
```

- em seguida, verifica se  $f'(x) = 0$ , pois o método realiza uma divisão com a derivada no denominador. Logo, essa checagem serve para evitar divisões por zero;

```
...
der = polyval(polyder(f), x0);
if (der == 0)
    root = NaN;
    return;
endif
...
```

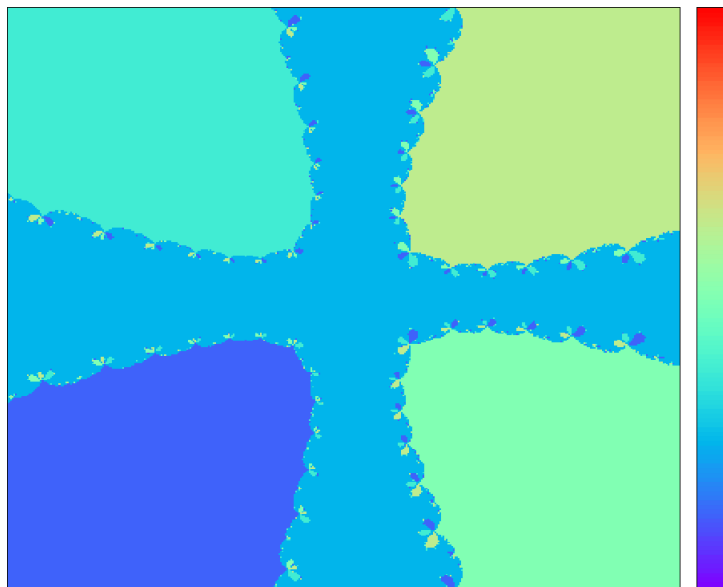
- caso o método ultrapasse 100 iterações, considera-se que o ponto dado não converge para uma das raízes da função, retornando NaN.

```
for iter = 0 : 100
    ...
endfor
```

### Exemplos de saída do programa

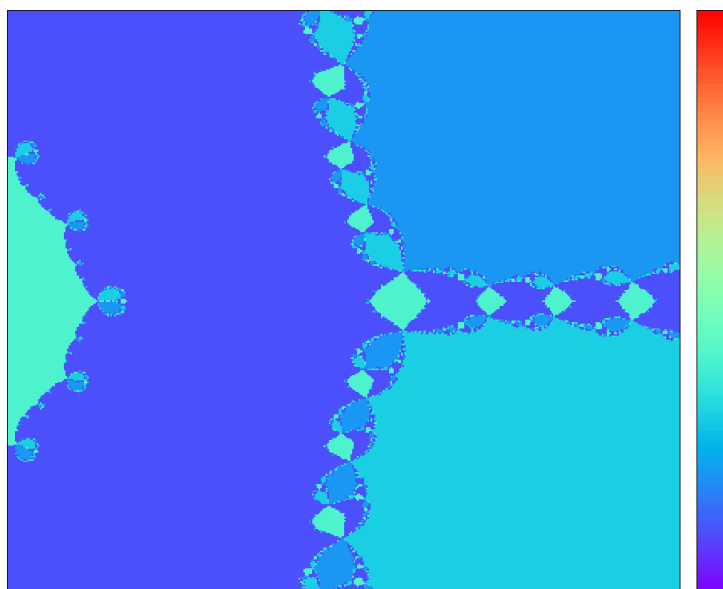
O enunciado foi feito com os limites  $[-2, 2]$  para o parte real e  $[-2i, 2i]$  para a parte imaginária.

Os exemplos gerados estão dentro dos intervalos  $[-4, 4]$  para a parte real e  $[-4i, 4i]$  para a parte imaginária e intervalo  $[0:8]$  no arquivo `plot_basins.gp`



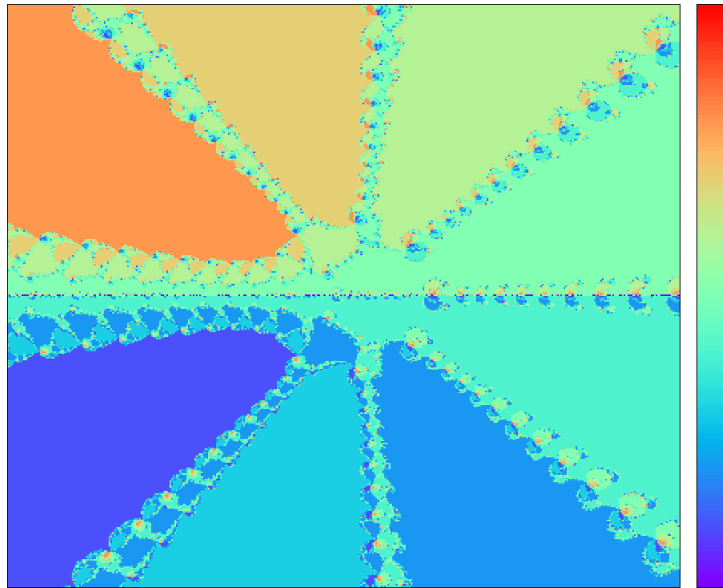
$$-\frac{9}{7}x^5 + \frac{7}{11}x^4 + \frac{3}{17}x^3 - 5x - 1$$

n = 400



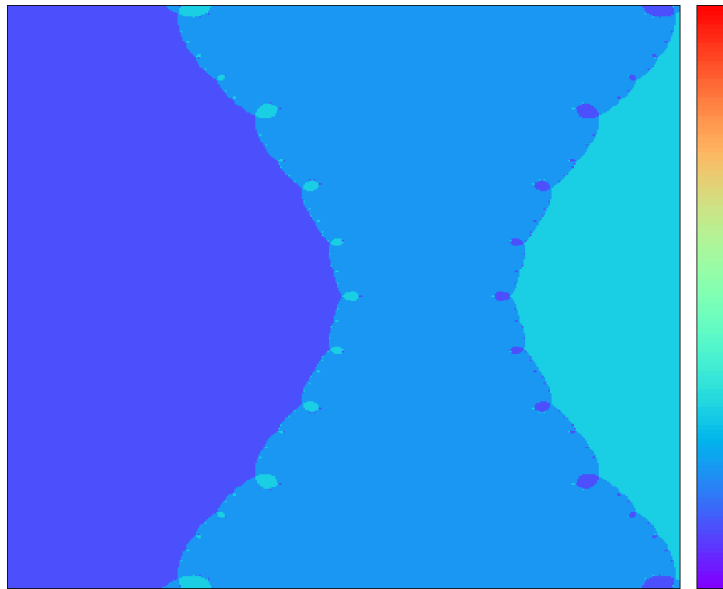
$$x^4 + 2x^3 + -7x^2 + 8x - 12$$

n = 400



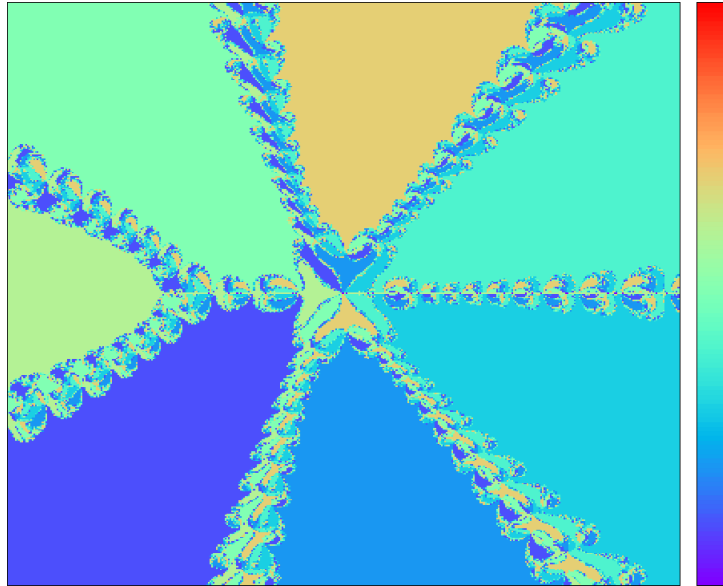
$$x^8 + -2x^7 + 3x^6 - 4x^5 + 5x^4 - 6x^3 + 7x^2 - 8x + 10$$

n = 400



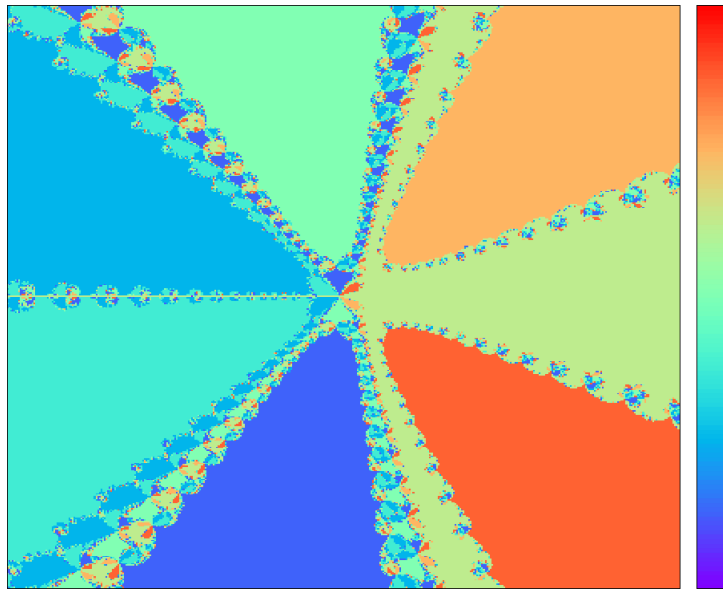
$$x^3 - 3x^2 + 2$$

n = 400



$$5x^7 + 12x^6 - \frac{3}{5}x^5 + \frac{7}{9}x^4 - 3x^2 + 21$$

n = 400



$$13x^7 - 5x^6 + 3x^4 + x^3 + 2x^2 + \frac{1}{9}x - 1$$

n = 400

■

### 3 Parte 3: Encontrando todas as raízes de uma função

O programa `root_finder` não trata de casos em que há mais do que uma raiz em um subintervalo uma vez que fica por conta do usuário colocar um `ninter` pequeno o suficiente para que isso não ocorra.

Ele também utiliza exclusivamente o método da secante e não o de Newton uma vez que seria muito mais difícil implementar ou encontrar métodos que realizariam a derivada de funções genéricas. Isso não é um problema na segunda parte do exercício pois apenas polinômios são utilizadas nela, havendo a função `polyder()`.

No caso de não haver decrescimento suficiente como especificado pelo enunciado ( $|f(x_k)| > 0.5|f(x_{k-1})|$ ), a função secante retorna `NaN` como indicador de que precisa rodar o método da bissecção três vezes antes de tentar o da secante novamente.

■