Disciplina: Métodos Numéricos</p>

Semestre: 2020.1</p>

Aluno: Jardel Brandon de Araujo Regis</p>

Mátricula: 201621250014</p>

1ª Unidade: REPRESENTAÇÃO NUMÉRICA E TIPOS DE ERROS</p>

Tratamento de Erros

Exercícios #1

In [1]:

```
import numpy as np
import pandas as pd
from itertools import *
from sympy import *
from decimal import *
import matplotlib.pyplot as plt
```

In [2]:

```
#funções auxiliares

erro_total = lambda valor_verdadeiro, aproximacao: valor_verdadeiro - aproximacao

erro_relativo_fracionario = lambda valor_verdadeiro, aproximacao: (valor_verdadeiro - a
proximacao) / valor_verdadeiro

erro_relativo_percentual = lambda valor_verdadeiro, aproximacao: (valor_verdadeiro - ap
roximacao) / valor_verdadeiro * 100

def imprimir_tabela(dados, index = None):
    pd.set_option("display.precision", 20)
    df = pd.DataFrame(dados, index)
    print(df)

def imprimir_grafico(dados):
    plt.xlabel('Intervalo')
    plt.ylabel('Valor resultante')
    plt.grid(True)
    plt.style.use('fivethirtyeight')
    plt.plot(np.arange(len(dados)), dados)
    plt.show
```

Exercício 1

O método “divisão e média”, um método antigo para estimação de raiz quadrada de um número positivo a, pode ser formulado como

$$xi + 1 = xi + a = xi2$$

Calcule o erro relativo da aproximação para as 10 primeiras iterações

In [3]:

```
divisao_e_media = lambda x, i: (i + np.divide(x, i)) / 2

valor = 55
iteracoes = 10

raiz_quadrada = np.sqrt(valor)
divisoes_e_medias = [divisao_e_media(valor, i) for i in range(iteracoes)]
erros_totais = [erro_total(raiz_quadrada, aproximacao) for aproximacao in divisoes_e_me
dias]
erros_relativos = [erro_relativo_fracionario(raiz_quadrada, aproximacao) for aproximaca
o in divisoes_e_medias]
erros_relativos_percentuais = [erro_relativo_percentual(raiz_quadrada, aproximacao) for
aproximacao in divisoes_e_medias]

tabela = {'Valor':                       valor,
          'Raiz quadrada':               raiz_quadrada,
          'Divisao e media (x_i+1)':     divisoes_e_medias,
          'Erro total':                  erros_totais,
          'Erro relativo fracionario':   erros_relativos,
          'Erro relativo percentual':    erros_relativos_percentuais}

imprimir_tabela(tabela)
```

```
   Valor           Raiz quadrada  Divisao e media (x_i+1)  \
0     55  7.41619848709566298339                      inf   
1     55  7.41619848709566298339  28.00000000000000000000   
2     55  7.41619848709566298339  14.75000000000000000000   
3     55  7.41619848709566298339  10.66666666666666607455   
4     55  7.41619848709566298339   8.87500000000000000000   
5     55  7.41619848709566298339   8.00000000000000000000   
6     55  7.41619848709566298339   7.58333333333333303727   
7     55  7.41619848709566298339   7.42857142857142882519   
8     55  7.41619848709566298339   7.43750000000000000000   
9     55  7.41619848709566298339   7.55555555555555535818   

               Erro total  Erro relativo fracionario  \
0                    -inf                       -inf   
1 -20.58380151290433701661    -2.77551922979415577331   
2  -7.33380151290433701661    -0.98888959426656419804   
3  -3.25046817957100309116    -0.43829303992158302750   
4  -1.45880151290433701661    -0.19670475587225469405   
5  -0.58380151290433701661    -0.07871977994118732613   
6  -0.16713484623767005388    -0.02253645806925045139   
7  -0.01237294147576584180    -0.00166836708824541491   
8  -0.02130151290433701661    -0.00287229541407259866   
9  -0.13935706845989237479    -0.01879090327778800903   

   Erro relativo percentual  
0                     -inf  
1 -277.55192297941556489604  
2  -98.88895942665641314306  
3  -43.82930399215830163939  
4  -19.67047558722546796162  
5   -7.87197799411873244679  
6   -2.25364580692504512527  
7   -0.16683670882454149087  
8   -0.28722954140725986960  
9   -1.87909032777880091736  

C:\Users\jarde\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: Runtim
eWarning: divide by zero encountered in true_divide
  """Entry point for launching an IPython kernel.
```

Exercício 2

Para computadores, o épsilon da máquina, ", pode ser definido como o menor número que, adicionado a um, retorna um número maior que um, como definimos anteriormente. Usando o algoritmo abaixo, implemente um programa que calcula o épsilon da sua máquina. Compare com os resultados obtidos via $numpy$.

> Passo 1: Defina e = 1
>
> Passo 2: Se 1 + e for menor ou igual a 1, vá para o Passo 5; caso contrário, vá ao Passo 3
>
> Passo 3: e = e/2
>
> Passo 4: Retorne ao Passo 2
>
> Passo 5: e = 2 x e

In [4]:

```
def epsilon(limiar):
    while((limiar + 1) > 1):
        limiar /= 2
    return limiar * 2

imprimir_tabela({'funcao criada': [epsilon(1)],
                 'funcao do numpy': [np.finfo(float).eps]})
```

```
           funcao criada        funcao do numpy
0  0.00000000000000022204  0.00000000000000022204
```

Exercício 3

Considere o seguinte processo iterativo:

$$x^{(1)} = \frac{1}{3}$$
$$x^{(n+1)} = 4x^{(n)} - 1, \quad n = 1, 2, \dots$$

Observe que $x^{(1)} = \frac{1}{3}, x^{(2)} = 4 \cdot \frac{1}{3} - 1 = \frac{1}{3}, x^{(3)} = \frac{1}{3}$, ou seja, temos uma sequência constante igual a $\frac{1}{3}$.
Implemente essa série iterativa, verificando se a convergência de fato ocorre e justifique o resultado obtido

In [5]:

```
memorizacao = dict() #Atal
def processo_iterativo(indice):
    if indice == 0: return 1 / 3
    if indice not in memorizacao:
        memorizacao[indice] = 4 * processo_iterativo(indice - 1) - 1
    return memorizacao[indice]

QUANTIDADE_ITERACOES = 540
resultado = processo_iterativo(QUANTIDADE_ITERACOES)
imprimir_tabela({'Valor': '1/3','Resultado': 1 / 3, 'Processo iterativo': list(memoriza
cao.values())})
```

```
    Valor              Resultado          Processo iterativo
0     1/3  0.3333333333333333331483   3.333333333333333259318e-01
1     1/3  0.3333333333333333331483   3.333333333333333037274e-01
2     1/3  0.3333333333333333331483   3.333333333333332149095e-01
3     1/3  0.3333333333333333331483   3.333333333333328596382e-01
4     1/3  0.3333333333333333331483   3.333333333333314385527e-01
..    ...                     ...                         ...
535   1/3  0.3333333333333333331483 -9.362985077407894833301e+305
536   1/3  0.3333333333333333331483 -3.745194030963157933320e+306
537   1/3  0.3333333333333333331483 -1.498077612385263173280e+307
538   1/3  0.3333333333333333331483 -5.992310449541052693120e+307
539   1/3  0.3333333333333333331483                         -inf

[540 rows x 3 columns]
```

Comentário:

Como podemos observar, a medida em que o valor da iteração aumenta, a difernça do valor "Real" também aumenta, de tal maneira que chega ao um número "infinito", isso implica que pelo fato do computador não possuir mémoria infinita, se faz necessário realizar um truncamento na precisão decimal de certos números, fazendo existir asim uma propragação do erro, como o erro visto acima.

Questão 4

Considere as expressões:

$$\frac{\exp(\frac{1}{\mu})}{1+\exp(\frac{1}{\mu})}$$

e

$$\frac{1}{\exp(\frac{-1}{\mu})+1}$$

com $\mu > 0$. Verifique que elas são idênticas como funções reais. Teste no computador cada uma delas para $\mu = 0,1$, $\mu = 0,01$ e $\mu = 0,001$. Qual dessas expressões é mais adequada quando μ é um número pequeno? Por quê?

In [6]:

```
expressao_1 = lambda u: np.exp(np.divide(1, u)) / 1 + np.exp(np.divide(1, u))
expressao_2 = lambda u: 1 / np.exp(np.divide(-1, u)) + 1

iteracoes = range(10, -4, -1)
imprimir_tabela({'Expressao 1': [expressao_1(10 ** i) for i in iteracoes],
                 'Expressao 2': [expressao_2(10 ** i) for i in iteracoes]}, index=itera
coes)
```

```
                     Expressao 1                  Expressao 2
10  2.0000000002000000001655e+00  2.0000000001000000000827e+00
9   2.0000000020000000016548e+00  2.0000000010000000008274e+00
8   2.0000000199999999987845e+00  2.0000000099999999993923e+00
7   2.0000002000000000988674e+00  2.0000001000000000472132e+00
6   2.0000020000000099992437e+00  2.0000010000000050018423e+00
5   2.0000200001000013930e+00  2.0000100000500000006965e+00
4   2.0002000100003334282 0e+00  2.0001000050001667141 0e+00
3   2.0020010003334167691 9e+00  2.0010005001667083846 0e+00
2   2.0201003341683358982 6e+00  2.0100501670841679491 3e+00
1   2.2103418361512954248 8e+00  2.1051709180756477124 4e+00
0   5.4365636569180901815 9e+00  3.7182818284590450908 0e+00
-1  4.4052931589613435789 9e+04  2.2026465794806714257 0e+04
-2  5.3762342836322712188 5e+43  2.6881171418161351142 5e+43
-3                           inf                          inf
```

```
C:\Users\jarde\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: Runtim
eWarning: overflow encountered in exp
  """Entry point for launching an IPython kernel.
C:\Users\jarde\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: Runtim
eWarning: divide by zero encountered in double_scalars
```

Comentário:

Como podemos observar, no gráfico abaixo, ou em resultados de calculadoras, a tendência para um valor de entrada que tende a -infinito é convergir para $0,5$ de tal forma que o valor da expressão 1 se torna mais fiel ao valor correto.

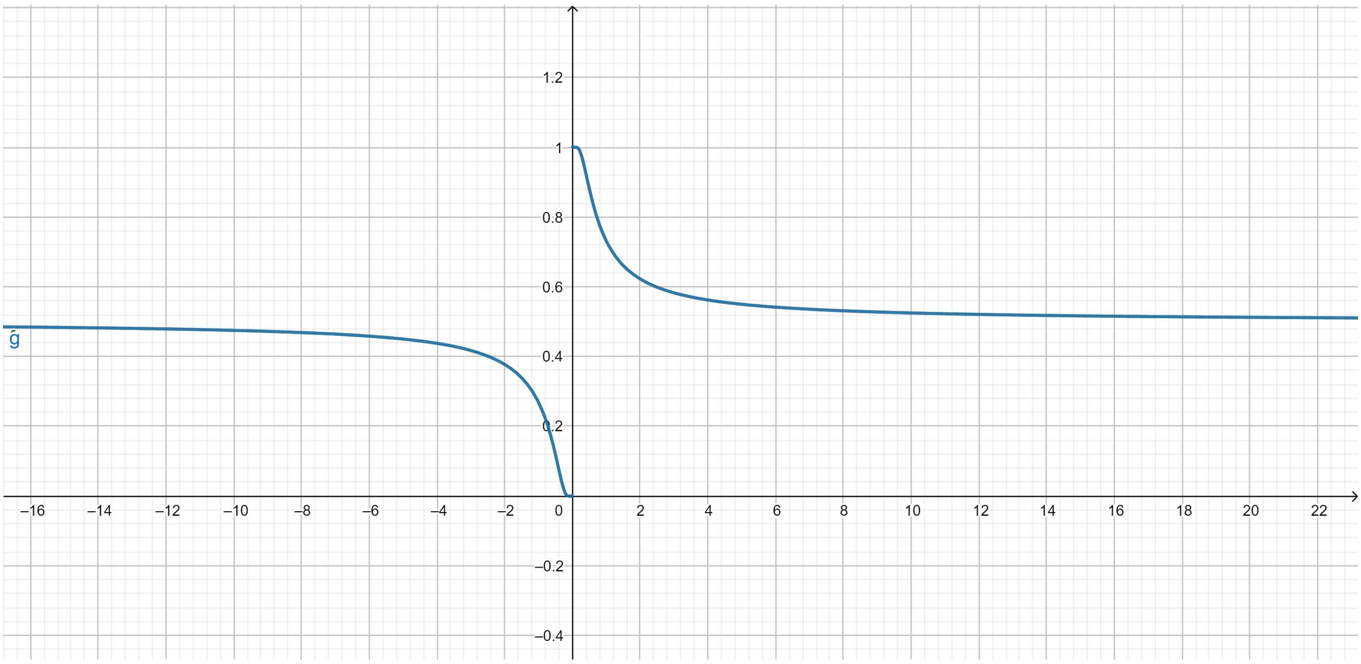

Gráfico disponível no seguinte link: https://www.geogebra.org/calculator/gdwzwtn2 (https://www.geogebra.org/calculator/gdwzwtn2)

Questão 5

Observe a seguinte identidade

$$f(x) = \frac{(1+x)-1}{x} = 1$$

Calcule o valor da expressão à esquerda para $x = 10^{-12}, x = 10^{-13}, x = 10^{-14}, x = 10^{-15}, x = 10^{-16}$ e $x = 10^{-17}$. Explique os resultados.

In [7]:

```
identidade = lambda x: ((1 + x) - 1) / x
imprimir_tabela([identidade(10 ** x) for x in range(-12, -18, -1)], index= list(range(-
12, -18, -1)))
```

```
                         0
-12  1.00008890058234101161
-13  0.99920072216264088638
-14  0.99920072216264088638
-15  1.11022302462515654042
-16  0.00000000000000000000
-17  0.00000000000000000000
```

Comentário:

Como é possível observar, a identidade matemática apresentada acima não se faz correspondida fielmente computacionalmente, pois de acordo com a fórmula para quaisquer valores da variável (x), o resultado deveria ser 1. Porém os resultados obtidos demonstram a quebra dessa regra, isso acontece devido ao fato da limitação computação para realizar trabalhos com grandes números ou com decimais, assim pode-se observar o efeito acontecido nessa questão.