Métodos Numéricos I

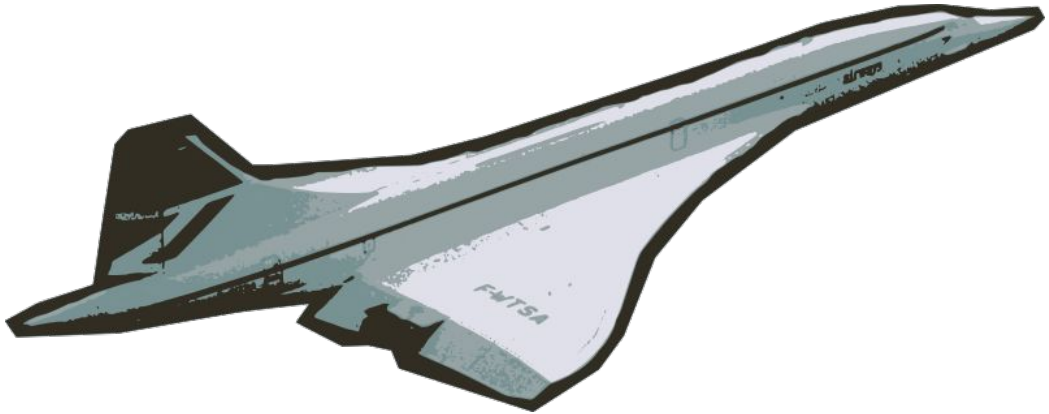
Implementação e comparativo de métodos

Introdução:

Deslocamento dado pela equação:

$$f(d) = a * d - d * \ln(d)$$

Onde, por segurança, **d** precisa ser menor, no máximo, 2 cm.

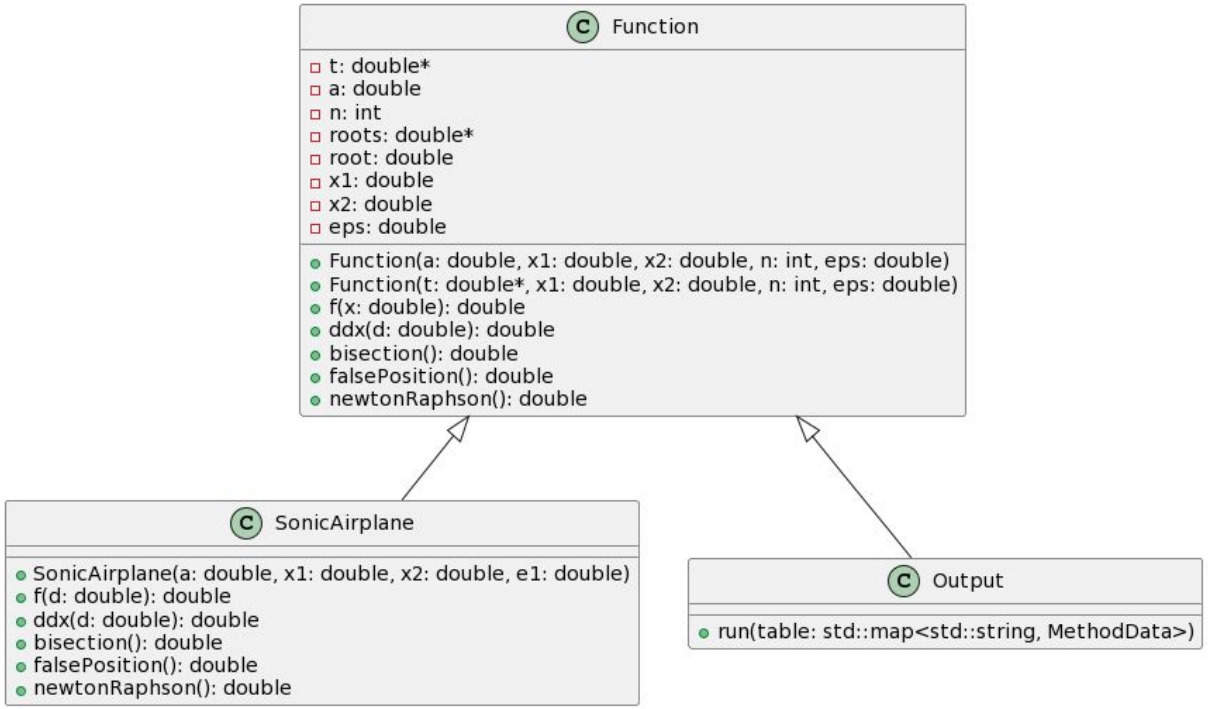


Metodologia:

Divisão do projeto em:

- **Implementação dos métodos**
- **Tratamento dos dados**
- **Exibição dos dados na tela**
- **Comparação com dados reais**

Diagrama


Function
t: double*
a: double
n: int
roots: double*
root: double
x1: double
x2: double
eps: double
Function(a: double, x1: double, x2: double, n: int, eps: double)
Function(t: double*, x1: double, x2: double, n: int, eps: double)
f(x: double): double
ddx(d: double): double
bisection(): double
falsePosition(): double
newtonRaphson(): double
SonicAirplane
SonicAirplane(a: double, x1: double, x2: double, e1: double)
f(d: double): double
ddx(d: double): double
bisection(): double
falsePosition(): double
newtonRaphson(): double
Output
run(table: std::map<std::string, MethodData>)

Cálculo da função no ponto e da derivada no ponto:

f(d) = a * d - d * ln(d)

f’(d) = a - ln(d) + 1

```
double SonicAirplane::f(double d){
      return (getA() * d - d * (log(d)));
}

double SonicAirplane::ddx(double d){
      return (getA() - (log(d) + 1));
}
```

Isolamento

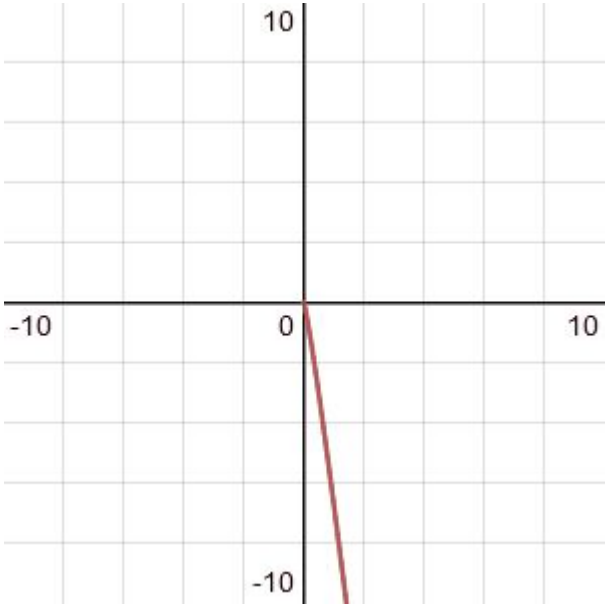
1.) $0 = a * d - d * \ln(d)$

2.) $0 = d * (a - \ln(d))$

3.) $0 = (a - \ln(d))$

4.) $a = \ln(d)$

5.) $d = e^a$



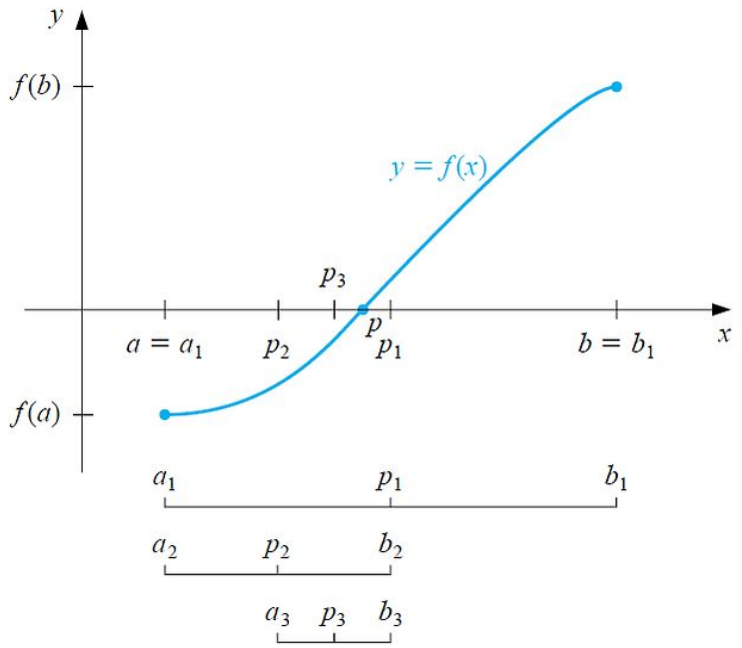
Isolamento

- **e^a** é sempre a raiz.
- Usamos como isolamento padrão:

$[\ \llcorner e^a \lrcorner, \ulcorner e^a \urcorner\ ]$

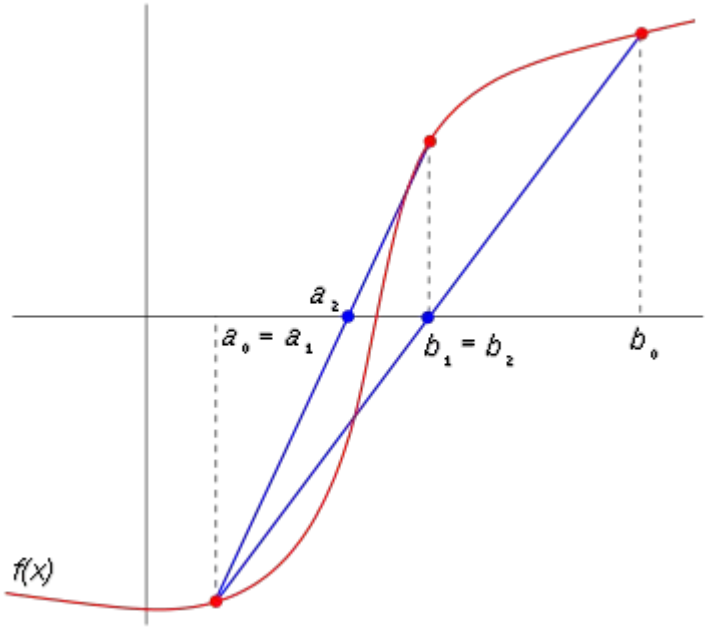
Bisseção:

```
double SonicAirplane::bisection(){
	double xk, f1, f2;
	int i = 0;
	double x1 = getX1();
	double x2 = getX2();
	double epsk = abs(x1 - x2) - getEps();
	while((epsk > 0) && (i < 100)){
	put(xk, epsk);
	xk = (x1 + x2) / 2;
	f1 = f(x1);
	f2 = f(x2);
	// Choose between x1 and x2
	if (f1 == 0){
		return f1;
	} else if (f2 == 0) {
		return f2;
	} else if (f1 * f2 < 0) {
		x2 = xk;
	} else {
		x1 = xk;
	}
	epsk = abs(x1 - x2) - getEps();
	i++;
	}
	return xk;
}
```



Posição Falsa:

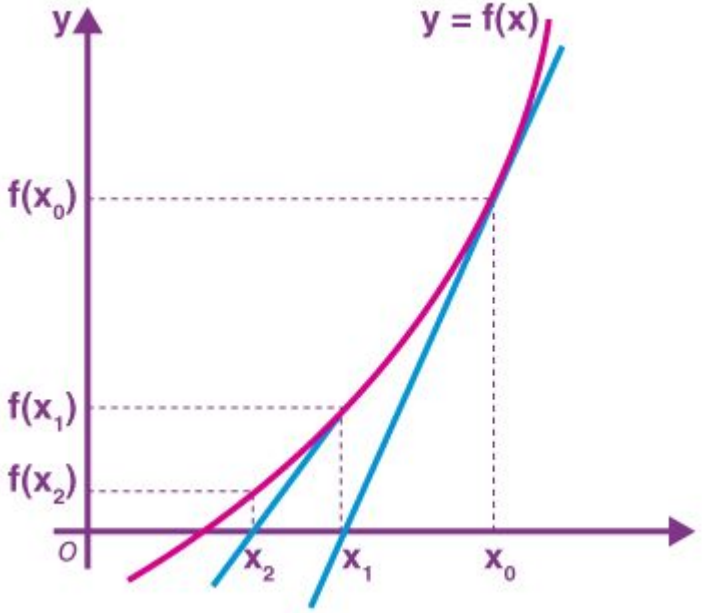
```
double SonicAirplane::falsePosition(){
        double xk, f1, f2;
        int i = 0;
        double x1 = getX1();
        double x2 = getX2();
        double epsk = abs(x1 - x2) - getEps();
        while((epsk > 0) && (i < 100)){
        put(xk, epsk);
        if ((f(x2) - f(x1)) != 0){
                xk = (x1 * f(x2) - x2 * f(x1)) / (f(x2) - f(x1));
        }  else {
                break;
        }
        f1 = f(x1);
        f2 = f(x2);
        // Choose between x1 and x2
        if (f1 == 0){
                return x1;
        } else if (f2 == 0) {
                return x2;
        } else if (f1 * f2 < 0) {
                x2 = xk;
        } else {
                x1 = xk;
        }
        epsk = abs(x1 - x2) - getEps();
        i++;
        }
        return xk;
}
```



Newton-Raphson:

```
double SonicAirplane::newtonRaphson(){
	double xk0, xk1, f1, f2;
	int i = 0;
	xk0 = getX1();
	try{
	xk1 = xk0 - (f(xk0)/ddx(xk0));
	} catch (...){
	return xk0;
	}
	double epsk = abs(xk0 - xk1) - getEps();

	while((epsk > 0) && (i < 100)){
	put(xk1, epsk);
	try{
		xk0 = xk1;
		xk1 = xk0 - (f(xk0)/ddx(xk0));
		i++;
	} catch (...){
		return xk0;
	}
	epsk = abs(xk0 - xk1) - getEps();
	}
	return xk1;
}
```



Interface:

O usuário digita erro e a quantidade de aviões que ele vai inserir. O programa pede a entrada de cada **a**, exibindo uma tabela com os resultados e comparativo para cada método

```
Enter an error (epsilon):
Enter the number of 'as' you want to insert:
```

Interface

Exibindo o quadro resposta

```
Enter an error (epsilon): 0.001
Enter the number of 'as' you want to insert: 1

Default case a = 1. ; insulation = (2., 3.) ; eps = 10e-5
|*******        Bisection       ********||***** False-Position          *****||*****    Newton-Raphson          *****|
|  Iter         Result          Err        ||  Iter     Result          Err        ||  Iter     Result          Err        |
|  [000]        0.000000        0.999900   ||  [000]    0.000000        0.999900   ||  [000]    2.885390        0.885290   |
|  [001]        2.500000        0.499900   ||  [001]    2.674741        0.674641   ||  [001]    2.722939        0.162351   |
|  [002]        2.750000        0.249900   ||  [002]    2.725822        0.050980   ||  [002]    2.718286        0.004553   |
|  [003]        2.625000        0.124900   ||  [003]    2.718221        0.007501   ||  [003]    2.718282        0.000000   |
|  [004]        2.687500        0.062400   ||  [004]    2.718282        0.007440   ||  [---]    --------        --------   |
|  [005]        2.718750        0.031150   ||  [005]    2.718282        0.007440   ||  [---]    --------        --------   |
|  [006]        2.703125        0.015525   ||  [006]    2.718282        0.007440   ||  [---]    --------        --------   |
|  [007]        2.710938        0.007712   ||  [007]    2.718282        0.007440   ||  [---]    --------        --------   |
|  [008]        2.714844        0.003806   ||  [008]    2.718282        0.007440   ||  [---]    --------        --------   |
|  [009]        2.716797        0.001853   ||  [009]    2.718282        0.000000   ||  [---]    --------        --------   |
|  [010]        2.717773        0.000877   ||  [---]    --------        --------   ||  [---]    --------        --------   |
|  [011]        2.718262        0.000388   ||  [---]    --------        --------   ||  [---]    --------        --------   |
|  [012]        2.718506        0.000144   ||  [---]    --------        --------   ||  [---]    --------        --------   |
|  [013]        2.718384        0.000022   ||  [---]    --------        --------   ||  [---]    --------        --------   |
|  [014]        2.718323        0.000000   ||  [---]    --------        --------   ||  [---]    --------        --------   |
```

Aproximações de π ao longo da história

Pré métodos numéricos + computadores:

- Arquimedes - 3 casas decimais de precisão
- Ghiyath al-Kashi - 35 casas decimais de precisão
- William Shanks - 527 casas decimais de precisão

Era dos métodos números + computadores:

- 1961 - 100.265 casas decimais de precisão
- Prof. Roberto N. Onody - 62,8 trilhões casas decimais de precisão

Interface: plot dos gráficos

Com a finalidade de visualizar os resultados graficamente, utilizamos o software Gnuplot para fazer os gráficos para os seguintes casos de a = 1; a = 0,5 e a = 0,25; todos com eps = 10^-4

Para isso criamos dois scripts, um que é possível ver o comportamento geral da função e outro com um foco maior no posicionamento dos resultados dos métodos e na raiz.

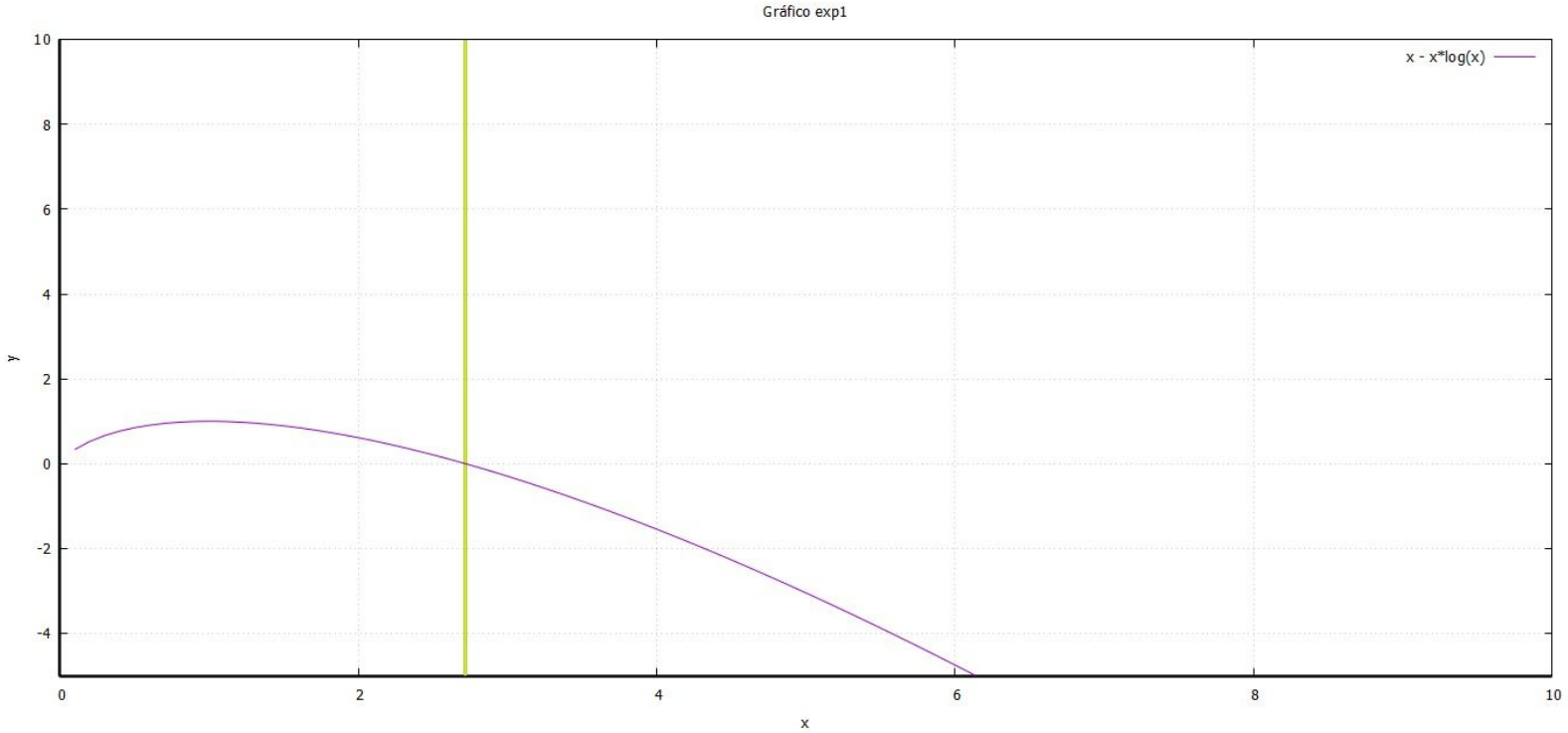
Para representar as raízes traçamos 4 retas paralelas ao eixo y, que representavam os seguintes resultados:

- A raiz representada pela reta vermelha
- O resultado encontrado pela bisseção pela reta azul
- O resultado encontrado pelo método de newton pela reta verde
- O resultado encontrado pela posição falsa pela reta amarela

Obs:Como o resultado da posição falsa e de newton foram iguais, terminamos com uma linha esverdeada nos casos com o zoom maior

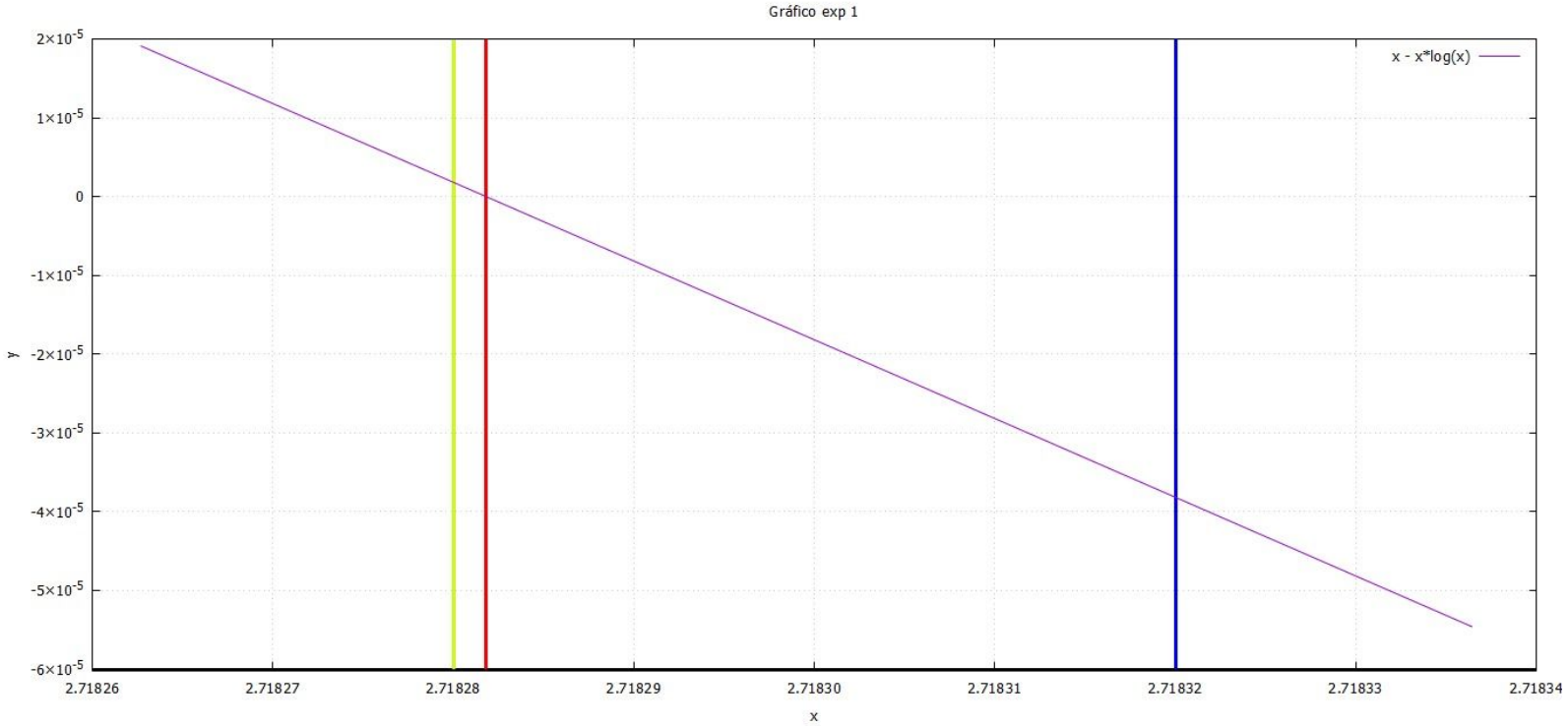
Interface: Resultado dos gráficos

a = 1



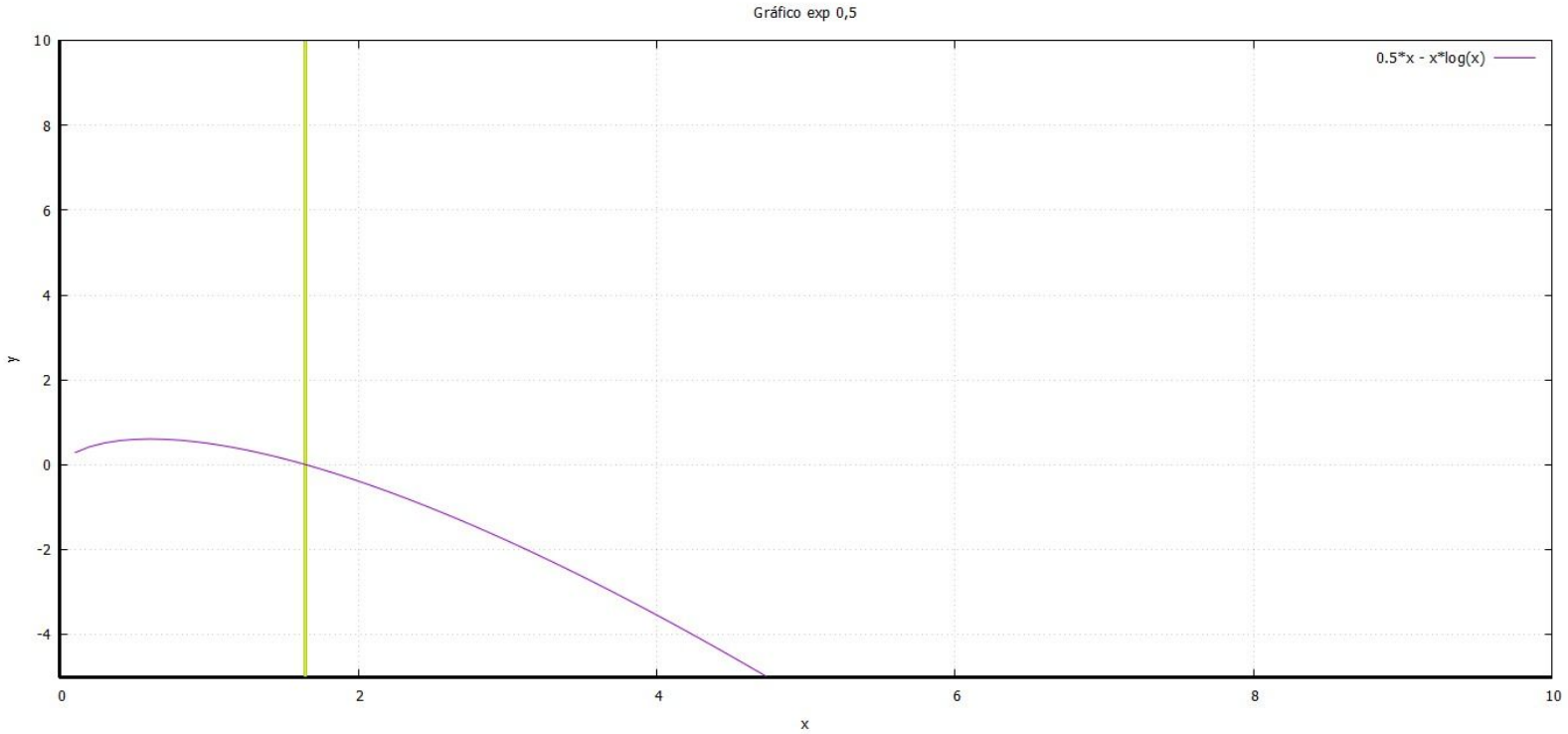
Interface: Resultado dos gráficos

a = 1



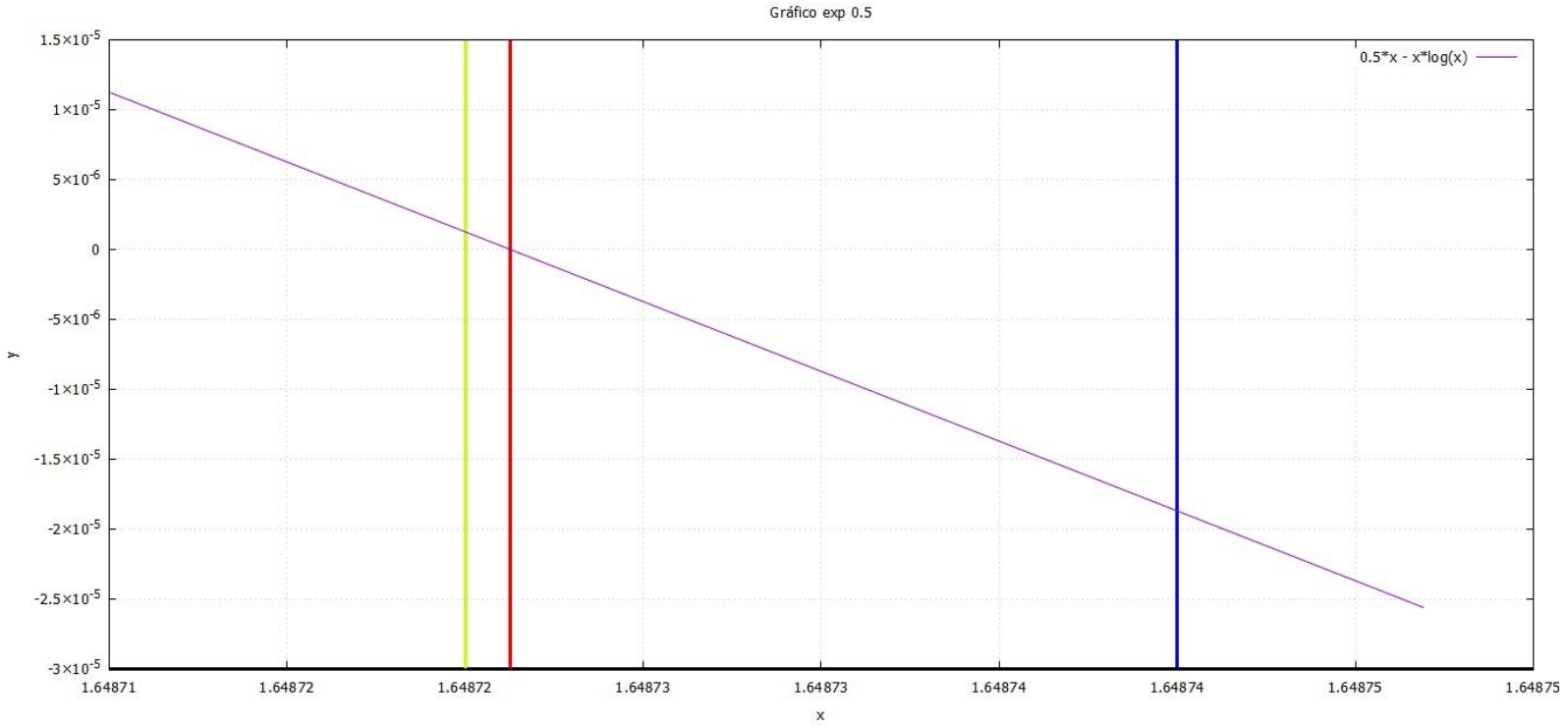
Interface: Resultado dos gráficos

a = 0.5



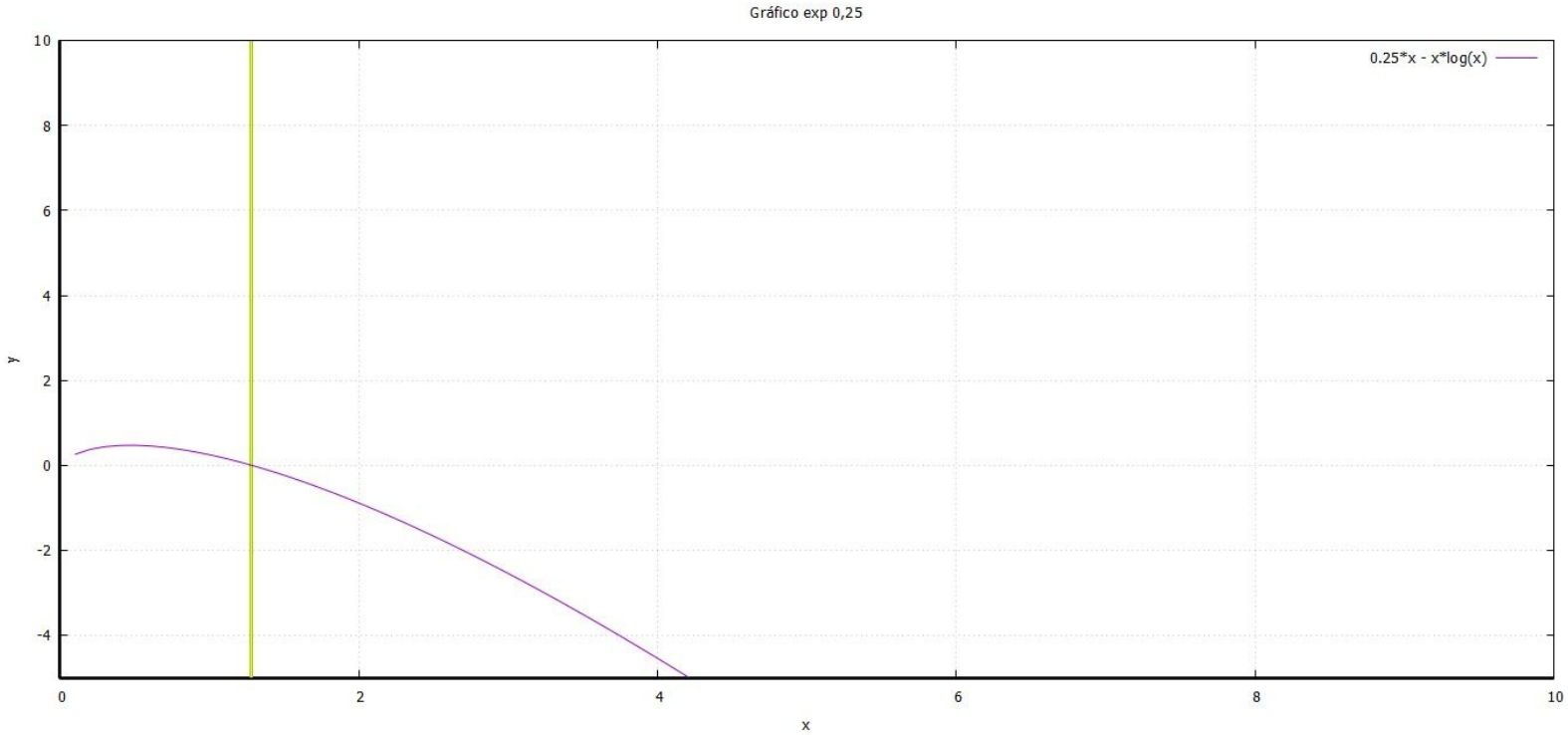
Interface: Resultado dos gráficos

a = 0.5



Interface: Resultado dos gráficos

a = 0.25



Interface: Resultado dos gráficos

a = 0.25