UNIVERSIDADE FEDERAL DO CEARÁ - UFC
CIÊNCIA DA COMPUTAÇÃO
PROJETO DE MÉTODOS NUMÉRICOS I

TEMA (2): Encontrar valor de PI

EQUIPE: π kachu

Equipe

* Caio Viktor
* Cristiano Melo
* Lucas Falcão
* Geraldo Braz
* Matheus Mayron

Objetivo

* **Encontrar e analisar aproximações de pi pelos métodos:**
  - Posição falsa;
  - Newton-Raphson;
  - Secante.

Ferramentas de desenvolvimento

C++

Qt

Função

* f(x) = cos(x) + (1 – a)

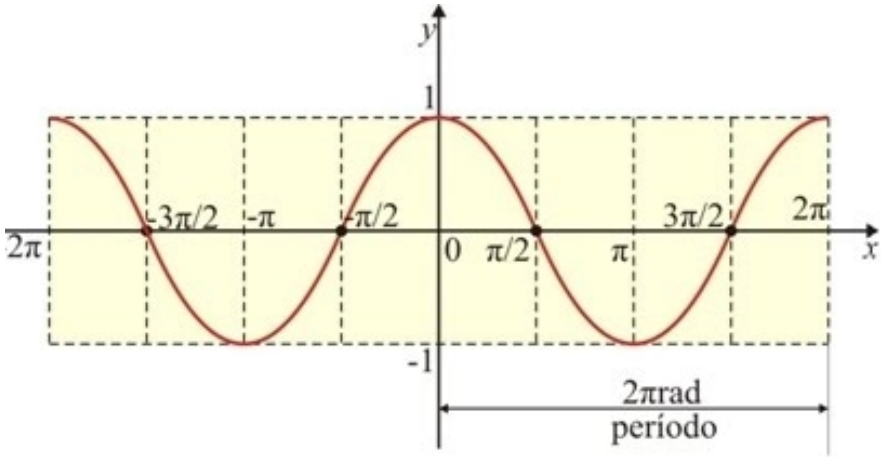
* Sabemos que é a raíz:

cos() + (1 - a) = 0

cos() = - (1 – a)

* Visto que cos() = -1, então:

-1 = -1 + a , portanto, a = 0



Métodos

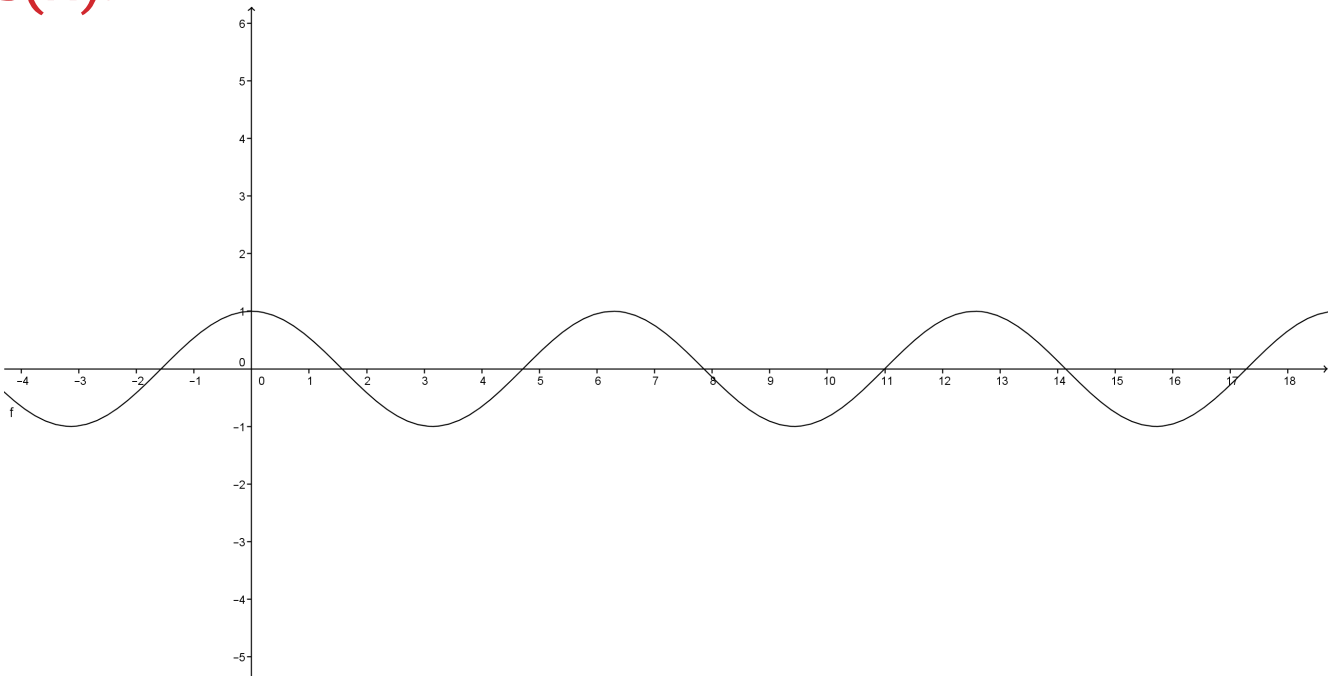
* Posição falsa:

```
.
:
repita
   x ← (aFb – bFa)/(Fb–Fa); Fx ← f(x)
   escreva k, a, Fa, b, Fb, x, Fx, intervX
   se abs(f(x)) < ε2 ou k ≥ iterMax então raiz ← x;
Fim.
   se Fa * Fx > 0 então a ← x; Fa ← Fx
   senão b ← x; Fb ← Fx
   intervX ← abs(b–a)
   se intervX ≤ ε1 então
      raiz ← escolha(a,b); Fim.
   fim se
   k ← k+1
 fim repita
fim algoritmo
```

```
104         if( Fa * Fb < 0 ){
105
106                 range = fabs( b - a ); // Computing range
107                 k = 0;
108
109                 do{
110
111                         x = b - Fb*(b - a)/(Fb - Fa);
112                         Fx = this->Function(x);
113                         if( Fa * Fx > 0 ){
114                                 a = x;
115                                 Fa = Fx;
116                         }else{
117                                 b = x;
118                                 Fb = Fx;
119                         }
120                         range = fabs( b - a );
121                         k++;
122
123                 }while( fabs( Fx ) > error && k <= maxIter );
124
125         }else{
126
127                 cout << "There is no root in this range" << endl;
128
129         }
```

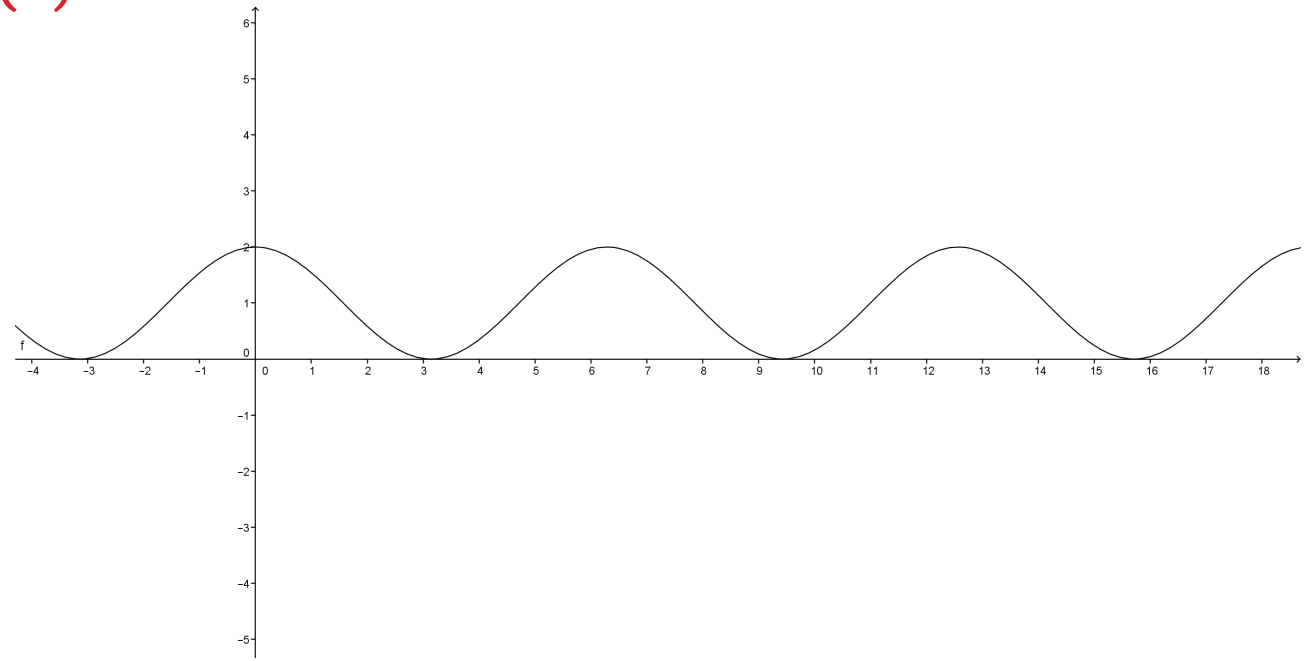
Estudo de caso: qual melhor valor de "a" na função para posição falsa?

* Veja a função cos(x)+(1 – a), para a = 1:
* cos(x):



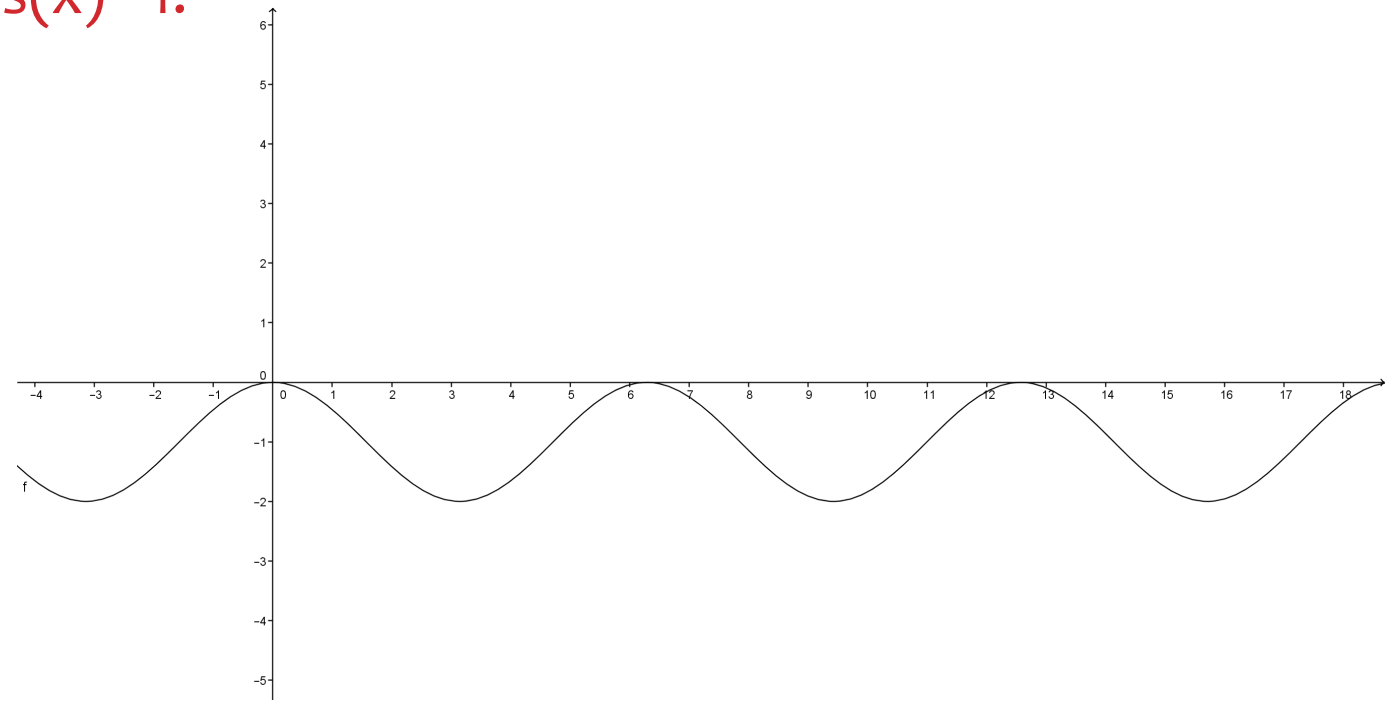
Estudo de caso: qual melhor valor de "a" na função para posição falsa?

* Veja a função cos(x)+(1 – a), para a = 0:
* cos(x) + 1:



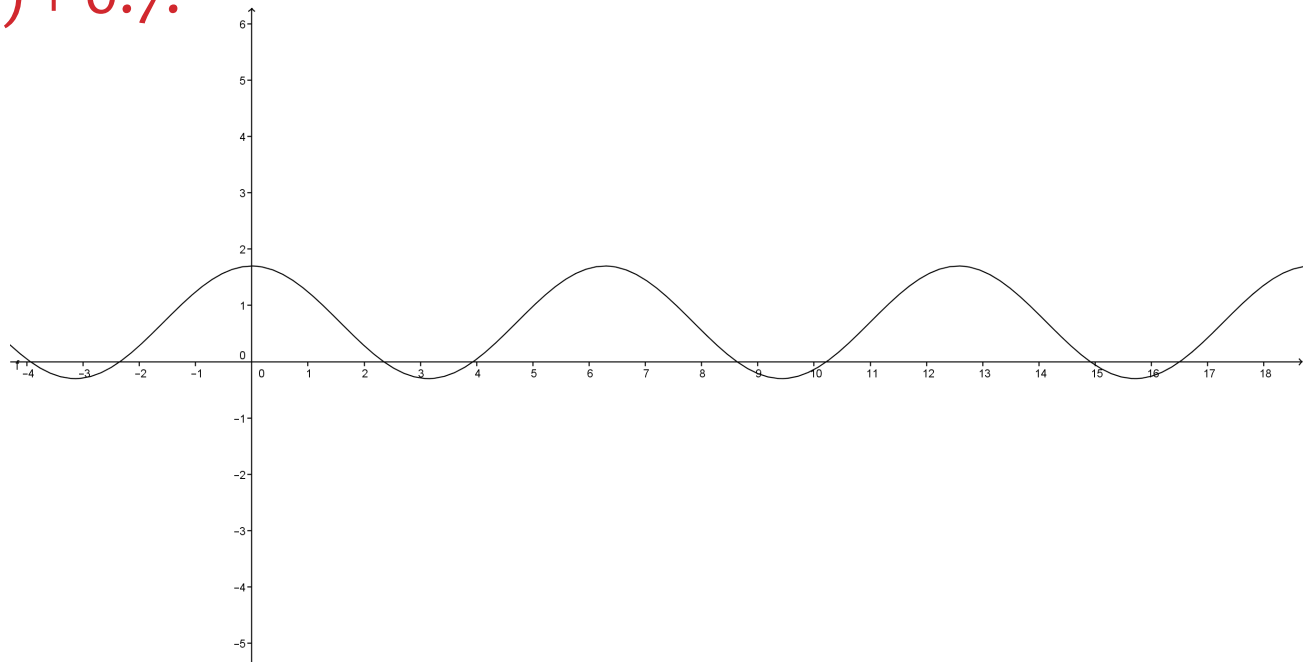
Estudo de caso: qual melhor valor de “a” na função para posição falsa?

* Veja a função cos(x)+(1 – a), para a = 2:
* cos(x) - 1:



Estudo de caso: qual melhor valor de "a" na função para posição falsa?

* Veja a função cos(x)+(1 – a), para a = 0.3:
* cos(x) + 0.7:



Métodos

* Newton-Raphson:

Algoritmo: Newton–Raphson
Entrada: x_0, ε_1, ε_2, iterMax
Saída: raiz
se $abs(f(x_0)) < \varepsilon_1$ **então** raiz ← x_0; **Fim.**
$k \leftarrow 1$
repita
$x_1 \leftarrow x_0 - f(x_0)/f'(x_0)$
escreva k, x_1, $f(x_1)$
se $abs(f(x_1)) < \varepsilon_1$ **ou** $abs(x_1 - x_0) < \varepsilon_2$ ou $k \geq$ iterMax **então**
raiz ← x_1; **Fim.**
fim se
$x_0 \leftarrow x_1$
$k \leftarrow k+1$
fim repita
fim algoritmo

```
17  void GenericMethod::loop() {
18      operacoesAntesDoLoop();
19      do{
20          calcularAproximacaoSeguinte();
21          salvarEmLista();
22      }while(!testeParadaErro1() && !testeParadaErro2() && (iterationsNumber < maxIteration));
23      operacoesAposLoop();
24      this->value = this->getAproximacaoAtualDaRaiz();
25  }
```

Métodos

* Secante:

Algoritmo: Secante
Entrada: x_0, x_1, ε_1, ε_2, iterMax
Saída: raiz
se $abs(f(x_0)) < \varepsilon_1$ **então** raiz ← x_0; **Fim.**
se $abs(f(x_1)) < \varepsilon_1$ **ou** $abs(x_1 - x_0) < \varepsilon_2$ **então** raiz ← x_1; **Fim.**
k ← 1
repita
  $x_2 \leftarrow x_1 - f(x_1)/(f(x_1) - f(x_0)) * (x_1 - x_0)$
  escreva k, x_2, $f(x_2)$
  se $abs(f(x_2)) < \varepsilon_1$ **ou** $abs(x_2 - x_1) < \varepsilon_2$ ou $k \geq$ iterMax **então**
    raiz ← x_2; **Fim.**
  fim se
  $x_0 \leftarrow x_1$
  $x_1 \leftarrow x_2$
  k ← k+1
fim repita
fim algoritmo

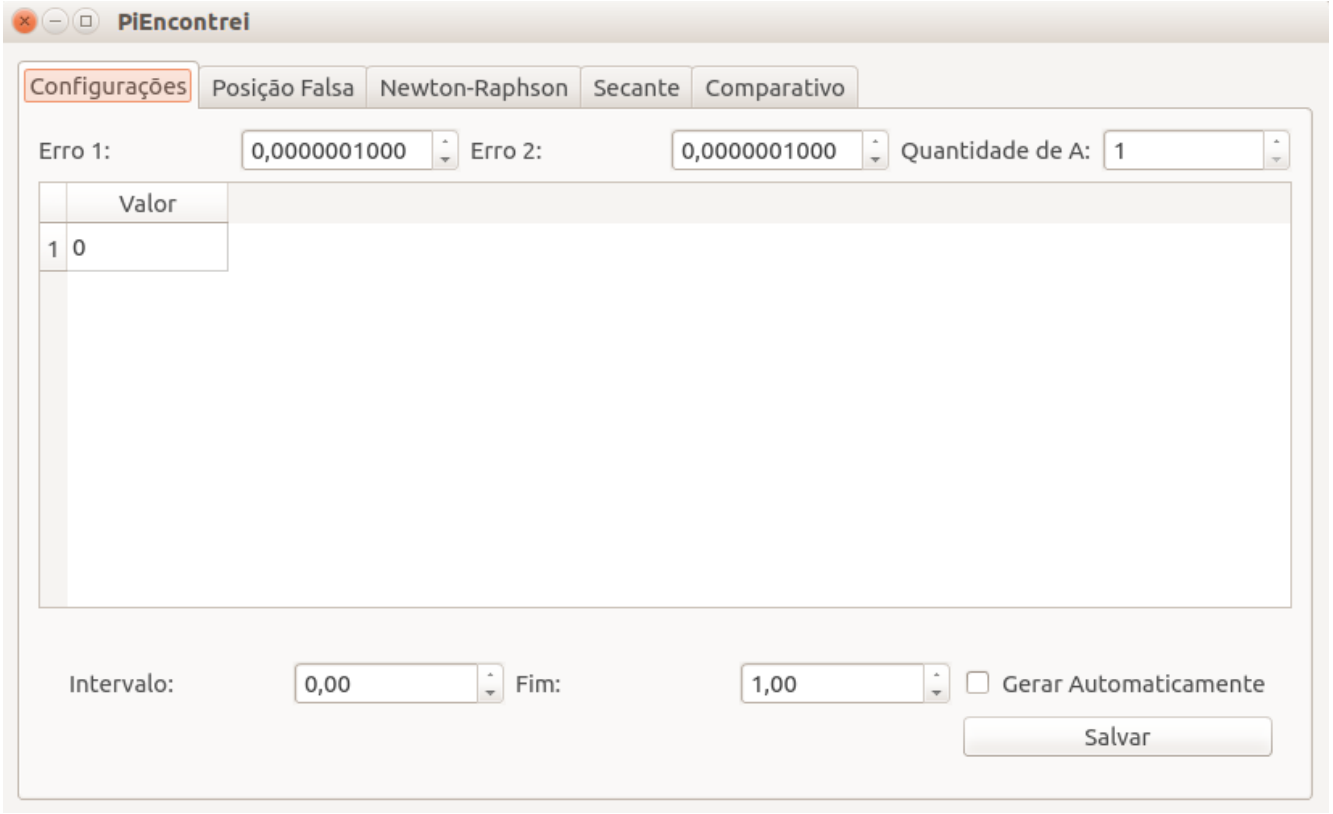
```
17  void GenericMethod::loop() {
18      operacoesAntesDoLoop();
19      do{
20          calcularAproximacaoSeguinte();
21          salvarEmLista();
22      }while(!testeParadaErro1() && !testeParadaErro2() && (iterationsNumber < maxIteration));
23      operacoesAposLoop();
24      this->value = this->getAproximacaoAtualDaRaiz();
25  }
```

Critério de parada

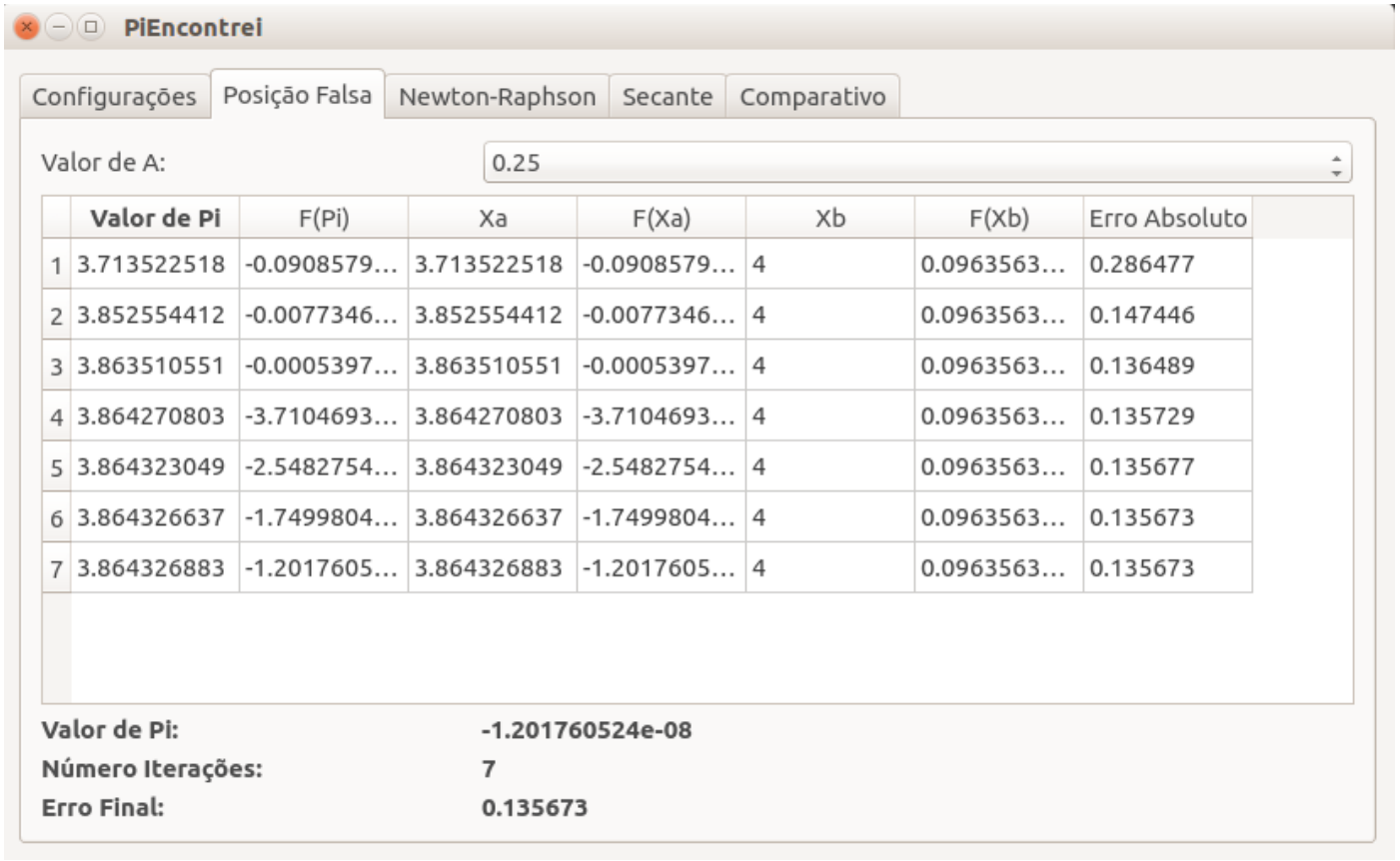
```
33  bool GenericMethod::testeParadaErro1() {
34      if(this->useTest1)
35          return (abs(function(getAproximacaoSeguinteDaRaiz())) < getErro1());
36      else
37          return false;
38  }
39
40  bool GenericMethod::testeParadaErro2() {
41
42      return (abs(getAproximacaoSeguinteDaRaiz() - getAproximacaoAtualDaRaiz()) < getErro2());
43  }
```

Pessoal, é importante observar que os critérios de paradas são os mesmos para o método da secante e do Newton-Raphson, pois ambos são derivados do ponto fixo.

Interface

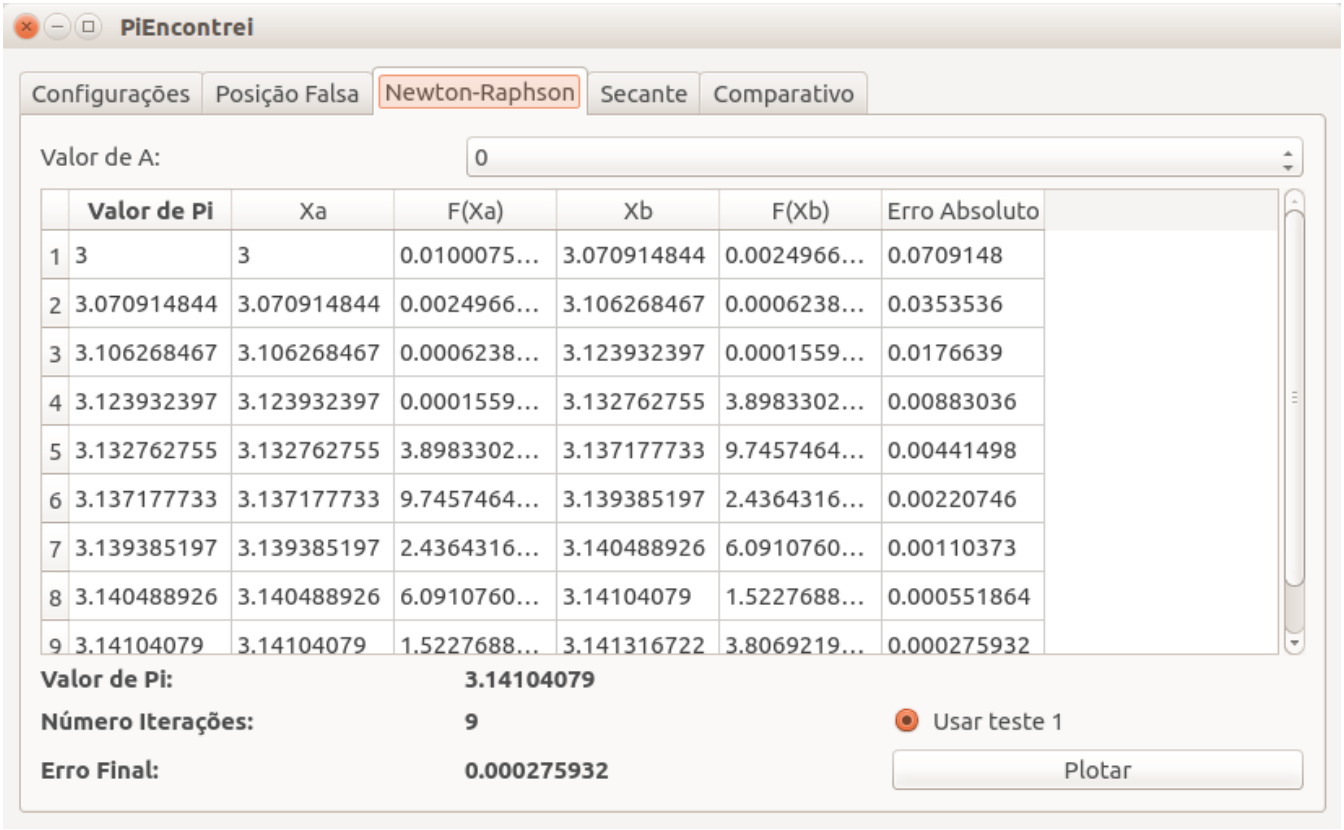

PiEncontrei
Configurações
Posição Falsa
Newton-Raphson
Secante
Comparativo
Erro 1:
0,0000001000
Erro 2:
0,0000001000
Quantidade de A:
1
Valor
1
0
Intervalo:
0,00
Fim:
1,00
Gerar Automaticamente
Salvar

Interface



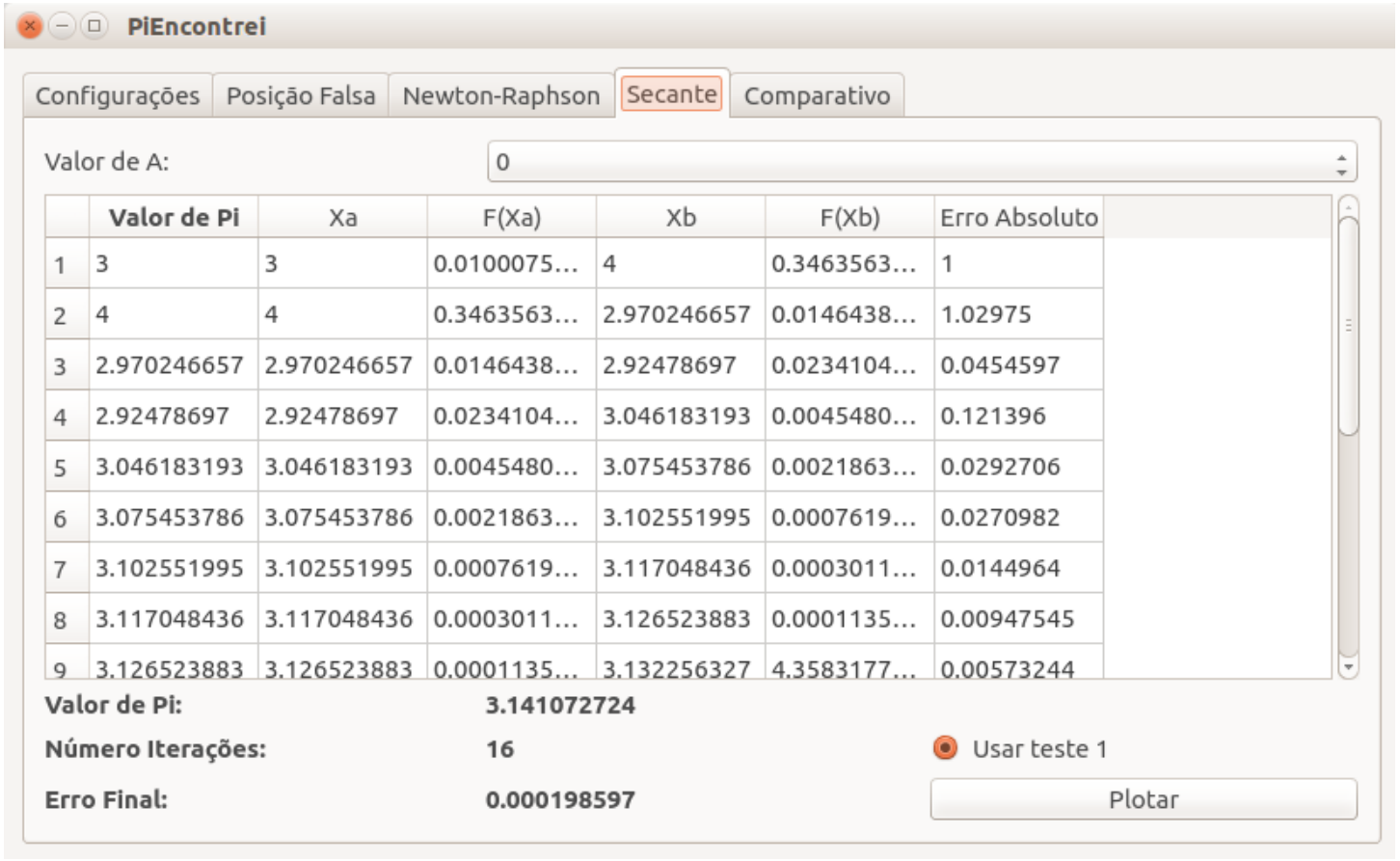
	Valor de Pi	F(Pi)	Xa	F(Xa)	Xb	F(Xb)	Erro Absoluto
1	3.713522518	-0.0908579…	3.713522518	-0.0908579…	4	0.0963563…	0.286477
2	3.852554412	-0.0077346…	3.852554412	-0.0077346…	4	0.0963563…	0.147446
3	3.863510551	-0.0005397…	3.863510551	-0.0005397…	4	0.0963563…	0.136489
4	3.864270803	-3.7104693…	3.864270803	-3.7104693…	4	0.0963563…	0.135729
5	3.864323049	-2.5482754…	3.864323049	-2.5482754…	4	0.0963563…	0.135677
6	3.864326637	-1.7499804…	3.864326637	-1.7499804…	4	0.0963563…	0.135673
7	3.864326883	-1.2017605…	3.864326883	-1.2017605…	4	0.0963563…	0.135673

Interface



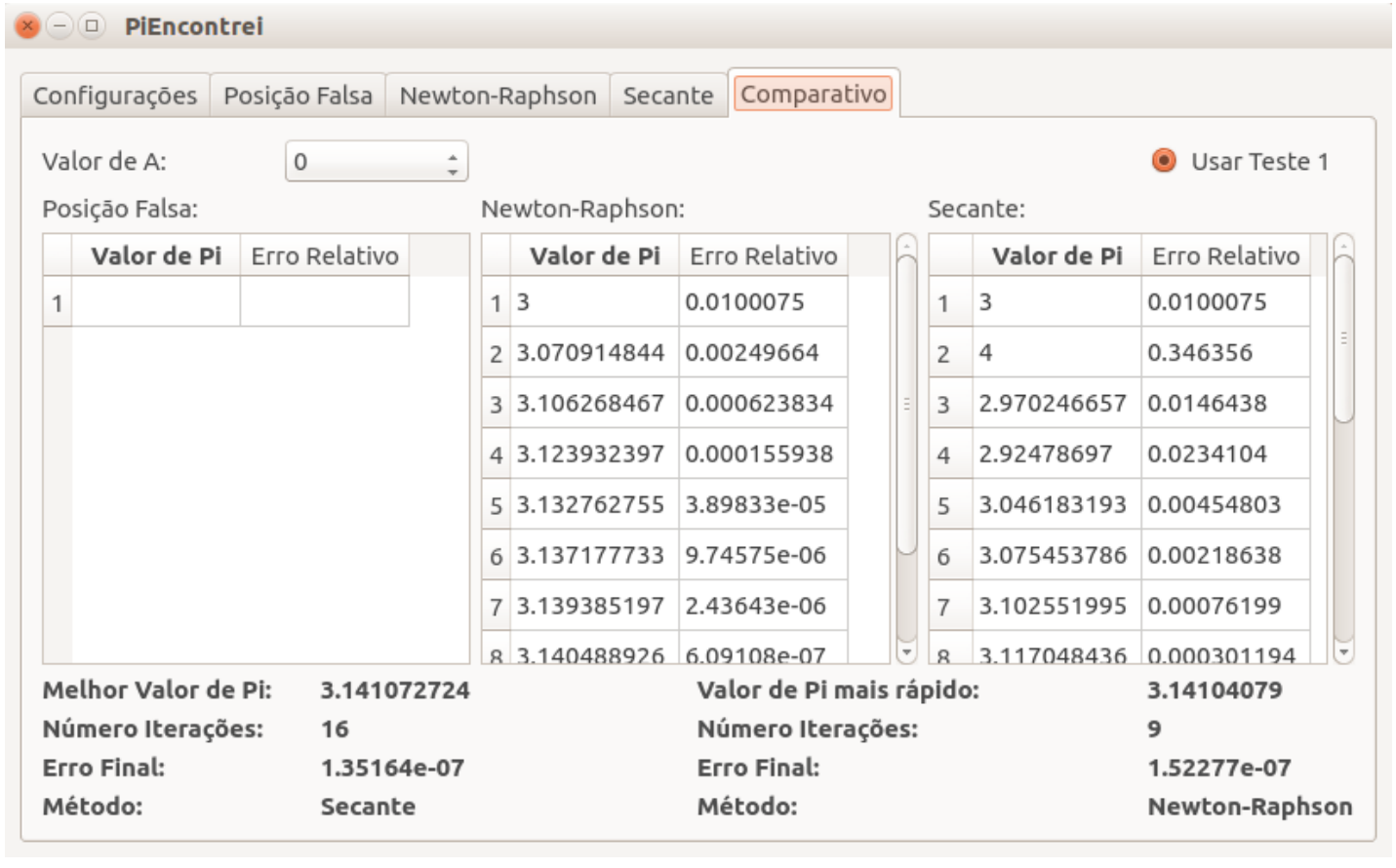
	Valor de Pi	Xa	F(Xa)	Xb	F(Xb)	Erro Absoluto
1	3	3	0.0100075…	3.070914844	0.0024966…	0.0709148
2	3.070914844	3.070914844	0.0024966…	3.106268467	0.0006238…	0.0353536
3	3.106268467	3.106268467	0.0006238…	3.123932397	0.0001559…	0.0176639
4	3.123932397	3.123932397	0.0001559…	3.132762755	3.8983302…	0.00883036
5	3.132762755	3.132762755	3.8983302…	3.137177733	9.7457464…	0.00441498
6	3.137177733	3.137177733	9.7457464…	3.139385197	2.4364316…	0.00220746
7	3.139385197	3.139385197	2.4364316…	3.140488926	6.0910760…	0.00110373
8	3.140488926	3.140488926	6.0910760…	3.14104079	1.5227688…	0.000551864
9	3.14104079	3.14104079	1.5227688…	3.141316722	3.8069219…	0.000275932

Interface



	Valor de Pi	Xa	F(Xa)	Xb	F(Xb)	Erro Absoluto
1	3	3	0.0100075…	4	0.3463563…	1
2	4	4	0.3463563…	2.970246657	0.0146438…	1.02975
3	2.970246657	2.970246657	0.0146438…	2.92478697	0.0234104…	0.0454597
4	2.92478697	2.92478697	0.0234104…	3.046183193	0.0045480…	0.121396
5	3.046183193	3.046183193	0.0045480…	3.075453786	0.0021863…	0.0292706
6	3.075453786	3.075453786	0.0021863…	3.102551995	0.0007619…	0.0270982
7	3.102551995	3.102551995	0.0007619…	3.117048436	0.0003011…	0.0144964
8	3.117048436	3.117048436	0.0003011…	3.126523883	0.0001135…	0.00947545
9	3.126523883	3.126523883	0.0001135…	3.132256327	4.3583177…	0.00573244

Interface


PiEncontrei
Configurações
Posição Falsa
Newton-Raphson
Secante
Comparativo
Valor de A:
0
Usar Teste 1
Posição Falsa:
Valor de Pi
Erro Relativo
1
Newton-Raphson:
Valor de Pi
Erro Relativo
1 3 0.0100075
2 3.070914844 0.00249664
3 3.106268467 0.000623834
4 3.123932397 0.000155938
5 3.132762755 3.89833e-05
6 3.137177733 9.74575e-06
7 3.139385197 2.43643e-06
8 3.140488926 6.09108e-07
Secante:
Valor de Pi
Erro Relativo
1 3 0.0100075
2 4 0.346356
3 2.970246657 0.0146438
4 2.92478697 0.0234104
5 3.046183193 0.00454803
6 3.075453786 0.00218638
7 3.102551995 0.00076199
8 3.117048436 0.000301194
Melhor Valor de Pi: 3.141072724
Número Iterações: 16
Erro Final: 1.35164e-07
Método: Secante
Valor de Pi mais rápido: 3.14104079
Número Iterações: 9
Erro Final: 1.52277e-07
Método: Newton-Raphson

Conclusão

- Dificuldades encontradas:
  - Ponta pé inicial
  - Cansaço

Monstro Imaginado

Monstro Encontrado.

Conclusão

- “Novidades” testadas nesse trabalho.
  - O uso de um Design Pattern.
  - Documentação só onde é realmente necessário.