Universidade de São Paulo
Instituto de Física de São Carlos

Lista 2

Pedro Calligaris Delbem 5255417

Professor: Attilio Cucchieri

Março de 2025

Sumário

1 Finding roots

1.1 Exercício 1

Tarefa: Demonstrar que no método de Newton-Raphson

$$x_{k+1} = x_k + \frac{f(x_k)}{f'(x_k)} \tag{1}$$

a convergência é quadrática.

Código Escrito:

```
!------------------------------------------------------------
! File: L2-5255417-ex-1.f90
!
! Description:
!   Computes Newton-Raphson method convergence
!
! Dependencies:
!   - None
!
! Since:
!   - 03/2025
!
! Authors:
!   - Pedro C. Delbem <pedrodelbem@usp.br>
!------------------------------------------------------------
program newton_raphson

    !deactivate implicit typing
    implicit none

    !define variables
    real x_k, x_kplus1, f_x_k, df_x_k, f_x(5)
    integer iteration

    write(*,*) "Insert x^4 coefficient:"
    read(*,*) f_x(5)
    write(*,*) "Insert x^3 coefficient:"
    read(*,*) f_x(4)
    write(*,*) "Insert x^2 coefficient:"
    read(*,*) f_x(3)
    write(*,*) "Insert x coefficient:"
    read(*,*) f_x(2)
    write(*,*) "Insert constant coefficient:"
    read(*,*) f_x(1)

    !initialize variables
    x_k = 10.0
    x_kplus1 = 0.0
    f_x_k = f(x_k,f_x)
    df_x_k = df(x_k,f_x)
    iteration = 1

    !open output file
```

```
    open(unit=1, file='output.txt', action='write')

    !print firstiteration
    write(1,*) iteration, f(x_k,f_x)

    !update f_x_k and df_x_k
    f_x_k = f(x_k,f_x)
    df_x_k = df(x_k,f_x)

    !update x_kplus1
    x_kplus1 = x_k - f_x_k/df_x_k

    !update iteration
    iteration = iteration + 1

    !print second iteration
    write(1,*) iteration, f(x_kplus1,f_x)

    !Newton-Raphson method
    do while (abs(x_kplus1 - x_k) > 1e-6)

        !update x_k
        x_k = x_kplus1

        !update f_x_k and df_x_k
        f_x_k = f(x_k,f_x)
        df_x_k = df(x_k,f_x)

        !update x_kplus1
        x_kplus1 = x_k - f_x_k/df_x_k

        !update iteration
        iteration = iteration + 1

        !print iteration
        write(1,*) iteration, f(x_kplus1,f_x)

    end do

    !print root
    write(*,*) 'Root:', x_kplus1

    !close output file
    close(1)

contains

    function f(x,f_x) result(result)
        real, intent(in) :: x
        real, intent(in) :: f_x(5)
        real result

        result = f_x(5)*x**4. + f_x(4)*x**3. + f_x(3)*x**2. + f_x
   (2)*x + f_x(1)
    end function f

    function df(x,f_x) result(result)
        real, intent(in) :: x
```

```
pedro@Pedro-Lenovo ~/Documentos/GitHub/quanticacomp/lista2
> $ ./L2-5255417-ex-1.exe
 Insert x^4 coefficient:
0
 Insert x^3 coefficient:
0
 Insert x^2 coefficient:
3
 Insert x coefficient:
0
 Insert constant coefficient:
-10
 Root:   1.82574189
```

Figura 1: Teste feito para gerar o gráfico

```
        real, intent(in) :: f_x(5)
        real result

        result = 4.*f_x(5)*x**3. + 3.*f_x(4)*x**2. + 2.*f_x(3)*x + f_x(2)
    end function df

end program newton_raphson
```

Resultados:

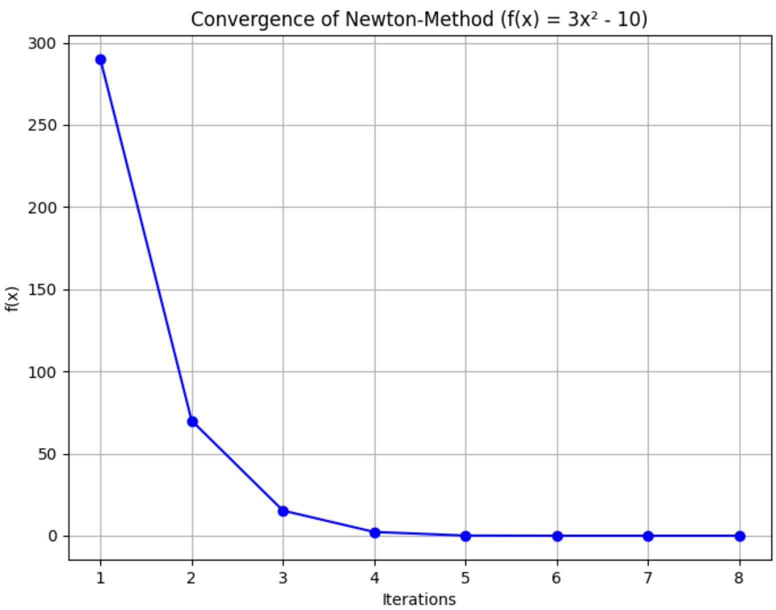

Figura 2: f(x) ao decorrer das iterações

Testando para o caso $f(x) = 3x^2 - 10 = 0$ percebe-se - pelo gráfico obtido - que o método Newton-Raphson converge quadraticamente.

1.2 Exercício 2

Tarefa: Achar as raízes das equações $f(x) = x^2 - 5 = 0$ e $f(x) = 5x^3 - 5x - 24 = 0$ usando os métodos de Newton-Raphson e da secante para diferentes chutes iniciais e diferentes condições de convergência.

Código Escrito:

```
!------------------------------------------------------------
! File: L2-5255417-ex-2.f90
!
! Description:
!
!
! Dependencies:
!   - None
!
! Since:
!   - 03/2025
!
! Authors:
!   - Pedro C. Delbem <pedrodelbem@usp.br>
!------------------------------------------------------------
program find_roots

    !deactivate implicit typing
    implicit none

    !define variables
    real x_kminus1, x_k, x_kplus1, f_x_kminus1, f_x_k, df_x_k, f_x
   (5), initial_guess
    integer iteration

    !request coefficients
    write(*,*) "Insert x^4 coefficient:"
    read(*,*) f_x(5)
    write(*,*) "Insert x^3 coefficient:"
    read(*,*) f_x(4)
    write(*,*) "Insert x^2 coefficient:"
    read(*,*) f_x(3)
    write(*,*) "Insert x coefficient:"
    read(*,*) f_x(2)
    write(*,*) "Insert constant coefficient:"
    read(*,*) f_x(1)

    !request initial guess
    write(*,*) "Insert initial guess:"
    read(*,*) initial_guess

    !initialize variables
    x_k = initial_guess
    x_kplus1 = 0.0
    f_x_k = f(x_k,f_x)
    df_x_k = df(x_k,f_x)
    iteration = 1

    !open first output file
```

```
    open(unit=1, file='output1.txt', action='write')

    !print header
    write(1,*) 'Newton-Raphson Method'
    write(1,*) 'Initial guess: ', x_k
    write(1,*) 'f(x) = ',f_x(5),f_x(4),f_x(3),f_x(2),f_x(1)

    !print first iteration
    write(1,*) iteration, f(x_kplus1,f_x)

    !update f_x and df_x
    f_x_k = f(x_k,f_x)
    df_x_k = df(x_k,f_x)

    !update x_kplus1
    x_kplus1 = x_k - f_x_k/df_x_k

    !update iteration
    iteration = iteration + 1

    !print second iteration
    write(1,*) iteration, f(x_kplus1,f_x)

    !Newton-Raphson method
    do while (abs(x_kplus1 - x_k) > 1e-6)

        !update x_k
        x_k = x_kplus1

        !update f_x and df_x
        f_x_k = f(x_k,f_x)
        df_x_k = df(x_k,f_x)

        !update x_kplus1
        x_kplus1 = x_k - f_x_k/df_x_k

        !update iteration
        iteration = iteration + 1

        !print iteration
        write(1,*) iteration, f(x_kplus1,f_x)

    end do

    !print root
    write(*,*) 'Root:', x_kplus1

    !close first output file
    close(1)

    !reinitialize variables
    x_kminus1 = initial_guess - 1.0
    x_k = initial_guess
    x_kplus1 = 0.0
    f_x_kminus1 = f(x_kminus1,f_x)
    f_x_k = f(x_k,f_x)
    iteration = 1

```

```
    !open second output file
    open(unit=2, file='output2.txt', action='write')

    !print header
    write(2,*) 'Secant Method'
    write(2,*) 'Initial guess: ', x_k
    write(2,*) 'f(x) = ',f_x(5),f_x(4),f_x(3),f_x(2),f_x(1)

    !print first iteration
    write(2,*) iteration, f(x_kplus1,f_x)

    !update f_x_k and f_x_k-1
    f_x_kminus1 = f(x_kminus1,f_x)
    f_x_k = f(x_k,f_x)

    !update x_kplus1 and x_kminus1
    x_kplus1 = x_k - f_x_k*(x_k - x_kminus1)/(f_x_k - f_x_kminus1)
    x_kminus1 = x_k

    !update iteration
    iteration = iteration + 1

    !print second iteration
    write(2,*) iteration, f(x_kplus1,f_x)

    !Secant method
    do while (abs(x_kplus1 - x_k) > 1e-6)

        !update x_k
        x_k = x_kplus1

        !update f_x_k and f_x_k-1
        f_x_kminus1 = f(x_kminus1,f_x)
        f_x_k = f(x_k,f_x)

        !update x_kplus1 and x_kminus1
        x_kplus1 = x_k - f_x_k*(x_k - x_kminus1)/(f_x_k -
   f_x_kminus1)
        x_kminus1 = x_k

        !update iteration
        iteration = iteration + 1

        !print iteration
        write(2,*) iteration, f(x_kplus1,f_x)

    end do

    !print root
    write(*,*) 'Root:', x_kplus1

    !close second output file
    close(2)

contains

    function f(x,f_x) result(result)
        real, intent(in) :: x
```

```fortran
        real, intent(in) :: f_x(5)
        real result

        result = f_x(5)*x**4. + f_x(4)*x**3. + f_x(3)*x**2. + f_x(2)*x + f_x(1)
    end function f

    function df(x,f_x) result(result)
        real, intent(in) :: x
        real, intent(in) :: f_x(5)
        real result

        result = 4.*f_x(5)*x**3. + 3.*f_x(4)*x**2. + 2.*f_x(3)*x + f_x(2)
    end function df

end program find_roots
```

Resultados:

1.2.1 Raizes de $f(x) = x^2 - 5 = 0$

Para as seguintes informações iniciais:

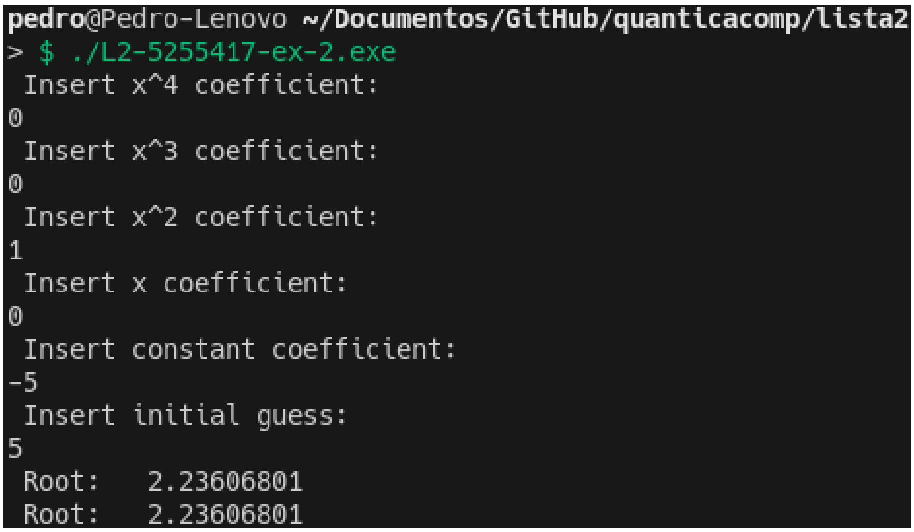

Figura 3: Teste feito para gerar o gráfico onde as duas últimas linhas são as raizes encontradas pelos métodos de Newton-Raphson e da secante, respectivamente

Obteve-se o valor da raiz = 2.23606801 para ambos os métodos.

Com o seguinte gráfico para o método de Newton-Raphson:

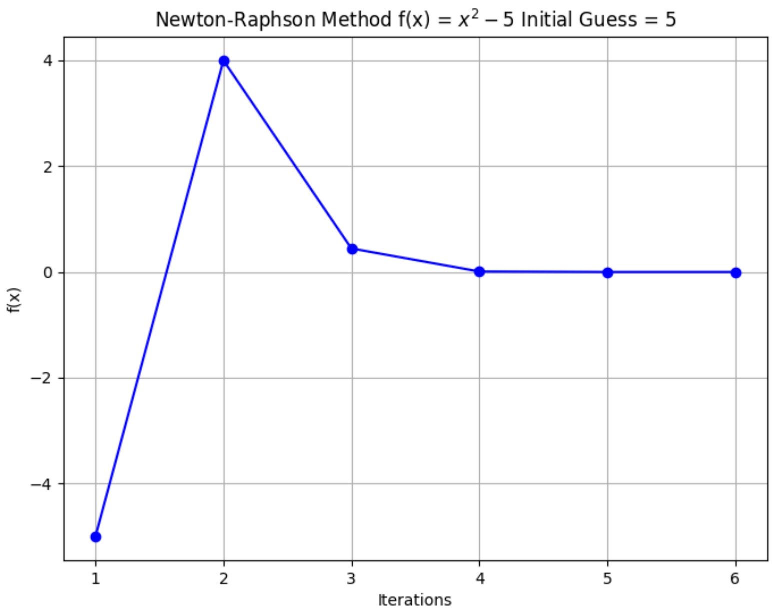

Figura 4: f(x) ao decorrer das iterações

Com o seguinte gráfico para o método da secante:

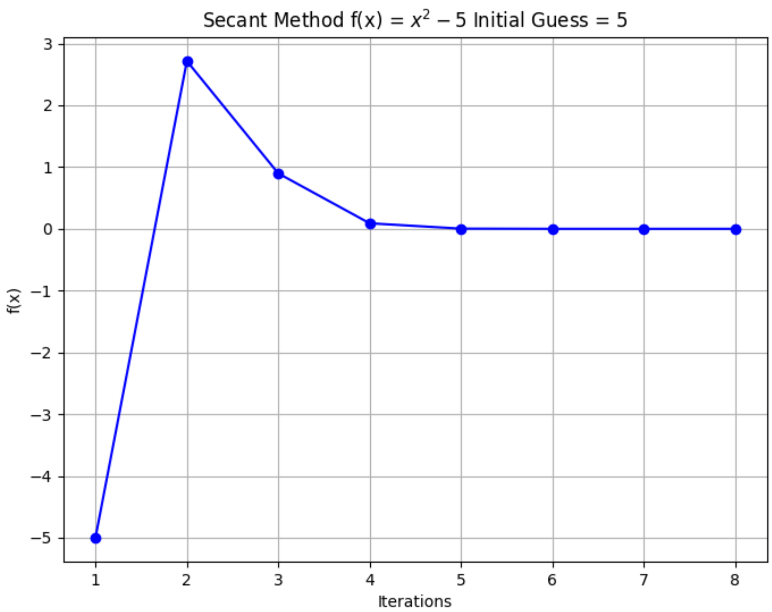

Figura 5: f(x) ao decorrer das iterações

Para as seguintes informações iniciais:

```
pedro@Pedro-Lenovo ~/Documentos/GitHub/quanticacomp/lista2
> $ ./L2-5255417-ex-2.exe
 Insert x^4 coefficient:
0
 Insert x^3 coefficient:
0
 Insert x^2 coefficient:
1
 Insert x coefficient:
0
 Insert constant coefficient:
-5
 Insert initial guess:
10
 Root:   2.23606801
 Root:   2.23606801
```

Figura 6: Teste feito para gerar o gráfico onde as duas últimas linhas são as raizes encontradas pelos métodos de Newton-Raphson e da secante, respectivamente

Obteve-se o valor da raiz = 2.23606801 para ambos os métodos.

Com o seguinte gráfico para o método de Newton-Raphson:

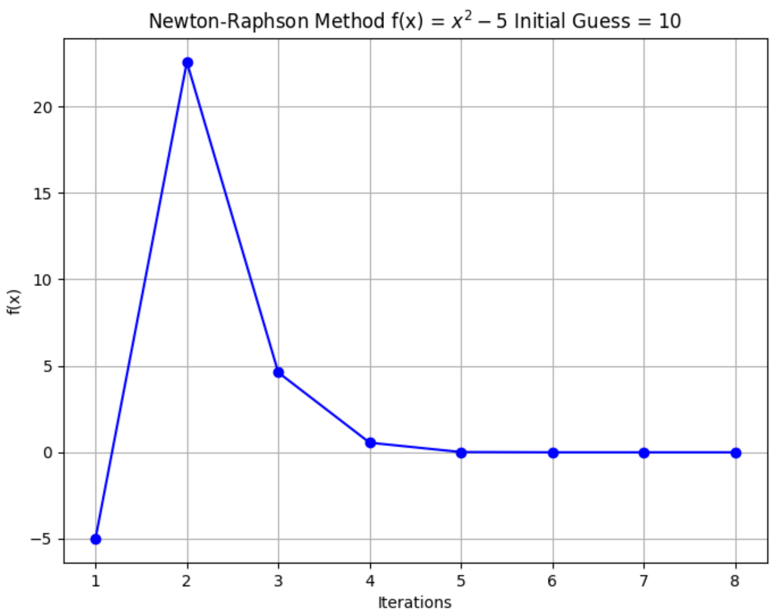

Figura 7: f(x) ao decorrer das iterações

Com o seguinte gráfico para o método da secante:

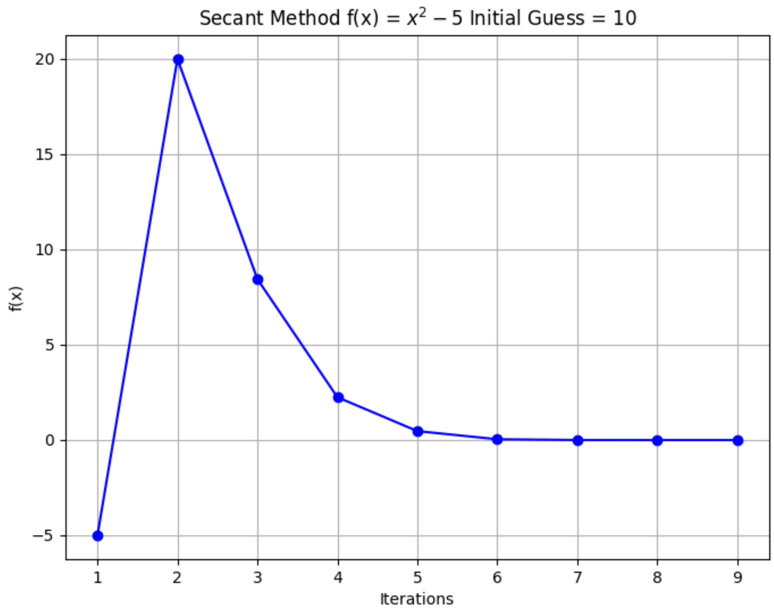

Figura 8: f(x) ao decorrer das iterações

Para as seguintes informações iniciais:

```
pedro@Pedro-Lenovo ~/Documentos/GitHub/quanticacomp/lista2
> $ ./L2-5255417-ex-2.exe
 Insert x^4 coefficient:
0
 Insert x^3 coefficient:
0
 Insert x^2 coefficient:
1
 Insert x coefficient:
0
 Insert constant coefficient:
-5
 Insert initial guess:
20
 Root:   2.23606801
 Root:   2.23606801
```

Figura 9: Teste feito para gerar o gráfico onde as duas últimas linhas são as raizes encontradas pelos métodos de Newton-Raphson e da secante, respectivamente

Obteve-se o valor da raiz = 2.23606801 para ambos os métodos.

Com o seguinte gráfico para o método de Newton-Raphson:

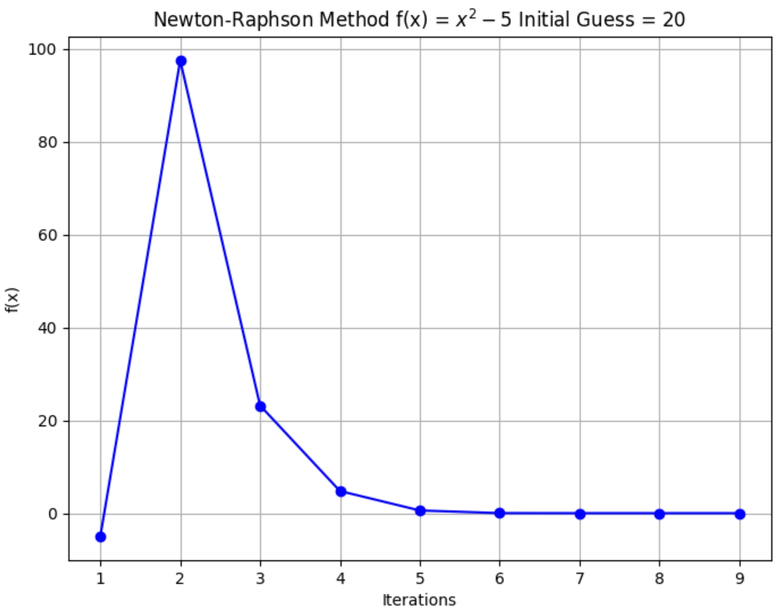

Figura 10: f(x) ao decorrer das iterações

Com o seguinte gráfico para o método da secante:

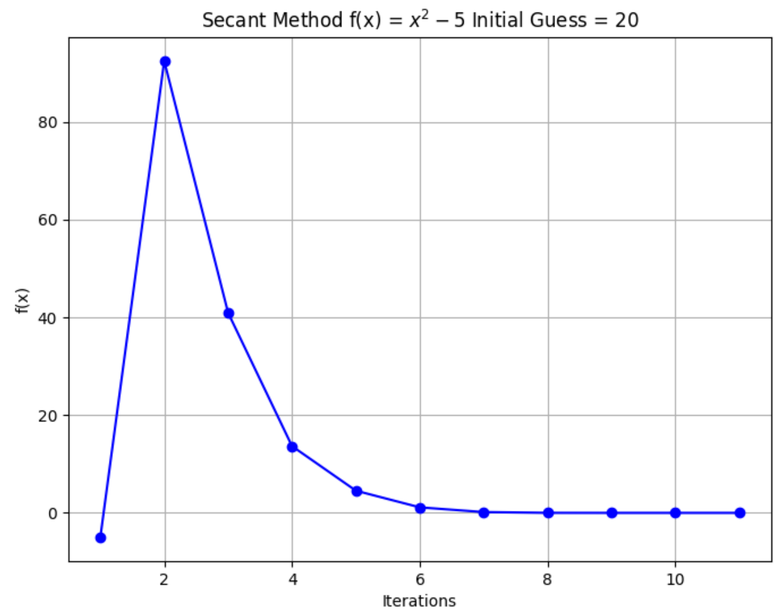

Figura 11: f(x) ao decorrer das iterações

Percebe-se que ambos os métodos foram suficientes para encontrar as raizes da função. Contudo o método de Newton-Raphson foi mais eficiente em encontrar em menos iterações as raizes da função.

1.2.2 Raizes de $f(x) = 5x^3 - 5x - 24 = 0$

Para as seguintes informações iniciais:

```
pedro@Pedro-Lenovo ~/Documentos/GitHub/quanticacomp/lista2
> $ ./L2-5255417-ex-2.exe
 Insert x^4 coefficient:
0
 Insert x^3 coefficient:
5
 Insert x^2 coefficient:
0
 Insert x coefficient:
-5
 Insert constant coefficient:
-24
 Insert initial guess:
5
 Root:   1.88367081
 Root:   1.88367081
```

Figura 12: Teste feito para gerar o gráfico onde as duas últimas linhas são as raizes encontradas pelos métodos de Newton-Raphson e da secante, respectivamente

Obteve-se o valor da raiz = 1.88367081 para ambos os métodos.

Com o seguinte gráfico para o método de Newton-Raphson:

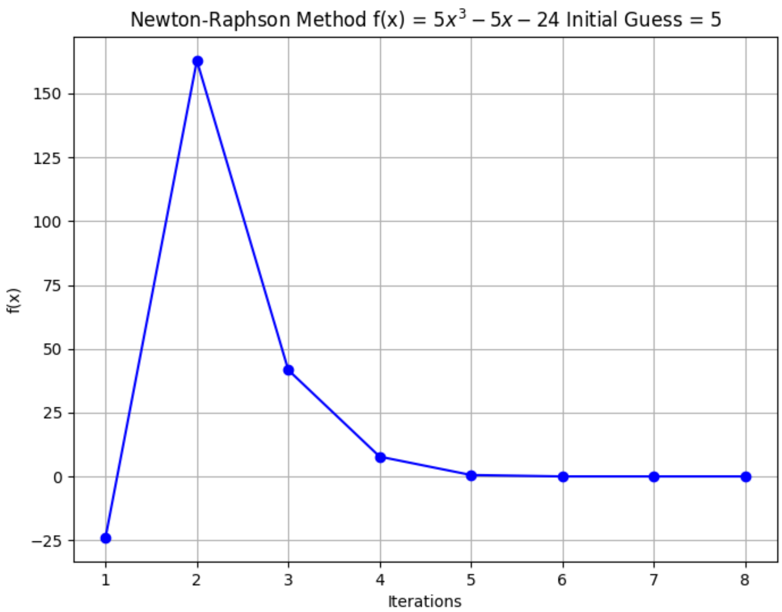

Figura 13: f(x) ao decorrer das iterações

Com o seguinte gráfico para o método da secante:

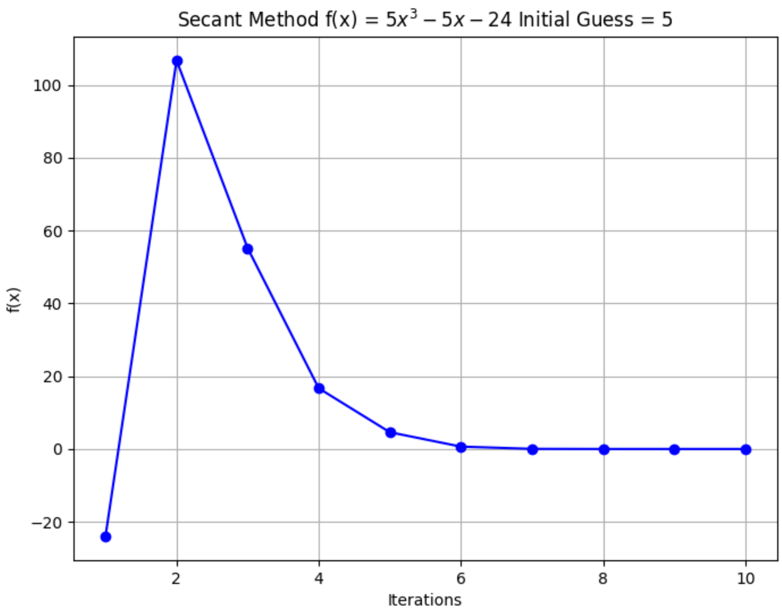

Figura 14: f(x) ao decorrer das iterações

Para as seguintes informações iniciais:

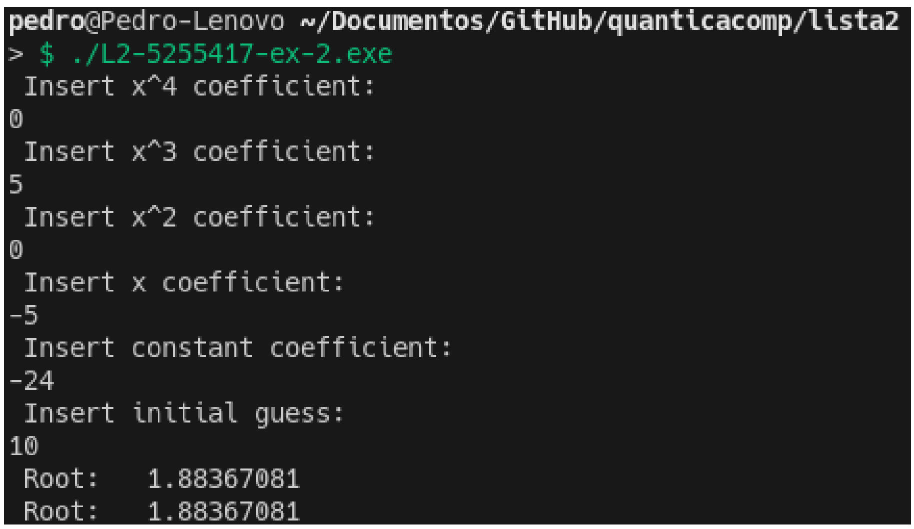

Figura 15: Teste feito para gerar o gráfico onde as duas últimas linhas são as raizes encontradas pelos métodos de Newton-Raphson e da secante, respectivamente

Obteve-se o valor da raiz = 1.88367081 para ambos os métodos.

Com o seguinte gráfico para o método de Newton-Raphson:

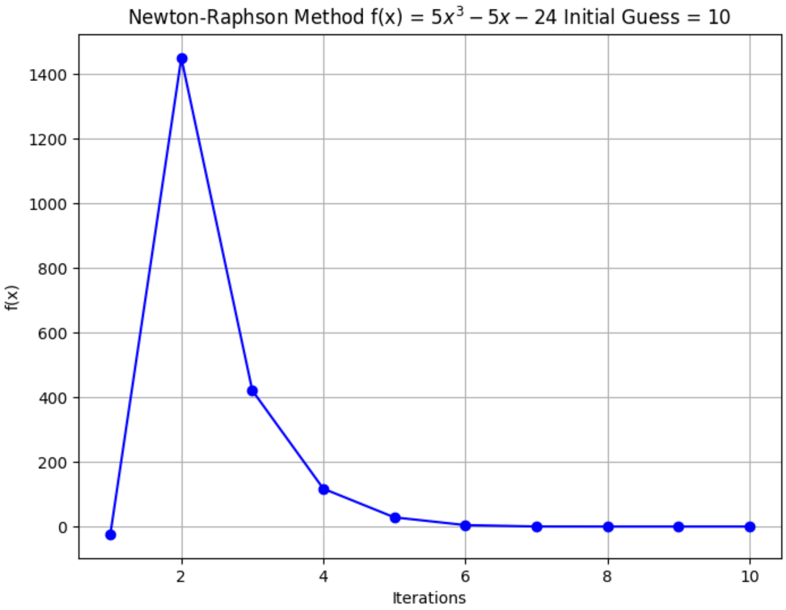

Figura 16: f(x) ao decorrer das iterações

Com o seguinte gráfico para o método da secante:

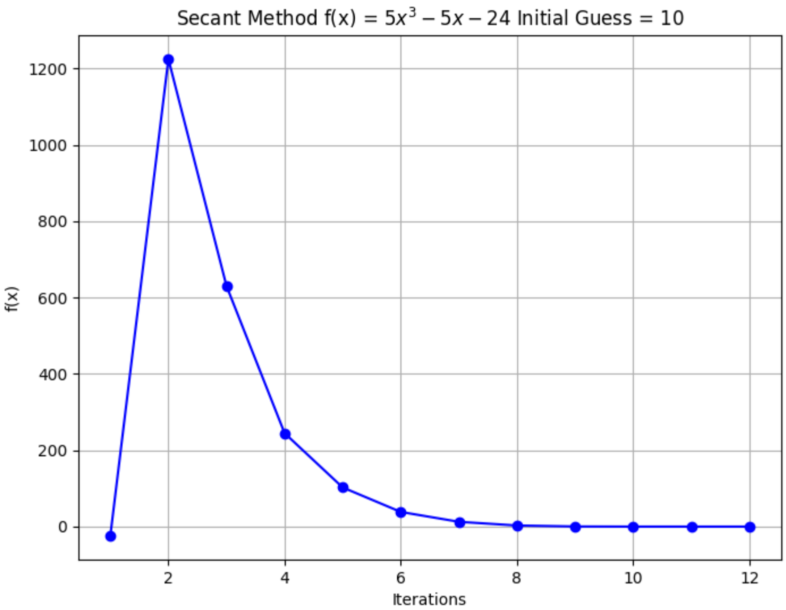

Figura 17: f(x) ao decorrer das iterações

Para as seguintes informações iniciais:

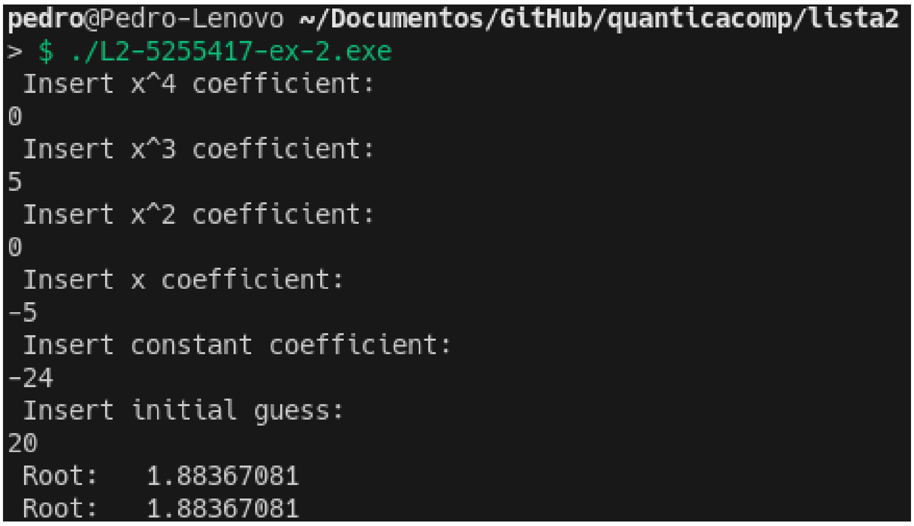

Figura 18: Teste feito para gerar o gráfico onde as duas últimas linhas são as raizes encontradas pelos métodos de Newton-Raphson e da secante, respectivamente

Obteve-se o valor da raiz = 1.88367081 para ambos os métodos.

Com o seguinte gráfico para o método de Newton-Raphson:

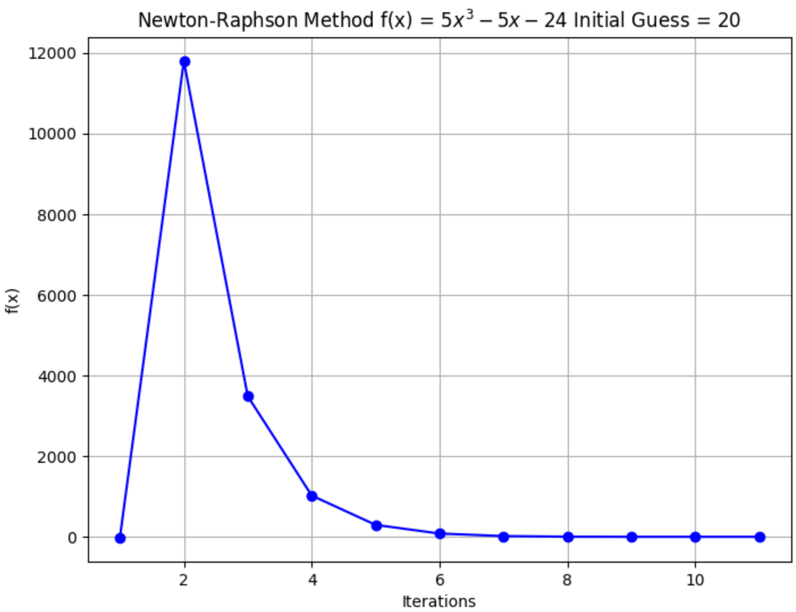

Figura 19: f(x) ao decorrer das iterações

Com o seguinte gráfico para o método da secante:

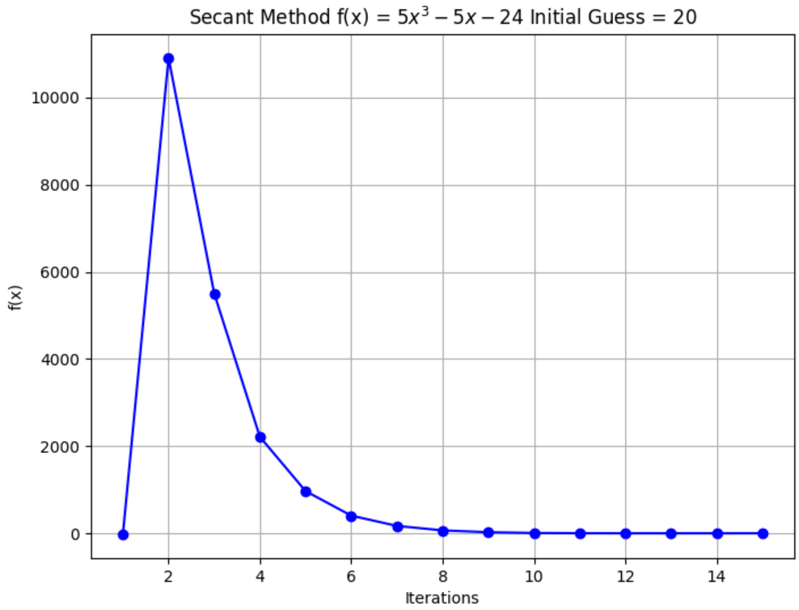

Figura 20: f(x) ao decorrer das iterações

Percebe-se que ambos os métodos foram suficientes para encontrar as raizes da função. Contudo o método de Newton-Raphson foi mais eficiente em encontrar em menos iterações as raizes da função.

2 Eigenvalues of the wave equation

2.1 Exercício 1

Tarefa: Achar a precisão do computador, i.e., o maior número positivo tal que $1 + \epsilon = 1$, usando precisão simples e dupla.

Código Escrito:

```
!------------------------------------------------------------
! File: L2-5255417-ex-3.f90
!
! Description:
!
!
! Dependencies:
!   - None
!
! Since:
!   - 03/2025
!
! Authors:
!   - Pedro C. Delbem <pedrodelbem@usp.br>
```

```
!------------------------------------------------------------
```

Resultados:

Nota-se que os valores obtidos correspondem aos valores esperados.

2.2 Exercício 2

Tarefa: Calcular

$$e^{-x} = 1 - x + x^2/2! - x^3/3! + ... \tag{2}$$

para x = 0.1, 1, 10, 100 e 1000 com um erro menor do que 10^{-8}. Problema: quando truncar a série? É preciso calcular o fatorial explicitamente? Comparar o valor obtido usando a série com o resultado exato.

Código Escrito:

```
!------------------------------------------------------------
! File: L2-5255417-ex-4.f90
!
! Description:
!
!
! Dependencies:
!   - None
!
! Since:
!   - 03/2025
!
! Authors:
!   - Pedro C. Delbem <pedrodelbem@usp.br>
!------------------------------------------------------------
```

Resultados:

Descrição: O código faz um loop onde, em cada iteração, calcula o valor da série para um valor de x diferente. Na subrotina "compute-exponencial"calcula-se a série definindo o próximo termo como a multiplicação do termo anterior por -x/n, pois - deste modo - não se faz necessário calcular o fatorial explicitamente. Ademais, interrompe-se a soma quando o termo atual for menor que 10^{-8} garantindo a precisão desejada, uma vez que cada termo da série é menor - em módulo - que o anterior.

Por fim, percebe-se que o resultado eperado foi obtido para todos os casos testados - com exeção dos casos onde o resultado é menor que a precisão.

2.3 Exercício 3

Tarefa: Considerar a somatória

$$\Sigma(N) = \sum_{n=1}^{2N} (-1)^n \frac{n}{n+1} = -\sum_{n=1}^{N} \frac{2n-1}{2n} + \sum_{n=1}^{N} \frac{2n}{2n+1} = \sum_{n=1}^{N} \frac{1}{2n(2n+1)} \tag{3}$$

e calcular $\Sigma(N)$ para N = 1, 2, . . . , 10^6 usando as três fórmulas acima. Comparar os resultados usando precis ão simples.

Código Escrito:

```
!--------------------------------------------------------------
! File: L2-5255417-ex-5.f90
!
! Description:
!
!
! Dependencies:
!   - None
!
! Since:
!   - 03/2025
!
! Authors:
!   - Pedro C. Delbem <pedrodelbem@usp.br>
!--------------------------------------------------------------
```

Resultados:

Nota-se que a primeira série demora é mais instável do que as demais. Além disso, a terceira série se mostra mais estável que a segunda - sendo então a melhor versão.