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.

Expandimos f(x) em torno de $x_n - r$ - onde r é a raiz de f(x) - e obtemos:

$$f(x_n) = f(r) + f'(r)(x_n - r) + \frac{1}{2}f''(r)(x_n - r)^2 + O(x_n - r)^3 \tag{2}$$

E como f(r) = 0, obtemos:

$$f(x_n) = f'(r)(x_n - r) + \frac{1}{2}f''(r)(x_n - r)^2 + O(x_n - r)^3 \tag{3}$$

Expande-se, também, $f'(x_n)$ e obtemos:

$$f'(x_n) = f'(r) + f''(x_n - r)(x_n - r) + O(xn - r)^2 \tag{4}$$

Substituindo em $x_{n+1} = x_n + \frac{f(x_n)}{f'(x_n)}$ obtemos:

$$x_{n+1} = x_n - \frac{f'(r)(x_n - r) + \frac{1}{2}f''(r)(x_n - r)^2}{f'(r) + f''(x_n - r)(x_n - r)} \tag{5}$$

Subtraindo r de ambos os lados:

$$x_{n+1} - r = x_n - r - \frac{f'(r)(x_n - r) + \frac{1}{2}f''(r)(x_n - r)^2}{f'(r) + f''(x_n - r)(x_n - r)} \tag{6}$$

Colocando o termo $x_n - r$ em evidência:

$$x_{n+1} - r = (x_n - r)\left[1 - \frac{f'(r) + \frac{1}{2}f''(r)(x_n - r)}{f'(r) + f''(x_n - r)(x_n - r)}\right] \tag{7}$$

Para $x_n - r$ pequeno, f"(r)$(x_n - r)$ é disprezível e assim o desprezamos no denominador - obtendo:

$$x_{n+1} - r = (x_n - r)\left[1 - \frac{f'(r) + \frac{1}{2}f''(r)(x_n - r)}{f'(r)}\right] \tag{8}$$

Isolando $x_n - r$:

$$x_{n+1} - r = -(x_n - r)^2\left[\frac{\frac{1}{2}f''(r)}{f'(r)}\right] \tag{9}$$

Rearranjando:

$$r - x_{n+1} = (r - x_n)^2\left[\frac{\frac{1}{2}f''(r)}{f'(r)}\right] \tag{10}$$

Como $r - x_n$ é o erro cometido na n-éssima iteração e $r - x_{n+1}$ é o erro cometido na n+1-éssima iteração, temos que o erro da iteração n+1 é proporcional ao quadrado do erro da iteração n e portanto a convergência é quadratica.

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:

```fortran
!---------------------------------------------------------------
! 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
```

```fortran
    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

```

```fortran
    !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
```

O código foi compilado com o comando:

```
gfortran L2-5255417-ex-2.f90 -o L2-5255417-ex-2.exe
```

Resultados:

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

Para as seguintes informações iniciais:

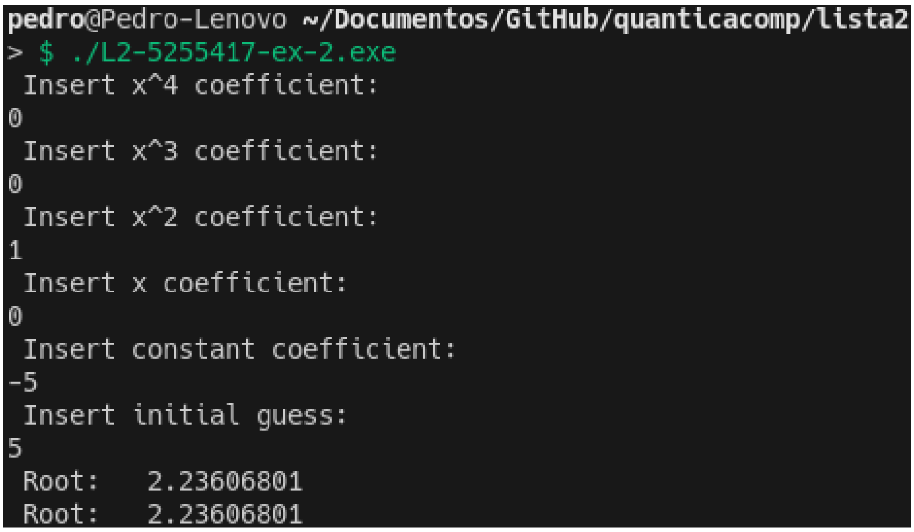


Figura 1: Teste feito para gerar o gráfico onde as duas últimas linhas são as raizes encontradas pelos métodos de Newton-Raphson e da secante, respectivamente

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

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

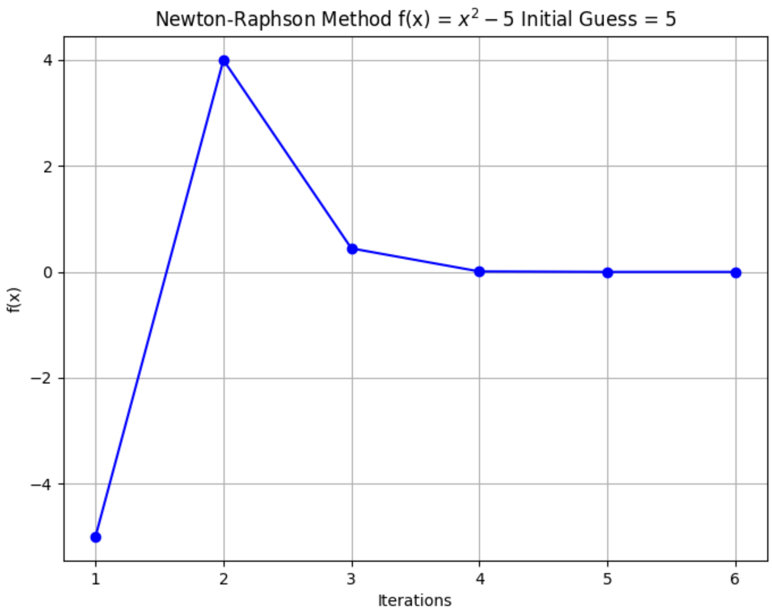


Figura 2: f(x) ao decorrer das iterações

Com o seguinte gráfico para o método da secante:

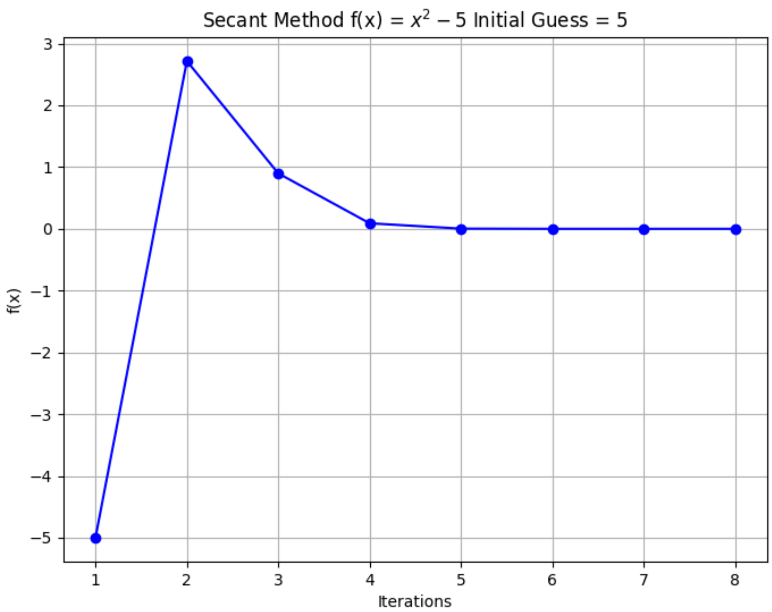
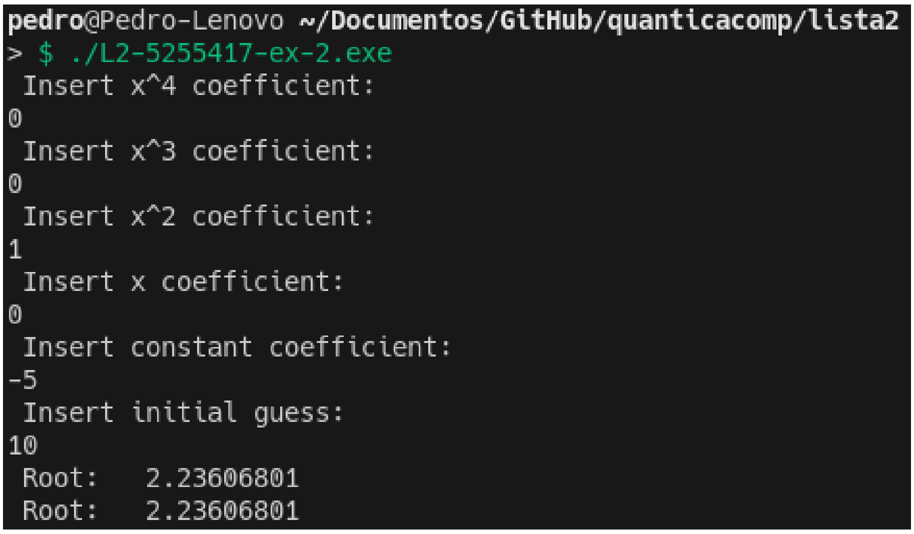


Figura 3: f(x) ao decorrer das iterações

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 4: 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:

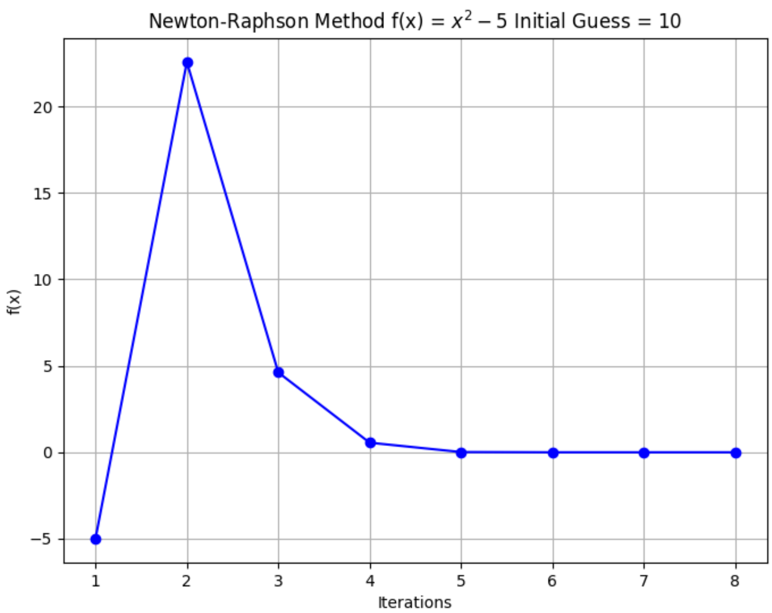


Figura 5: f(x) ao decorrer das iterações

Com o seguinte gráfico para o método da secante:

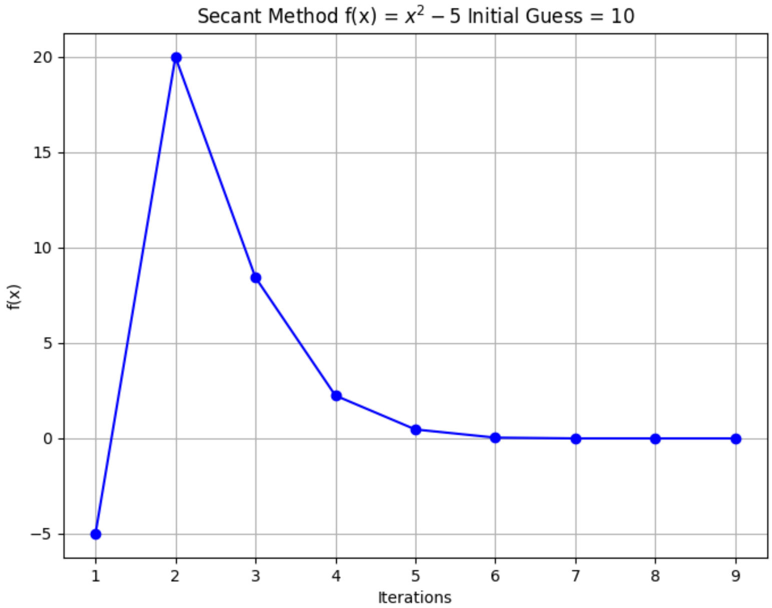


Figura 6: f(x) ao decorrer das iterações

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 7: 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:

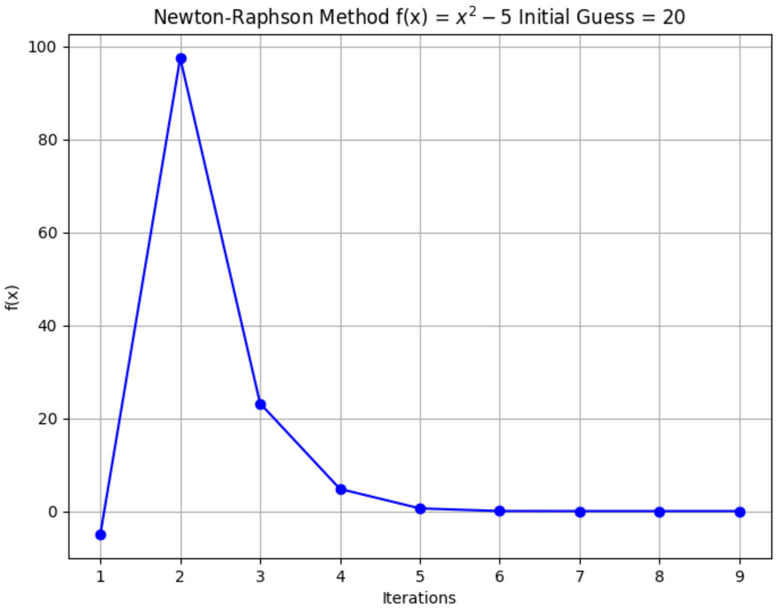


Figura 8: f(x) ao decorrer das iterações

Com o seguinte gráfico para o método da secante:

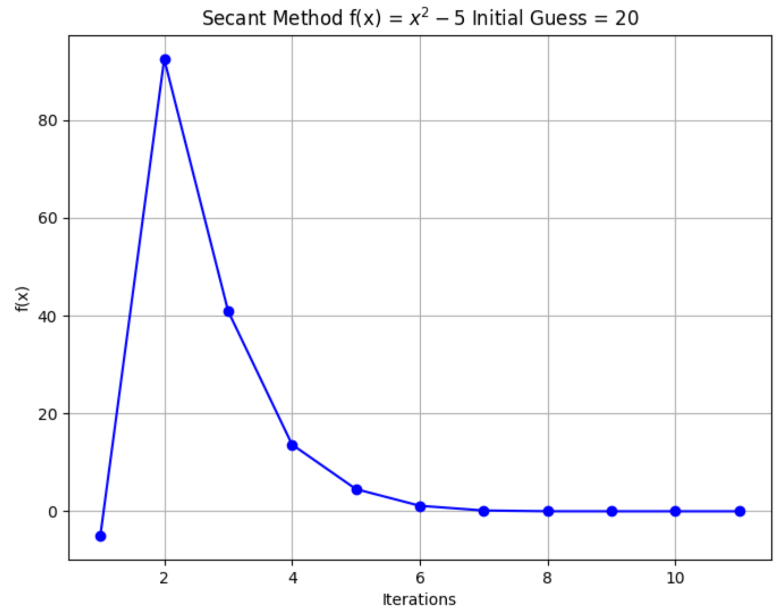


Figura 9: f(x) ao decorrer das iterações

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 10: 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:

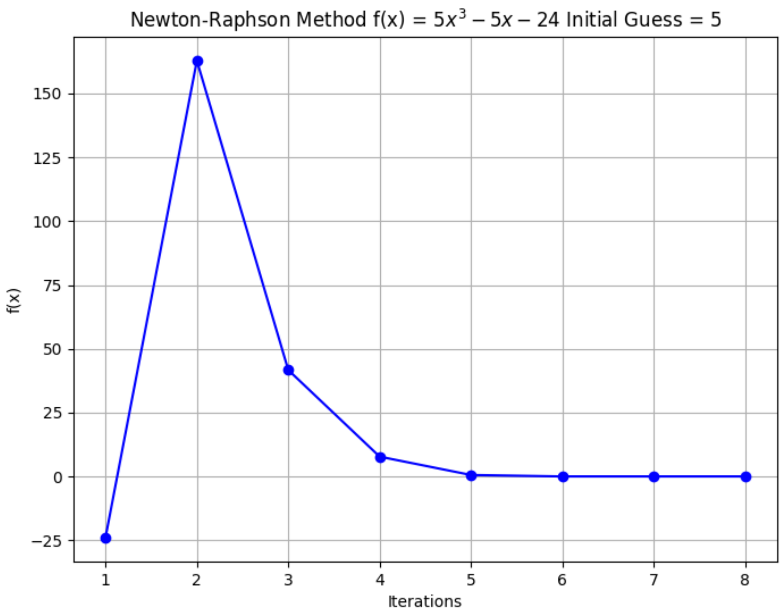


Figura 11: f(x) ao decorrer das iterações

Com o seguinte gráfico para o método da secante:

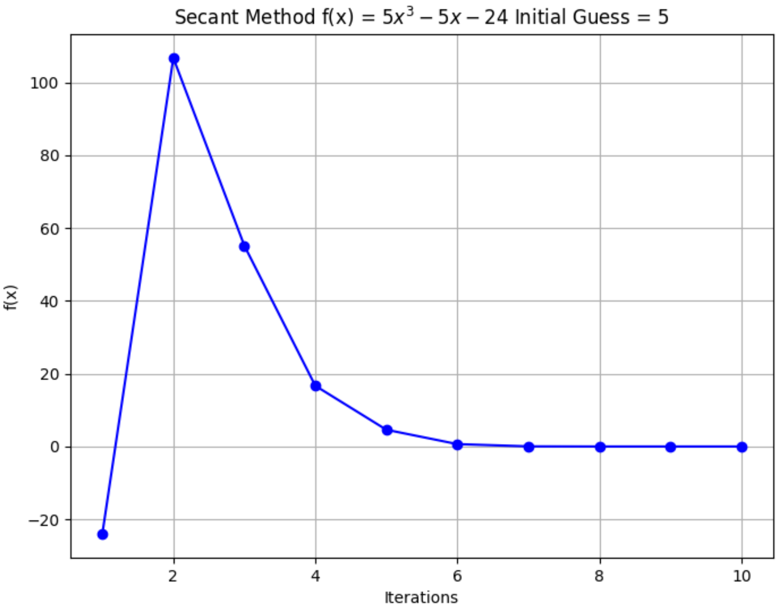


Figura 12: f(x) ao decorrer das iterações

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:
5
 Insert x^2 coefficient:
0
 Insert x coefficient:
-5
 Insert constant coefficient:
-24
 Insert initial guess:
10
 Root:   1.88367081
 Root:   1.88367081
```

Figura 13: 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:

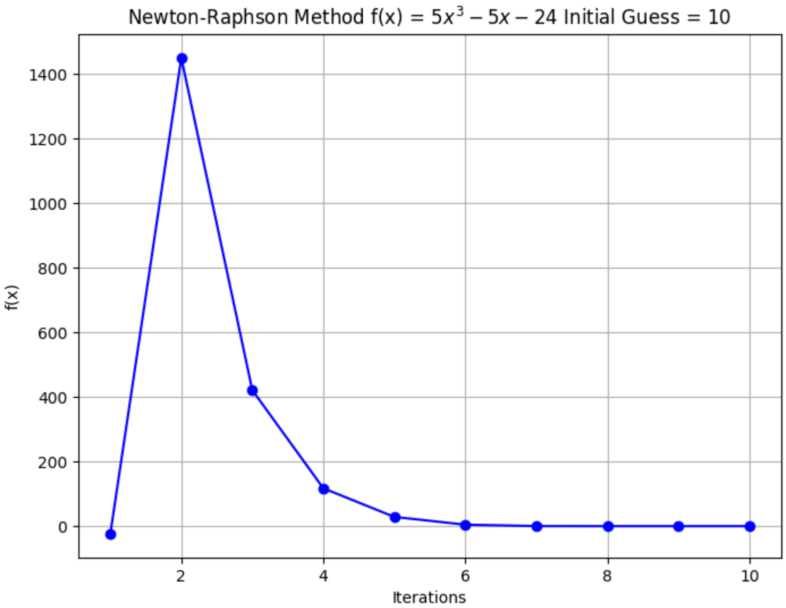


Figura 14: f(x) ao decorrer das iterações

Com o seguinte gráfico para o método da secante:

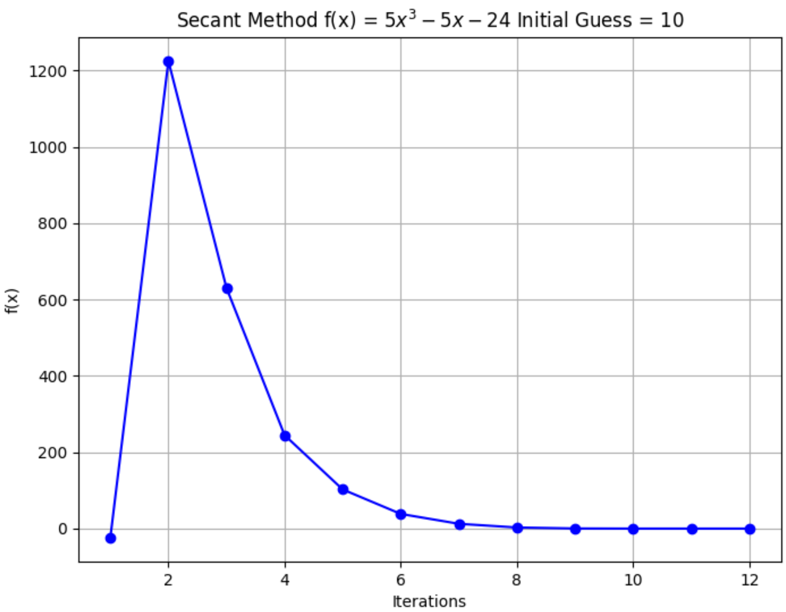


Figura 15: f(x) ao decorrer das iterações

Para as seguintes informações iniciais:

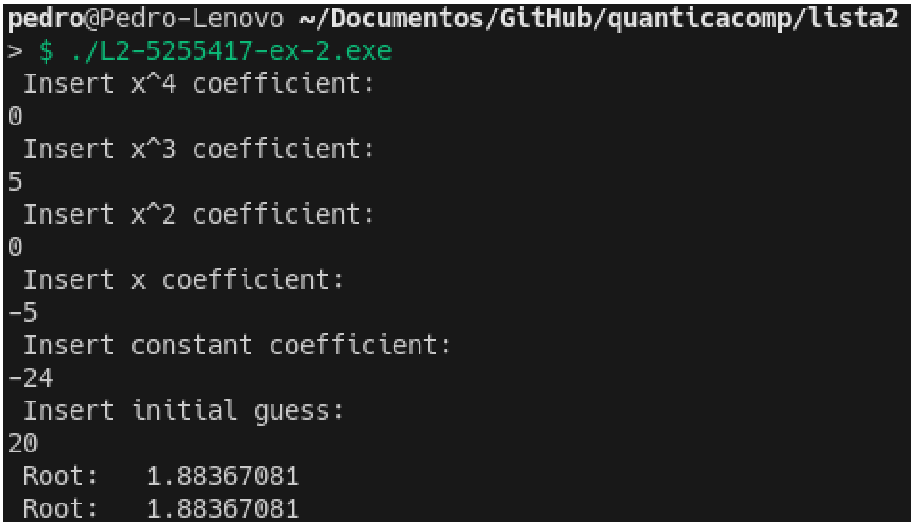


Figura 16: Teste feito para gerar o gráfico onde as duas últimas linhas são as raizes encontradas pelos métodos de Newton-Raphson e da secante, respectivamente

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

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

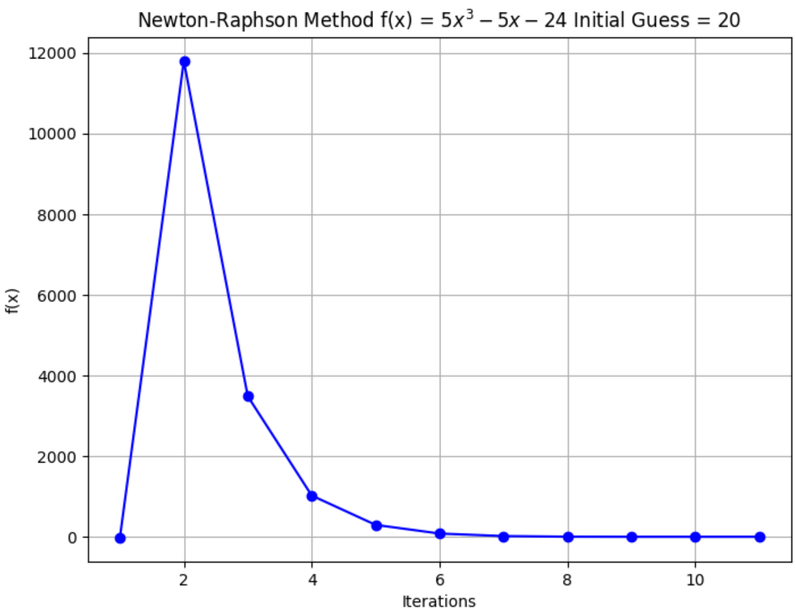


Figura 17: f(x) ao decorrer das iterações

Com o seguinte gráfico para o método da secante:

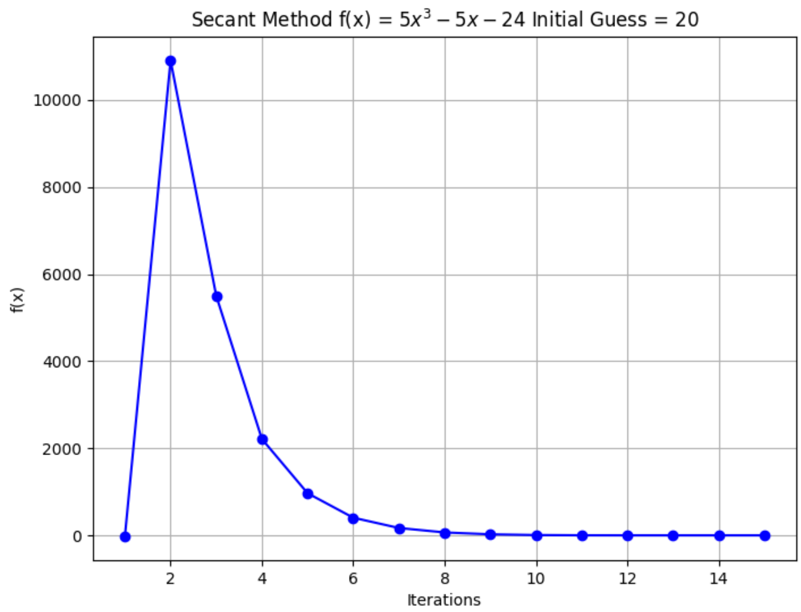


Figura 18: f(x) ao decorrer das iterações

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. Apesar da maior efiência do método de Newton-Raphson, o método da secante é útil para casos onde a derivada da função é complicada de se calcular - ou até mesmo não existe derivada.

2 Eigenvalues of the wave equation

2.1 Exercício 3

Tarefa: Escreva a transformação que permitem escrever a equação de Schrödinger para os autoestados de uma partícula em um poço infinito na forma

$$\frac{d^2}{dx^2}\psi(x) = -k^2\psi(x) \quad \text{com} \quad \psi(0) = 0 \text{ e } \psi(\infty) = 0 \tag{11}$$

Seja a equação de Schrödinger:

$$\left(-\frac{\hbar^2}{2m}\frac{d^2}{dx^2} + V(x)\right)\psi(x) = E\psi(x) \tag{12}$$

Para o caso do poço infinito a equação pode ser escrita como:

$$-\frac{\hbar^2}{2m}\frac{d^2}{dx^2}\psi(x) = E\psi(x) \quad \text{com} \quad 0 \leq x \leq L \tag{13}$$

Para tornar adimensional, fazemos a transformação $x \longrightarrow x/L$:

$$-\frac{\hbar^2}{2mL^2}\frac{d^2}{dx^2}\psi(x) = E\psi(x) \quad \text{com} \quad 0 \leq x \leq 1 \tag{14}$$

Rearranjando:

$$\frac{d^2}{dx^2}\psi(x) = -\frac{2mEL^2}{\hbar^2}\psi(x) \quad \text{com} \quad 0 \leq x \leq 1 \tag{15}$$

Note que $\frac{2mEL^2}{\hbar^2}$ é adimensional - como desejado. Então, definimos $k^2 = \frac{2mEL^2}{\hbar^2}$ - obtendo:

$$\frac{d^2}{dx^2}\psi(x) = -k^2\psi(x) \quad \text{com} \quad \psi(0) = 0 \text{ e } \psi(\infty) = 0 \tag{16}$$

que é a equação adimensional desejada.

2.2 Exercício 4

Tarefa: Escreva um código para calcular os primeiros três níveis de energia para o poço de potencial infinito, usando o shooting method e as condições de contorno $\psi(0) = 0$ e $\psi'(0)6 \neq 0$. Compare o resultado com a solução exata.

Código Escrito:

```
!--------------------------------------------------------------
! File: L2-5255417-ex-4.f90
!
! Description:
!   Find first three energy levels of quantum 1D infity square well
    using shooting method
!
! Dependencies:
!   - None
!
! Since:
!   - 03/2025
!
! Authors:
!   - Pedro C. Delbem <pedrodelbem@usp.br>
!--------------------------------------------------------------
program shooting_method

    !deactivate implicit typing
    implicit none

    !define variables
    real deltax, deltak, phi_deltax, dphi_0, k, phi_xminus1, phi_x,
    phi_xplus1, x
    integer i, number_of_iterations

    !define constants
    real phi_0

    !define phi(0)
    phi_0 = 0.0
```

```

    !initialize x
    x = 0.0

    write(*,*) "Insert the number of iterations:"
    read(*,*) number_of_iterations

    write(*,*) "Insert k:"
    read(*,*) k

    write(*,*) "Insert deltak:"
    read(*,*) deltak

    write(*,*) "Insert phi_deltax (non zero):"
    read(*,*) phi_deltax
    do while (phi_deltax == 0.0)
        write(*,*) "phi_deltax cannot be zero, input again"
        read(*,*) phi_deltax
    end do

    !define deltax
    deltax = 1.0/number_of_iterations
    write(*,*) "deltax: ", deltax

    !initialize phi
    phi_x = 1.0

    !update k until phi(1) >= deltak
    do while (phi_x >= deltak)

        !do the first iteration
        phi_xminus1 = phi_0
        phi_x = phi_deltax
        phi_xplus1 = 2.0*phi_x - phi_xminus1 - (k**2.0)*(deltax
  **2.0)*phi_x
        x = deltax

        !update phi
        call update_phi(phi_xminus1, phi_x, phi_xplus1, k, deltax,
  number_of_iterations, x)

        !update k
        k = k + deltak

    end do

    write(*,*) "First energy level: ", k

    !update k
    k = k + 2.0*deltak

    !second level
    do while (phi_x >= deltak) !update k until phi(1) >= deltak

        !do the first iteration
        phi_xminus1 = phi_0
        phi_x = phi_deltax
        phi_xplus1 = 2.0*phi_x - phi_xminus1 - (k**2.0)*(deltax
```

```
**2.0)*phi_x
        x = deltax

        !update phi
        call update_phi(phi_xminus1, phi_x, phi_xplus1, k, deltax,
number_of_iterations, x)

        !update k
        k = k + deltak

    end do

    write(*,*) "Second energy level: ", k

    !update k
    k = k + 2.0*deltak

    !third level
    do while (phi_x >= deltak) !update k until phi(1) >= deltak

        !do the first iteration
        phi_xminus1 = phi_0
        phi_x = phi_deltax
        phi_xplus1 = 2.0*phi_x - phi_xminus1 - (k**2.0)*(deltax
**2.0)*phi_x
        x = deltax

        !update phi
        call update_phi(phi_xminus1, phi_x, phi_xplus1, k, deltax,
number_of_iterations, x)

        !update k
        k = k + deltak

    end do

    write(*,*) "Third energy level: ", k

    write(*,*) deltax*number_of_iterations

contains

    subroutine update_phi(phi_xminus1, phi_x, phi_xplus1, k, deltax
, number_of_iterations, x)

        !deactivate implicit typing
        implicit none

        !define variables
        real, intent(inout) :: phi_xminus1, phi_x, phi_xplus1, k,
deltax, x
        integer, intent(in) :: number_of_iterations

        do i = 2, number_of_iterations
            phi_xminus1 = phi_x
            phi_x = phi_xplus1
            phi_xplus1 = 2.0*phi_x - phi_xminus1 - (k**2.0)*(deltax
**2.0)*phi_x
```

```
            x = x + deltax
            write(*,*) "x: ", x, "i", i
        end do

    end subroutine update_phi

end program shooting_method
```

O código foi compilado com o comando:

```
gfortran L2-5255417-ex-4.f90 -o L2-5255417-ex-4.exe
```

Resultados:

2.3 Exercício 5

Tarefa: Escreva um código para calcular os primeiros três níveis de energia para o poço de potencial infinito, usando o método da secante e as condições de contorno $\psi(0) = 0$ e $\psi'(0)6 \neq 0$. Compare o resultado com a solução exata e com o resultado do exercício 4.

Código Escrito:

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

O código foi compilado com o comando:

```
gfortran L2-5255417-ex-5.f90 -o L2-5255417-ex-5.exe
```

Resultados: