UNIVERSIDAD AUTÓNOMA DE QUERÉTARO
FACULTAD DE INGENIERÍA

Universidad Autónoma de Querétaro

Facultad de Ingeniería

Raíces múltiples y problemas de aplicación

Análisis numérico

Autor:
David Gómez Torres

Agosto 2021

ÍNDICE GENERAL

CAPÍTULO 1

INTRODUCCIÓN

Antes de presentar un serie de ejercicios resueltos y reportar el código con el que fueron resueltos, a continuación se hace un resumen de las páginas 127-134 del libro [**1**].

1.1. Raíces múltiples

Una **raíz múltiple** corresponde a un punto donde una función es tangencial al eje x. Por ejemplo, una raíz doble resulta de

$$f(x) = (x-3)(x-1)(x-1) \tag{1.1}$$

La ecuación tiene una raíz doble porque un valor de x hace que dos términos de la ecuación (1.1) sean iguales a cero. Gráficamente, esto significa que la curva toca en forma tangencial al eje x en la raíz doble.

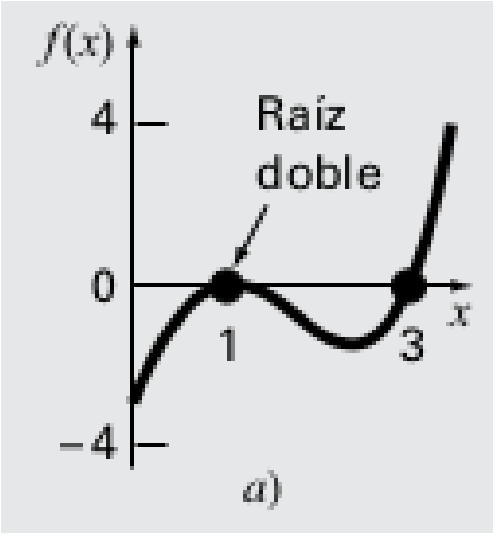


Figura 1.1: Representación gráfica de una raíz doble

Las raíces múltiples ofrecen algunas dificultades a muchos de los métodos numéricos expuestos en la parte dos:

1. El hecho de que la función no cambie de signo en raíces múltiples pares impide confiarse de los métodos cerrados.

2. Otro posible problema se relaciona con el hecho de que no sólo $f(x)$, sino también (x) se aproxima a cero en la raíz. Tales problemas afectan los métodos de Newton-Raphson y de la secante, los cuales contienen derivadas (o su aproximación) en el denominador de sus fórmulas respectivas. Esto provocará una división entre cero cuando la solución converge muy cerca de la raíz.

3. Es posible demostrar que el método de Newton-Raphson y el método de la secante convergen en forma lineal, en vez de cuadrática, cuando hay raíces múltiple.

   Una alternativa, también sugerida por Ralston y Rabinowitz (1978), consiste en definir una nueva función $u(x)$, que es el cociente de la función original entre su derivada:

$$u(x) = \frac{f(x)}{f'(x)} \tag{1.2}$$

   Por lo tanto, con la ecuación (1.2) permite desarrollar una forma alternativa del método de Newton-Raphson:

$$x_{i+1} = x_i \frac{u(x)}{u'(x} \tag{1.3}$$

   Podemos diferenciar la ecuación (1.3) para obtener

$$u'(x) = \frac{f'(x)f'(x) - f(x)f''(x)}{[f'(x)]^2} \tag{1.4}$$

   Se sustituyen las ecuaciones (1.2) y (1.4) en la ecuación (1.3) y se simplifica el resultado:

$$x_{i+1} = x_i - \frac{f(x_i)f'(x_i)}{[f'(x_i)]^2 - f(x_i)f''(x_i)} \tag{1.5}$$

1.2. Sistemas de ecuaciones lineales

Hasta aquí nos hemos ocupado de determinar las raíces de una sola ecuación no lineal. Un problema relacionado con éste consiste en obtener las raíces de un conjunto de ecuaciones simultáneas,

$$\begin{aligned} f_1(x_1, x_2, ..., x_n) &= 0 \\ f_2(x_1, x_2, ..., x_n) &= 0 \\ \vdots &= \vdots \\ f_n(x_1, x_2, ..., x_n) &= 0 \end{aligned}$$

La solución de este sistema consta de un conjunto de valores x que simultáneamente hacen que todas las ecuaciones sean iguales a cero.

A las ecuaciones algebraicas y trascendentes que no se pueden expresar de esta forma se les llama ecuaciones no lineales. Por ejemplo,

$$x^2 + xy = 10$$

y

$$y + 3xy^2 = 57$$

son dos ecuaciones simultáneas no lineales con dos incógnitas, x y y, las cuales se expresan en la forma de la ecuación como

$$u(x, y) = x^2 + xy–10 = 0$$

$$v(x, y) = y + 3xy–57 = 0$$

Así, la solución serían los valores de x y de y que hacen a las funciones u(x, y) y v(x, y) iguales a cero.

1.3. Newton-Raphson

Recuerde que el método de Newton-Raphson se utilizó empleando la derivada de una función, para calcular su intersección con el eje de la variable independiente

$$f(x_{++1}) = f(x_i) + f(x_{i+1} - x_i)f'(x)$$

donde x_i es el valor inicial de la raíz y x_{i+1} es el valor en el cual la recta tangente interseca el eje x. Esta se reordena para obtener la forma del método de Newton-Raphson para una sola ecuación.

La forma para múltiples ecuaciones se obtiene en forma idéntica. Sin embargo, se debe usar una serie de Taylor de múltiples variables para tomar en cuenta el hecho de que más de una variable independiente contribuye a la determinación de la raíz.

$$u(x_{++1}) = u_i + (x_{i+1} - x_i)\frac{\partial u_i}{\partial x} + (y_{i+1} - y_i)\frac{\partial u_i}{\partial y} \tag{1.6}$$

y

$$v(x_{++1}) = v_i + (x_{i+1} - x_i)\frac{\partial v_i}{\partial x} + (y_{i+1} - y_i)\frac{\partial v_i}{\partial y} \tag{1.7}$$

En consecuencia, se pueden usar manipulaciones algebraicas (por ejemplo, la regla de Cramer) para resolverlo:

$$x_{i+1} = x_i - \frac{u_i\dfrac{\partial v_i}{\partial y} - v_i\dfrac{\partial u_i}{\partial y}}{\dfrac{\partial u_i}{\partial x}\dfrac{\partial v_i}{\partial y} - \dfrac{\partial u_i}{\partial y}\dfrac{\partial u_i}{\partial x}} \tag{1.8}$$

y

$$y_{i+1} = y_i - \frac{v_i\dfrac{\partial v_i}{\partial y} - v_i\dfrac{\partial v_i}{\partial y}}{\dfrac{\partial u_i}{\partial x}\dfrac{\partial v_i}{\partial y} - \dfrac{\partial u_i}{\partial y}\dfrac{\partial u_i}{\partial x}} \tag{1.9}$$

El denominador de cada una de esas ecuaciones se conoce formalmente como el determinante **jacobiano** del sistema.

CAPÍTULO 2

METODOLOGÍA

2.1. Raíces de un sistema de ecuaciones

Problema 6.23

Encuentre las raíces de las ecuaciones simultáneas que siguen:

$$(x-4)^2 + (y-4)^2 = 5$$
$$x^2 + y^2 = 16$$

1. Use un enfoque gráfico para obtener los valores iniciales. Encuentre estimaciones refinadas con el método de Newton-Raphson para dos ecuaciones.

1. Enfoque gráfico

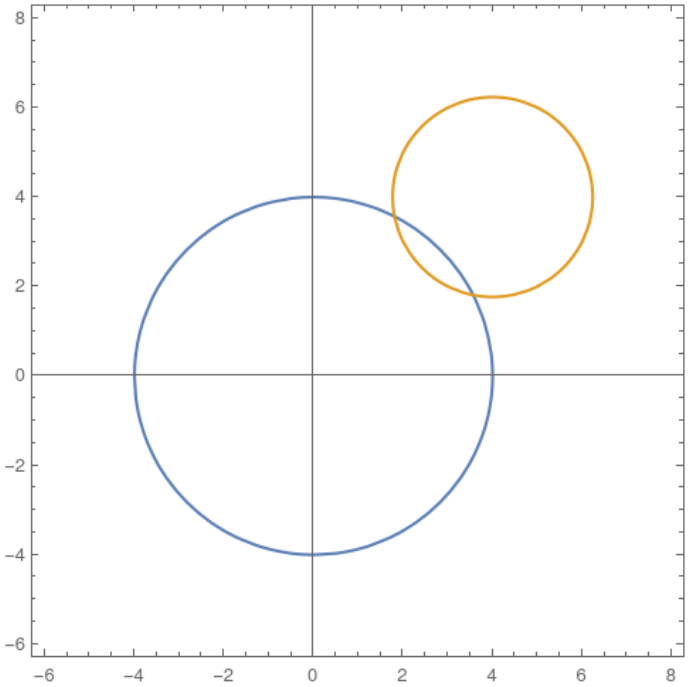


Figura 2.1: Gráfico de las dos funciones y su intersección

2. Mediante el método de Newton-Raphson para ecuaciones no lineales

```
import numpy as np

def F(x):

    f1=x[0]**2+x[1]**2-1+16
    f2=(x[0]**2-4)**2+(x[1]-4)**2-5
    return np.array([f1,f2])

def jacobiano(x):
    return np.array([2*x[0],2*x[1]],
                    2*x[0]-8,2*x[1]-8)

iter = 30
x=np.array([2,2])

for k in range(iter):
    xold = x
    jacobianoInv=np.linalg.inv(jacobiano(x))
    x = np.linalg.solve(jacobiano(x))
    e = np.linalg.norm(x - xold)

    print(iter,x,e)
```

2.2. Ecuación de caída de presión en un tubo

Problema 8.13

En una sección de tubo, la caída de presión se calcula así:

$$\Delta p = f\frac{L\rho V^2}{2D} \tag{2.1}$$

donde Δp = caída de presión (Pa), f = factor de fricción, L = longitud del tubo [m], ρ = densidad (kg/m^3), V = velocidad (m/s) y D = diámetro (m).

Para el flujo turbulento, la ecuación de Colebrook proporciona un medio para calcular el factor de fricción,

$$\frac{1}{\sqrt{f}} = -2\log\left(\frac{\epsilon}{3{,}7D} + \frac{2{,}51}{Re\sqrt{f}}\right) \tag{2.2}$$

donde ϵ = rugosidad (m) y Re = número de Reynolds,

$$\mathrm{Re} = \frac{\rho V D}{\mu}$$

donde μ = viscosidad dinámica (N · s/m^2)

1. Determine Δp para un tramo horizontal de tubo liso de 0.2 m de longitud, dadas ρ = 1.23 kg/m^3 , m = 1.79×10^{-5} N· s/m^2, D = 0.005 m, V = 40 m/s y ϵ = 0.0015 mm.
   Utilice un método numérico para determinar el factor de fricción. Obsérvese que los tubos lisos tienen Re < 10^5 , un valor inicial apropiado se obtiene con el uso de la fórmula de Blasius, $f = 0.316/Re^{0,25}$.

2. Repita el cálculo pero para un tubo de acero comercial más rugoso (ϵ = 0.045 mm).

1.
```
import math

L = 0.2 #longitud
r = 1.23 #densidad
m = 1.79e-5 #viscosidad
D = 0.005 #diametro
v = 40 #velocidad
e = 0.0015 #rugosidad


Re = (r*v*D)/m #numero Reynolds
fp = 0.316/Re**0.25 #formula de Blasius

print("Numero de Reynolds: ",Re, "\n")

def f(x):
    return -2*math.log(e/3.7*D + 2.51/Re*math.sqrt(x))-1/
    math.sqrt(x)

def metodo_secante(f,x0,x1,es,imax):

    iter = 0
    ea = 100


    while(ea > es and iter <= imax):
        fp = (f(x1) - f(x0))/(x1 - x0)

        x = x1 - f(x1)/fp
        if(x != 0):
            ea = abs((x - x1)/x)

        x0 = x1
        x1 = x

        iter += 1

        print(iter, x, ea)

        Dp = fp*L*r*v**2/2*D #diferencia de presion

metodo_secante(f,0.1,0.3,0.01,100)
```

2.

```
import math

L = 0.2 #longitud
r = 1.23 #densidad
m = 1.79e-5 #viscosidad
D = 0.005 #diametro
v = 40 #velocidad
e = 0.045 #rugosidad


Re = (r*v*D)/m #numero Reynolds
fp = 0.316/Re**0.25 #formula de Blasius

print("Numero de Reynolds: ",Re, "\n")

def f(x):
    return -2*math.log(e/3.7*D + 2.51/Re*math.sqrt(x))-1/
    math.sqrt(x)

def metodo_secante(f,x0,x1,es,imax):

    iter = 0
    ea = 100


    while(ea > es and iter <= imax):
        fp = (f(x1) - f(x0))/(x1 - x0)

        x = x1 - f(x1)/fp
        if(x != 0):
            ea = abs((x - x1)/x)

        x0 = x1
        x1 = x

        iter += 1

        print(iter, x, ea)

        Dp = fp*L*r*v**2/2*D #diferencia de presion

metodo_secante(f,0.1,0.3,0.01,100)
```

2.3. Descarga en un drenaje de agua

Problema 8.15

En la ingeniería ambiental (una especialidad de la ingeniería civil), la ecuación siguiente se emplea para calcular el nivel de oxígeno c=(mg/L) en un río aguas abajo de la descarga de un drenaje:

$$c = 10 - 20(e^{-0,2x} - e^{-0,75x}) \tag{2.3}$$

donde x es la distancia aguas abajo en kilómetros.

1. Determine la distancia aguas abajo de la corriente, a la cual el nivel de oxígeno cae hasta una lectura de 5 mg/L. (Sugerencia: Está dentro de 2 km de la descarga.) Encuentre la respuesta con un error de 1 %.
   Obsérvese que los niveles de oxígeno por debajo de 5 mg/L por lo general son dañinos para ciertas especies de pesca deportiva, como la trucha y el salmón.

2. Calcule la distancia aguas abajo a la cual el oxígeno se encuentra al mínimo. ¿Cuál es la concentración en dicha ubicación?

1.

```python
import math

def f(x):
    return 5-20*math.exp(-0.2*x)+20*math.exp(-0.75*x)


def df(x):
    return 4*math.exp(-0.2*x)-15*math.exp(-0.75*x)

def newton_raphson(f,df,xi,es):
    x = xi
    ea = 100
    iter = 1

    while(ea > es):
        x = x - f(x)/df(x)
        ea = abs(f(x))
        iter += 1

        print(iter, x, ea)

newton_raphson(f,df,1,0.01)
```

2.

```python
import math

def f(x):
    return -20*math.exp(-0.2*x)+20*math.exp(-0.75*x)

def df(x):
    return 4*math.exp(-0.2*x)-15*math.exp(-0.75*x)

def newton_raphson(f,df,xi,es):
    x = xi
    ea = 100
    iter = 0

    while(ea > es):
        x = x - f(x)/df(x)
        ea = abs(f(x))
        iter += 1

        print(iter, x,ea)

newton_raphson(f,df,10,0.01)
```

2.4. Oscilación amortiguada

Problema 8.19

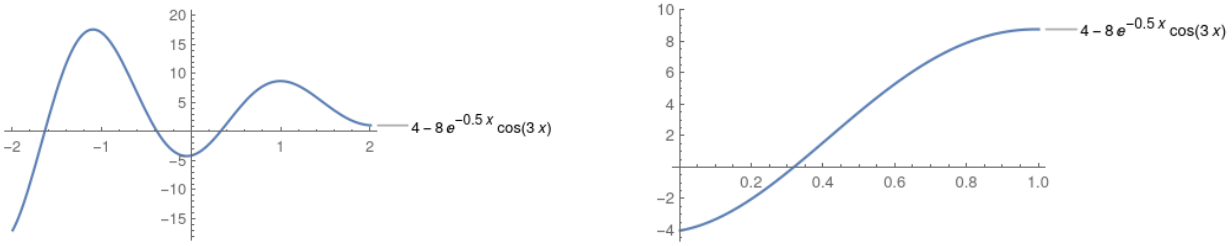
El desplazamiento de una estructura está definido por la ecuación siguiente para una oscilación amortiguada:

$$y = 8e^{-k}\cos(\omega t) \tag{2.4}$$

donde $k = 0.5$ y $\omega = 3$.

1. Utilice el método gráfico para realizar una estimación inicial del tiempo que se requiere para que el desplazamiento disminuya a 4.
2. Emplee el método de Newton-Raphson para determinar la raíz con $\epsilon_s = 0.01\,\%$.
3. Use el método de la secante para determinar la raíz con $\epsilon_s = 0.01\,\%$.

1. Graficamente



(a) Panorama de la función

(b) Ampliación de la rais

Figura 2.2: Raíces por el método gráfico

2.
```python
import math

def f(x):
    return 4-8*math.exp(-0.5*x)*math.cos(3*x)

def df(x):
    return 4*math.exp(-0.5*x)*
    math.cos(3*x)+24*math.exp(-0.5*x)*math.sin(3*x)

def newton_raphson(f,df,xi,es):
    x = xi
    ea = 100
    iter = 0

    while(ea > es):
        x = x - f(x)/df(x)
        ea = abs(f(x))
        iter += 1
        print(iter, x, ea)

newton_raphson(f,df,0.1,0.01)
```

3.
```python
import math

def f(x):
    return 4-8*math.exp(-0.5*x)*math.cos(3*x)

def metodo_secante(f,x0,x1,es,imax):

    iter = 0
    ea = 100

    while(ea > es and iter <= imax):
        fp = (f(x1) - f(x0))/(x1 - x0)

        x = x1 - f(x1)/fp
        if(x != 0):
            ea = abs((x - x1)/x)

        x0 = x1
        x1 = x

        iter += 1

        print(iter, x, ea)

metodo_secante(f,0,1,0.01,5)
```

2.5. Trayectoria de un proyectil

Problema 8.37

En ciertas ocasiones, los ingenieros aeroespaciales deben calcular las trayectorias de proyectiles, como cohetes. Un problema parecido tiene que ver con la trayectoria de una pelota que se lanza. Dicha trayectoria está definida por las coordenadas (x, y), como se ilustra en la figura (). La trayectoria se modela con la ecuación

$$y = (\tan\theta_0)x - \frac{g}{2v_0^2\cos^2\theta_0}x^2 + y_0$$

Calcule el ángulo inicial θ_0 , apropiado si la velocidad inicial $v_0 = 20$ m/s y la distancia x al catcher es de 40 m.

Obsérvese que la pelota sale de la mano del lanzador con una elevación $y_0 = 1.8$ m, y el catcher la recibe a 1 m. Exprese el resultado final en grados. Para g, utilice un valor de 9.81 m/s^2 , y emplee el método gráfico para elegir valores iniciales.

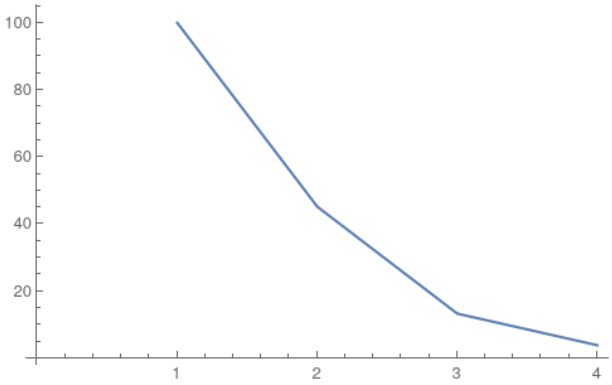
```
import math
y = 1 #altura del catcher
xc = 40 #distancia al catcher
g = 9.81 #aceleracion gravedad
v = 20 #velocidad
yi = 1.7 #altura inicial

def f(x):
    return math.tan(x)*xc-g/(2*v**2*math.cos(x))*xc**2+yi

def metodo_secante(f,x0,x1,es,imax):

    iter = 0
    ea = 100

    while(ea > es and iter <= imax):
        fp = (f(x1) - f(x0))/(x1 - x0)

        x = x1 - f(x1)/fp
        if(x != 0):
            ea = abs((x - x1)/x)

        x0 = x1
        x1 = x

        iter += 1

        print(iter, x, ea)
metodo_secante(f,0.1,5,0.01,5)
```

CAPÍTULO 3

RESULTADOS

Ejercicio 6.23

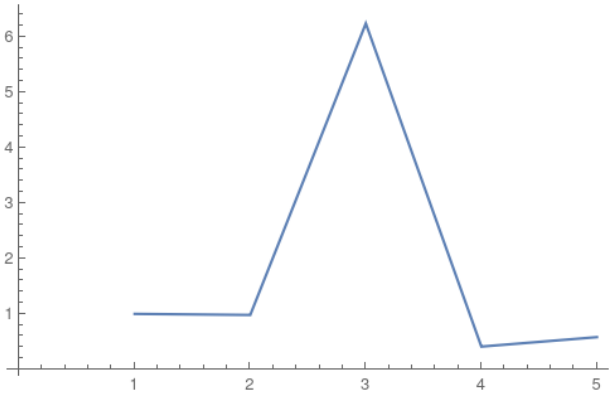
Iteración	Error relativo porcentual
1	1
2	0.687771
3	0.0357701
4	0.001135



Ejercicio 8.13

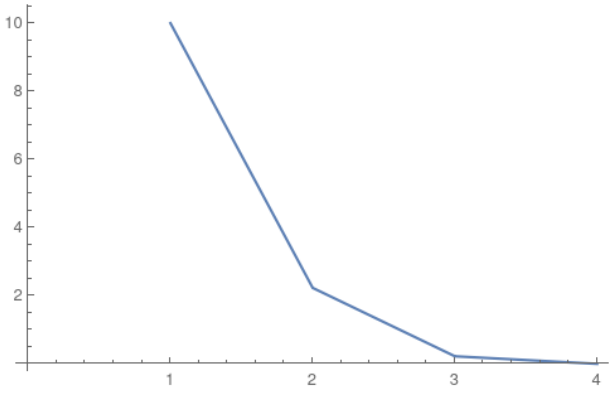
1. $\epsilon = 0.0015$

Iteración	Error relativo porcentual
1	1
2	0.67851
3	0.09467
4	0.07563



2. $\epsilon = 0.045$

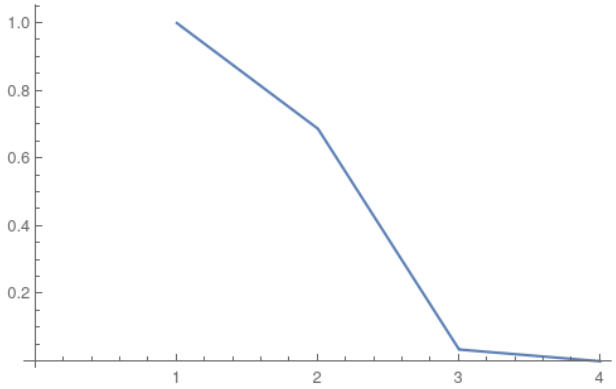
Iteración	Error relativo porcentual
1	0.5
2	0.25
3	0.125
4	0.0625
5	0.0312



Ejercicio 8.15

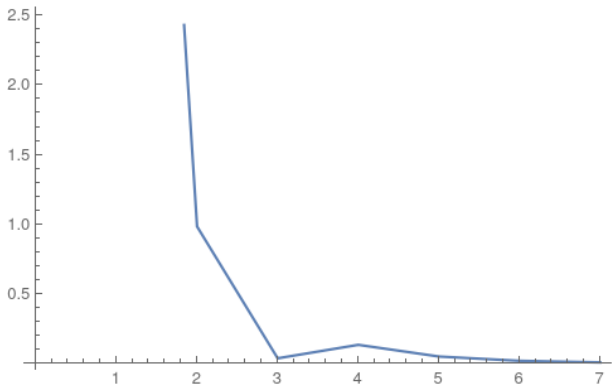
1. $c = 15$

Iteración	Error relativo porcentual
1	1
2	0.687771
3	0.0357701
4	0.001135



2. $c = 0$

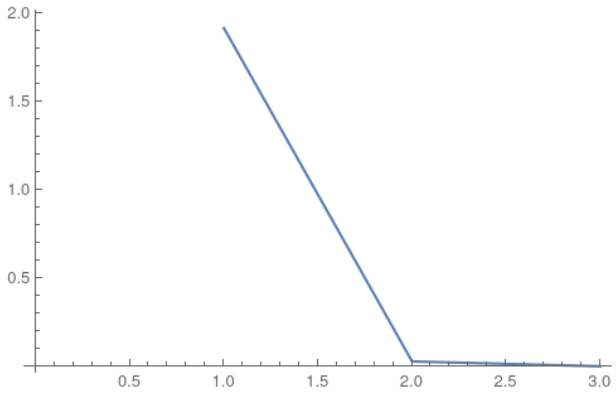
Iteración	Error relativo porcentual
1	10
2	0.984191
3	0.0361897
4	0.133130
5	0.048975
6	0.0180172
7	0.006628



Ejercicio 8.19

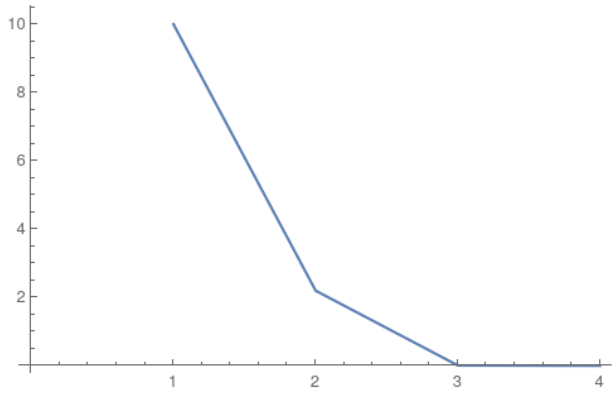
1. Newton-Raphson

Iteración	Error relativo porcentual
1	10
2	1.918827
3	0.02785798
4	0.0002924



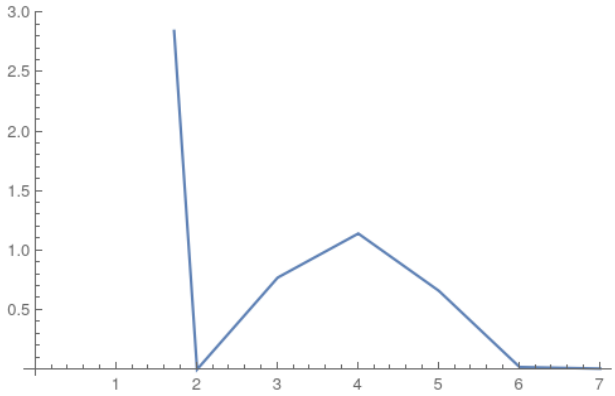
2. Secante

Iteración	Error relativo porcentual
1	10
2	2.200921
3	0.0125787
4	0.003881



Ejercicio 8.37

Iteración	Error relativo porcentual
1	100
2	0.0000357
3	0.7721
4	1.1428
5	0.6606
6	0.01997



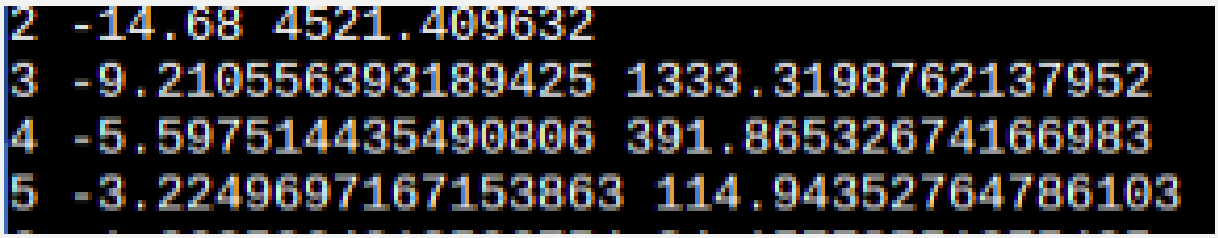
CAPÍTULO 4

ANEXOS

4.1. Anexo A: Evidencia del funcionamiento de los código reportados

Ejercicio 6.23

Figura 4.1: Evidencia del funcionamiento



Ejercicio 8.13

1. $\epsilon = 0.0015$

Figura 4.2: Evidencia del funcionamiento

```
Numero de Reynolds:  13743.016759776536

1 31.004233851344708 0.9677463405548198
2 188.19446382511342 0.835254272515919 9
3 1292.6080224838804 0.854407167098129
4 7178.058149486326 0.8199223250125967
5 36033.936883845956 0.800797282499979 7
6 156690.68767827871 0.770031407623714 5
7 588600.7078204704 0.733791200730137 3
8 1872080.574141743 0.6855900777185753
9 4949285.814477997 0.621747329955081 3
10 10650881.508529402 0.5353167894587385
11 18348769.11179671 0.4195315531175448 6
12 25294338.70548501 0.2745898864785174 5
13 28966650.112385936 0.126777221137168 1
14 29892130.205208004 0.030960660430310323
15 29972000.59189727 0.0026648333481903265

La diferencia de presion en el tubo es 8832.56
```

2. $\epsilon = 0.045$

Figura 4.3: Evidencia del funcionamiento

```
Numero de Reynolds:  13743.016759776536

1 96.34809043596353 0.98962096710505 5
2 472.68623146513875 0.7961690355622949
3 3230.9736564594955 0.8537016139020122
4 16627.0546923695 0.8056797360543804
5 78444.19167540141 0.7880396962827851
6 316200.6358328525 0.751916401215372 7
7 1093719.8399775417 0.710894303755754
8 3169447.727659373 0.654917848799718
9 7553932.439236423 0.5804241362820884
10 14512200.48435393 0.4794771166936013
11 22241354.362301614 0.3475127347032586
12 27640424.384570003 0.1953323851743161 7
13 29650572.62994629 0.0677945842889419 4
14 29957252.372238956 0.010237245341527434
15 29969781.619449165 0.0004180626795784643

La diferencia de presion en el tubo es 8832.56
```

Ejercicio 8.15

Figura 4.4: Evidencia del funcionamiento

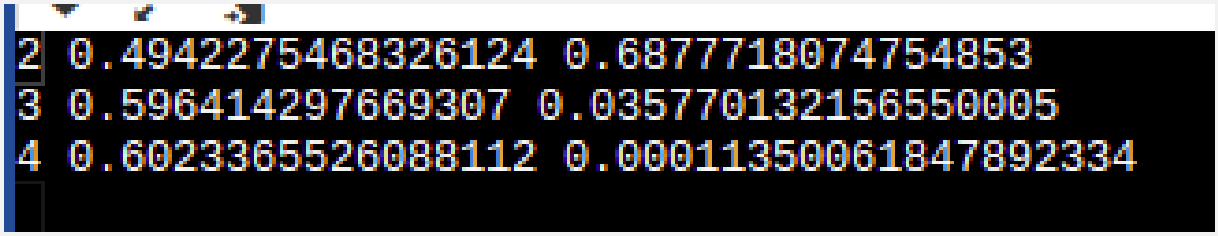
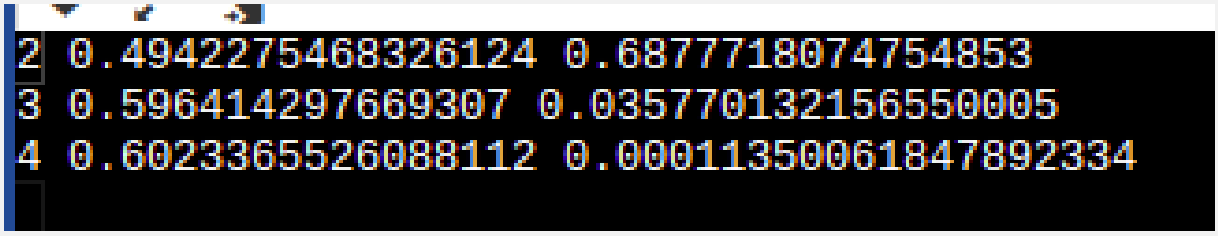


Figura 4.5: Evidencia del funcionamiento



Ejercicio 8.19

Figura 4.6: Evidencia del funcionamiento

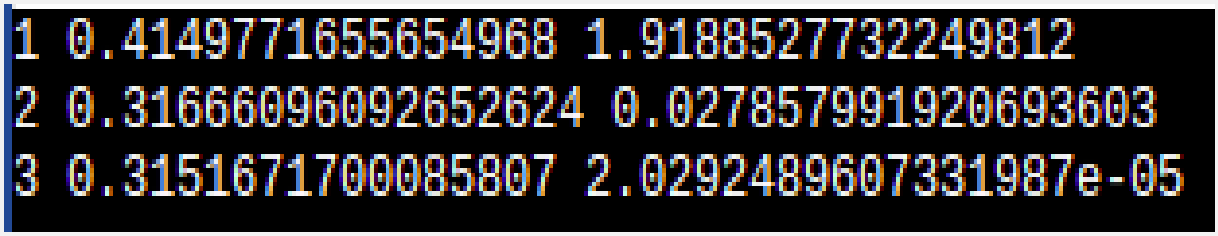


Figura 4.7: Evidencia del funcionamiento

```
1 0.3124100255077654 2.2009216041472506
2 0.3163898126690613 0.0125787462235981 4
3 0.3151644059458260 6 0.0038881507559767 72
```

Ejercicio 8.37

Figura 4.8: Evidencia del funcionamiento

```
2 -1.1575438530152034 0.7721837903674413
3 -0.022329507275313354 50.83920266323742
4 0.15635291832425752 1.1428147777133562
5 0.4606807587464919 5 0.6606046261847525
6 0.4699798834637507 5 0.0197862186115802 88
```

CAPÍTULO 5

BIBLIOGRAFÍA

1. Chapra, S. C., Canale, R. P. (2011). *Numerical methods for engineers* (Vol. 1221). New York: Mcgraw-hill.