



UNIVERSIDAD AUTÓNOMA DE QUERÉTARO
FACULTAD DE INGENIERÍA

Universidad Autónoma de Querétaro

FACULTAD DE INGENIERÍA

MÉTODOS ABIERTOS

Análisis numérico

Autor:
David Gómez Torres

Agosto 2021

1. Introducción	1
1.1. Iteración simple de punto fijo	1
1.2. Método de Newton-Raphson	2
1.2.1. Desventajas del método de Newton-Raphson	2
1.3. Método de la secante	3
2. Metodología	4
2.1. Raíz de función senoidal	4
2.2. Raíz de polinomio de 2do grado	5
2.3. Raíz de función trigonométrica	7
2.4. Raíz de función polinómica	10
2.5. Raíz de función exponencial	12
2.6. Llenado de un tanque	13
3. Resultados	14
4. Anexos	20
4.1. Anexo A: Evidencia del funcionamiento de los código reportados	20
5. Bibliografía	25

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 113-124 y 127-130 del libro [1].

1.1. Iteración simple de punto fijo

Esta fórmula puede desarrollarse como una iteración simple de punto fijo (también llamada iteración de un punto o sustitución sucesiva o método de punto fijo), al arreglar la ecuación $f(x) = 0$ de tal modo que x esté del lado izquierdo de la ecuación:

$$x = g(x) \quad (1.1)$$

La ecuación (1.1) se utiliza para obtener una nueva aproximación x_{i+1} , expresada por la fórmula iterativa

$$x_{i+1} = g(x_i) \quad (1.2)$$

El error aproximado de esta ecuación se calcula usando el error normalizado estimador

$$E = \frac{x_{i+1} - x_i}{x_{i+1}} 100 \% \quad (1.3)$$

El método de las dos curvas también se utiliza para ilustrar la convergencia y divergencia de la iteración de punto fijo. En primer lugar, la ecuación (1.1) se reexpresa como un par de ecuaciones $y_1 = x$ y $y_2 = g(x)$. Vea figura (1.1).

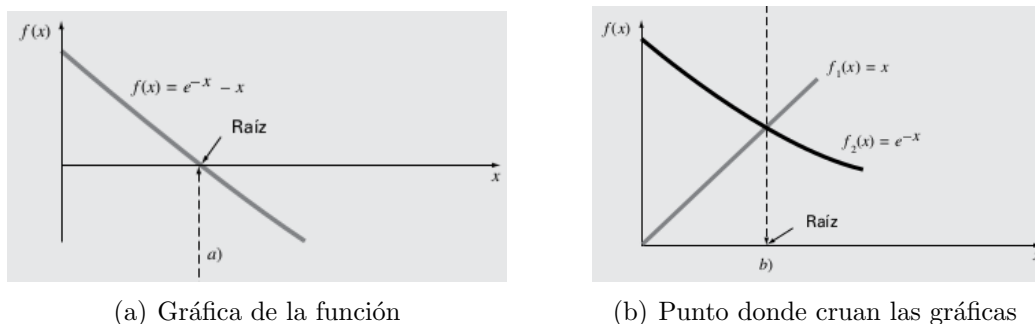


Figura 1.1: Convergencia y divergencia del método de punto fijo

Al analizar la figura anterior, se debe notar que la iteración de punto fijo converge si, en la región de interés, $g(x) < 1$. En otras palabras, la convergencia ocurre si la magnitud de la pendiente de $g(x)$ es menor que la pendiente de la recta $f(x) = x$.

1.2. Método de Newton-Raphson

El método de Newton-Raphson se deduce a partir la interpretación geométrica de que si el valor inicial para la raíz es x_i , entonces se puede trazar una tangente desde el punto $[x_i, f(x_i)]$. Por lo común, el punto donde esta tangente cruza al eje x representa una aproximación mejorada de la raíz.

De la figura (1.2), se tiene que la primera derivada en x es equivalente a la pendiente:

$$f'(x_i) = \frac{f(x_i) - 0}{x_i - x_{i+1}}$$

que se arregla para obtener

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \quad (1.4)$$

la cual se conoce como fórmula de Newton-Raphson.

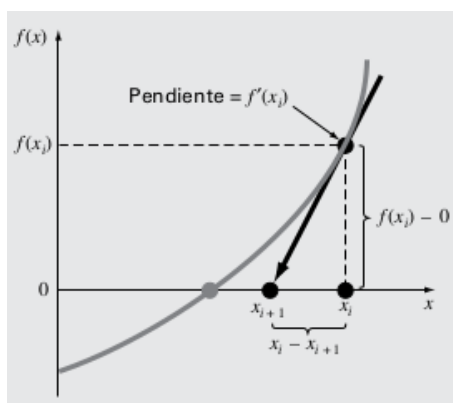


Figura 1.2: Método Newton-Raphson

1.2.1. Desventajas del método de Newton-Raphson

De manera que no hay un criterio general de convergencia para el método de Newton-Raphson. Su convergencia depende de la naturaleza de la función y de la exactitud del valor inicial. La única solución en estos casos es tener un valor inicial que sea “suficientemente” cercano a la raíz.

Los buenos valores iniciales por lo común se predicen con un conocimiento del problema físico o mediante el uso de recursos alternativos, como las gráficas, que proporcionan mayor claridad en el comportamiento de la solución.

1.3. Método de la secante

Un problema potencial en la implementación del método de Newton-Raphson es la evaluación de la derivada. Aunque esto no es un inconveniente para los polinomios ni para muchas otras funciones, existen algunas funciones cuyas derivadas en ocasiones resultan muy difíciles de calcular. En dichos casos, la derivada se puede aproximar mediante una diferencia finita dividida hacia atrás,

$$f'(x_i) = \frac{f(x_{i-1}) - f(x_i)}{x_{i-1} - x_i}$$

Esta aproximación se sustituye en la ecuación (1.3) para obtener la siguiente ecuación iterativa:

$$x_{i+1} = x_i - \frac{f(x_i)(x_{i+1} - x_i)}{f(x_{i-1}) - f(x_i)} \quad (1.5)$$

La ecuación (1.5) es la fórmula para el método de la secante. Observe que el método requiere de dos valores iniciales de x . Sin embargo, debido a que no se necesita que $f(x)$ cambie de signo entre los valores dados, este método no se clasifica como un método cerrado.

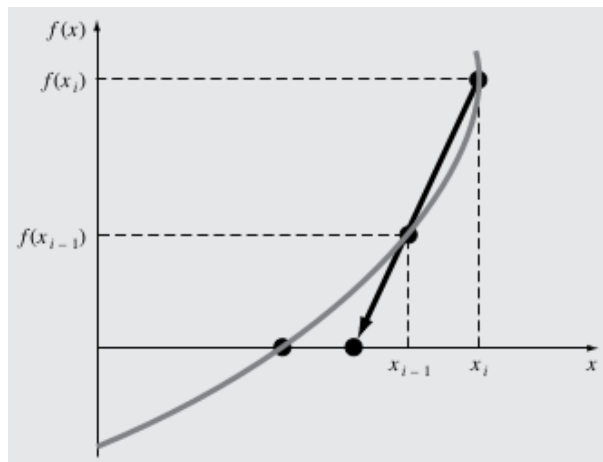


Figura 1.3: Representación gráfica del método de la secante

Este es similar a la del método de Newton-Raphson (1.3) en el sentido de que una aproximación de la raíz se predice extrapolando una tangente de la función hasta el eje x . Sin embargo, el método de la secante usa diferencias divididas en lugar de una derivada para estimar la pendiente.

2.1. Raíces de función senoidal

Problema 6.1

Utilice la iteración simple de punto fijo para localizar la raíz de $f(x) = \sin(\sqrt{x}) - x$:

1. Haga una elección inicial de $x_0 = 0.5$ e itere hasta que $e_a < 0.01\%$.

1. Mediante el método del punto fijo

```
1 import numpy as np
2
3 def f(x):
4     return np.sin(np.sqrt(x))-x
5
6 def g(x):
7     return np.sin(np.sqrt(x))
8
9 def punto_fijo(f,g,xr,es,imax):
10     ea = 100
11     iter = 0
12
13     while(ea > es and i <= imax):
14         if(xr != 0):
15             ea = np.abs(g(xi)-xr/xr)*100
16             xr = g(xi)
17             iter += 1
18
19         print(i, xr, ea)
20
21 punto_fijo(f,g,0.5,0.01,30)
```

2.2. Raíz de polinomio de 2do grado

Problema 6.3

Utilice diferentes métodos para determinar una raíz de $f(x) = -0,9x^2 + 1,7x + 2,5$ con el uso de $x_0 = 5$. Haga el cálculo hasta que e_a sea menor que $e_s = 0.01\%$. Asimismo, realice una comprobación del error de su respuesta final:

1. Iteración de punto fijo.
2. Newton-Raphson

```
1.
1 import numpy as np
2
3 def f(x):
4     return -0.9*x**2+1.7*x+2.5
5
6 def g(x):
7     return -0.529411*x**2-1.470588
8
9 def punto_fijo(f,g,xi,es,imax):
10     ea = 100
11     i = 0
12
13     while(ea > es and i <= imax):
14         if(i > 0):
15             ea = np.abs(g(xi)-xi)
16             xi = g(xi)
17             i += 1
18
19     print(i, xi, ea)
20
21 punto_fijo(f,g,5,0.01,30)
```

```
2.
1  import numpy as np
2
3  def f(x):
4      return -0.9*x**2+1.7*x+2.5
5
6  def df(x):
7      return -1.8*x+1.7
8
9  def newton_raphson(f,df,xr,es):
10     x = xr
11     ea = 100
12     iter = 1
13
14     while(ea > es):
15         x = x - f(x)/df(x)
16         ea = abs(f(x))
17         iter += 1
18
19         print(iter, x, ea)
20
21 newton_raphson(f,df,5,0.01)
```


2.3. Raíz de función trigonométrica

Problema 6.7

Localice la primera raíz positiva de $f(x) = \sin(x) + \cos(1+x^2) - 1$ donde x está en radianes. Para localizar la raíz, use cuatro iteraciones del método de la secante con valores iniciales de:

1. $x_{i-1} = 1.0$ y $x_i = 3.0$
2. $x_{i-1} = 1.5$ y $x_i = 2.5$
3. $x_{i-1} = 1.5$ y $x_i = 2.25$
4. Use el método gráfico para explicar su resultado

```

1.
1 import numpy as np
2
3 def f(x):
4     return np.sin(x) + np.cos(1+x**2) - 1
5
6 def metodo_secante(f, x0, x1, es, imax):
7
8     iter = 0
9     ea = 100
10
11     while(ea > es and iter <= imax):
12         fp = (f(x1) - f(x0))/(x1 - x0)
13
14         x = x1 - f(x1)/fp
15         if(x != 0):
16             ea = abs((x - x1)/x)
17
18         x0 = x1
19         x1 = x
20
21         iter += 1
22
23         print(iter, x, ea)
24
25 metodo_secante(f, 1, 3, 0.01, 4)

```

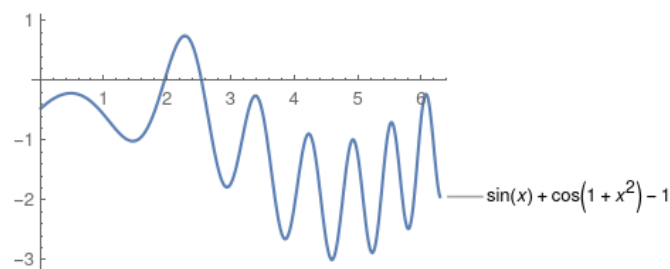
```
2.
1 import numpy as np
2
3 def f(x):
4     return np.sin(x) + np.cos(1+x**2) - 1
5
6 def metodo_secante(f,x0,x1,es,imax):
7
8     iter = 0
9     ea = 100
10
11     while(ea > es and iter <= imax):
12         fp = (f(x1) - f(x0))/(x1 - x0)
13
14         x = x1 - f(x1)/fp
15         if(x != 0):
16             ea = abs((x - x1)/x)
17
18         x0 = x1
19         x1 = x
20
21         iter += 1
22
23         print(iter, x, ea)
24
25 metodo_secante(f,1.5,2.5,0.01,4)
```

```

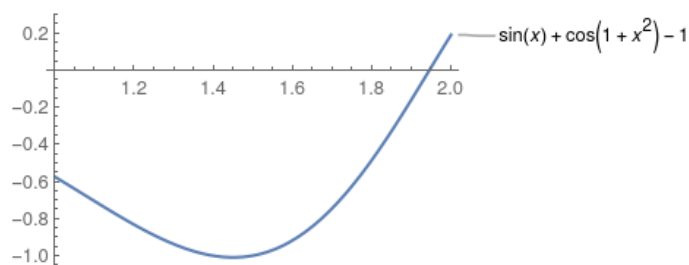
3.
1 import numpy as np
2
3 def f(x):
4     return np.sin(x) + np.cos(1+x**2) - 1
5
6 def metodo_secante(f, x0, x1, es, imax):
7
8     iter = 0
9     ea = 100
10
11     while(ea > es and iter <= imax):
12         fp = (f(x1) - f(x0))/(x1 - x0)
13
14         x = x1 - f(x1)/fp
15         if(x != 0):
16             ea = abs((x - x1)/x)
17
18         x0 = x1
19         x1 = x
20
21         iter += 1
22
23         print(iter, x, ea)
24
25 metodo_secante(f, 1.5, 2.25, 4)

```

4. Gráficamente



(a) Panorama de la función



(b) Ampliación de la raíz

Figura 2.1: Raíces por el método gráfico

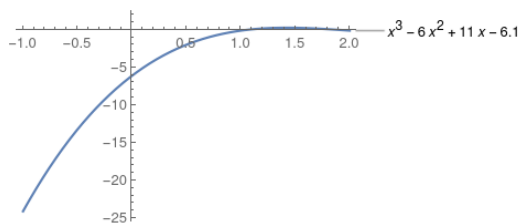
2.4. Raíz de función polinómica

Problema 6.9

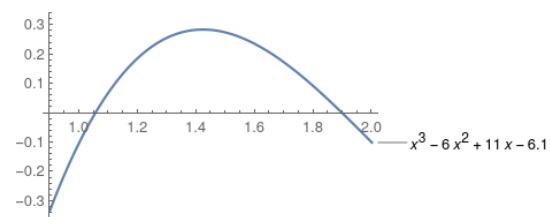
Determine la raíz real más grande de $f(x) = x^3 - 6x^2 + 11x - 6.1$:

1. En forma gráfica.
2. Con el uso del método de Newton-Raphson (tres iteraciones, $x_i = 3.5$).
3. Con el método de la secante (tres iteraciones, $x_{i+1} = 2.5$ y $x_i = 3.5$).

1. Gráficamente



(a) Panorama de la función



(b) Ampliación de la raíz

Figura 2.2: Raíces por el método gráfico

2.

```

1 import numpy as np
2
3 def f(x):
4     return x**3-6*x**2+4*x-6.1
5
6 def df(x):
7     return 3*x**2-12*x+4
8
9 def newton_raphson(f, df, xr, es):
10     x = xr
11     ea = 100
12     iter = 1
13
14     while(ea > es):
15         x = x - f(x)/df(x)
16         ea = abs(f(x))
17         iter += 1
18
19     print(iter, x, ea)
20
21 newton_raphson(f, df, 3.5, 0.01)

```

```
3.
1 import numpy as np
2
3 def f(x):
4     return x**3-6*x**2+4*x-6.1
5
6 def metodo_secante(f,x0,x1,es,imax):
7
8     iter = 0
9     ea = 100
10
11     while(ea > es and iter <= imax):
12         fp = (f(x1) - f(x0))/(x1 - x0)
13
14         x = x1 - f(x1)/fp
15         if(x != 0):
16             ea = abs((x - x1)/x)
17
18         x0 = x1
19         x1 = x
20
21         iter += 1
22
23         print(iter, x, ea)
24
25 metodo_secante(f,2.5,3.5,0.01,5)
```

2.5. Raíz de función exponencial

Problema 6.13

Debe determinar la raíz de la siguiente función fácilmente diferenciable

$$e^{0.5x} = 5 - 5x$$

Elija la mejor técnica numérica, justifique su elección y luego use esa técnica para determinar la raíz. Observe que se sabe que, para valores iniciales positivos, todas las técnicas, salvo la iteración de punto fijo, finalmente convergen.

Realice iteraciones hasta que el error relativo aproximado caiga por debajo de 2%. Si usted emplea un método cerrado, use valores iniciales de $x_l = 0$ y $x_u = 2$.

Si escoge el método de Newton-Raphson o el modificado de secante, use un valor inicial de $x_i = 0.7$. Si se decide por el método de secante, use valores iniciales de $x_{i-1} = 0$ y $x_i = 2$.

```

1 import numpy as np
2 import math
3
4 def f(x):
5     return math.exp(0.5*x)+5*x-5
6
7 def df(x):
8     return 0.5*math.exp(0.5*x)+5
9
10 def newton_raphson(f, df, xr, es):
11     x = xr
12     ea = 100
13     iter = 1
14
15     while(ea > es):
16         x = x - f(x)/df(x)
17         ea = abs(f(x))
18         iter += 1
19
20     print(iter, x, ea)
21
22
23 newton_raphson(f, df, 0.7, 0.2)

```

2.6. Llenado de un tanque

Problema 6.19

Suponga el lector que está diseñando un tanque esférico de almacenamiento de agua para un poblado pequeño de un país en desarrollo. El volumen del líquido que puede contener se calcula con

$$V = \pi h^2 \frac{[3R - h]}{3}$$

donde V = volumen [m^3], h = profundidad del agua en el tanque [m] y R = radio del tanque [m].

Si $R = 3$ m, ¿a qué profundidad debe llenarse el tanque de modo que contenga $30 m^3$? Haga tres iteraciones del método de Newton-Raphson para determinar la respuesta.

Encuentre el error relativo aproximado después de cada iteración. Observe que el valor inicial de R convergerá siempre.

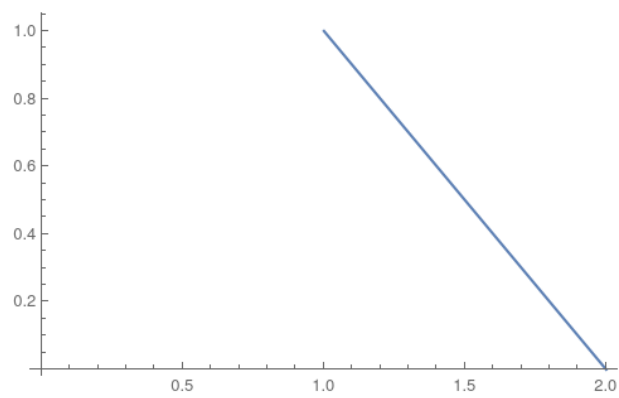
```

1 import numpy as np
2 import math
3
4 def f(x):
5     return (3-x/3)*math.pi*x**2-30
6
7 def df(x):
8     return 6*math.pi*x-math.pi*x**2
9
10 def newton_raphson(f,df,xi,tol):
11     x = xi
12     error = 100
13     n = 1
14
15     while(error > tol):
16         x = x - f(x)/df(x)
17         error = abs(f(x))
18         n += 1
19
20     print(n, x, error)
21
22 newton_raphson(f,df,1,0.01)

```

Ejercicio 6.1

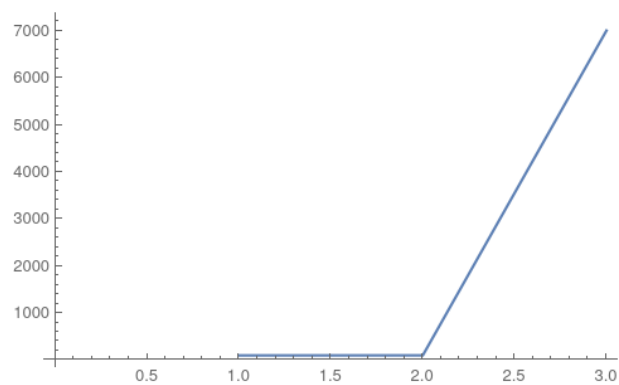
Iteración	Error relativo porcentual
1	1
2	0.0718886
3	0.0293773
4	0.01119568
5	0.00415129



Ejercicio 6.3

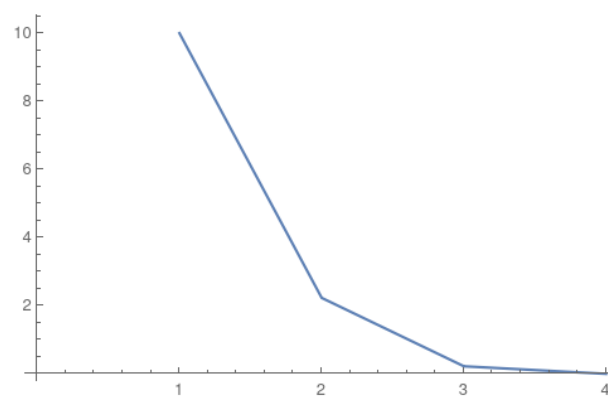
1. Punto fijo

Iteración	Error relativo porcentual
1	100
2	101.2564
3	7004.64



2. Newton-Raphson

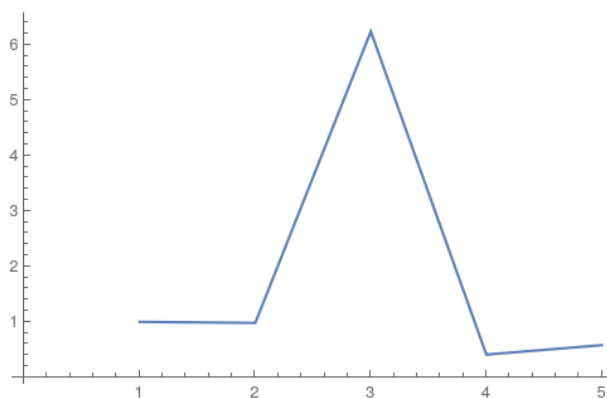
Iteración	Error relativo porcentual
1	0.5
2	0.25
3	0.125
4	0.0625
5	0.0312



Ejercicio 6.7

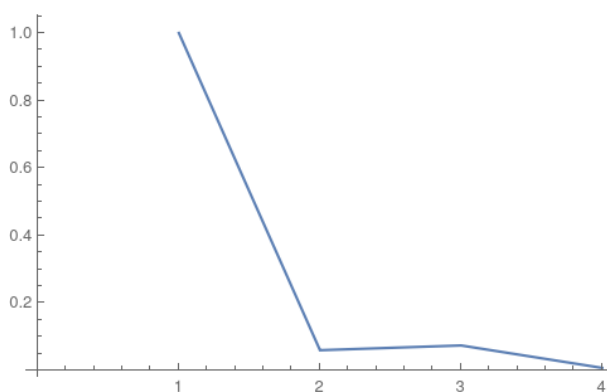
1. $x_{i-1} = 1.0$ y $x_i = 3.0$

Iteración	Error relativo porcentual
1	1
2	0.981070
3	6.241893
4	0.40975
5	0.580428



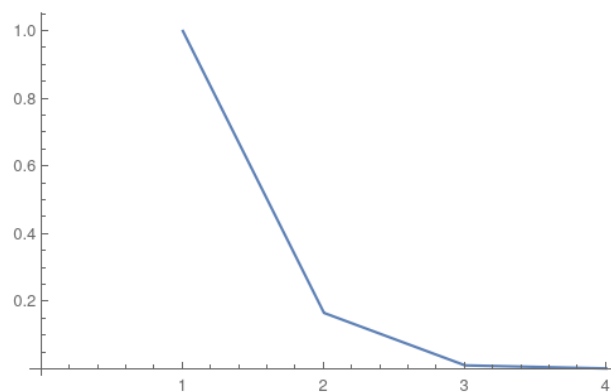
2. $x_{i-1} = 1.5$ y $x_i = 2.50$.

Iteración	Error relativo porcentual
1	1
2	0.0607024
3	0.0747298
4	0.008291



3. $x_{i-1} = 1.5$ y $x_i = 2.25$.

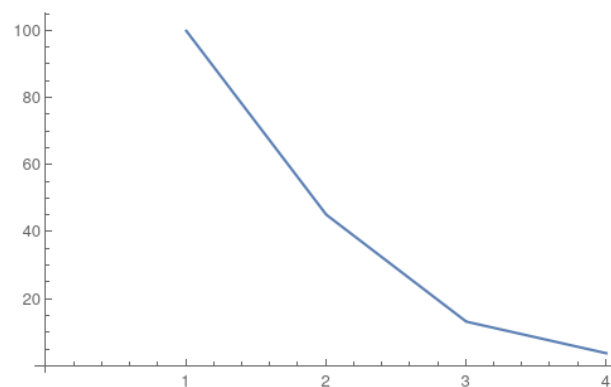
Iteración	Error relativo porcentual
1	1
2	0.167607
3	0.0125347
4	0.003535



Ejercicio 6.9

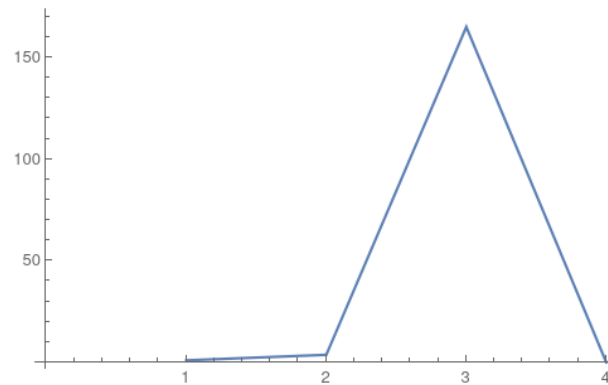
1. Newton-Raphson

Iteración	Error relativo porcentual
1	45.2140
2	13.3331
3	3.9186

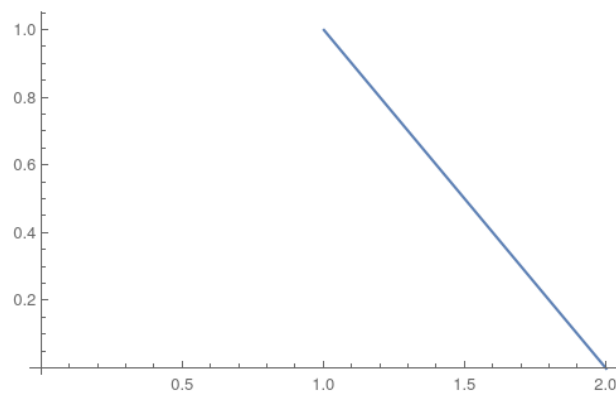


2. Secante

Iteración	Error relativo porcentual
1	1
2	3.725409
3	165.070857
4	0.000422

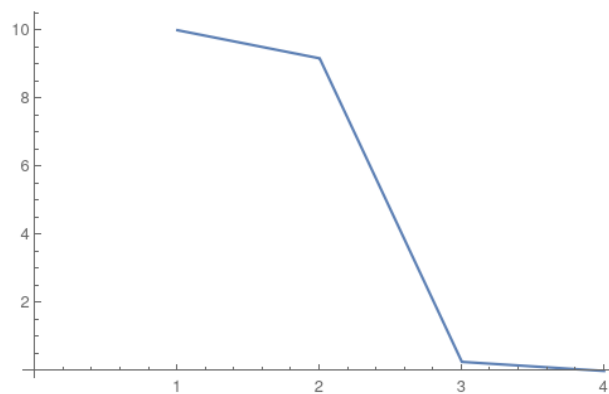
**Ejercicio 6.13**

Iteración	Error relativo porcentual
1	100
2	0.0000357



Ejercicio 6.19

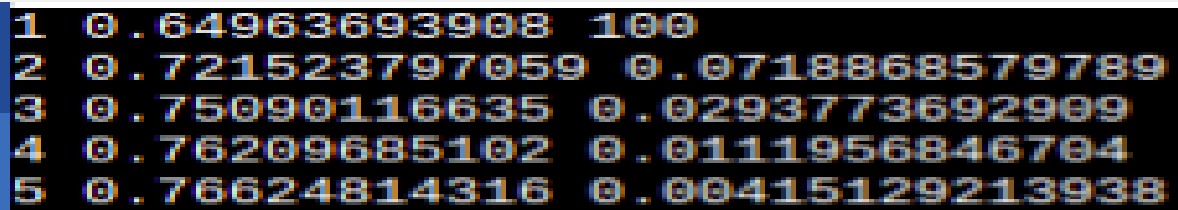
Iteración	Error relativo porcentual
1	10
2	9.174150
3	0.2666088
4	0.0003340



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

Ejercicio 6.1

Figura 4.1: Evidencia del funcionamiento



1	0.64963693908	100
2	0.721523797059	0.0718868579789
3	0.75090116635	0.0293773692909
4	0.76209685102	0.0111956846704
5	0.76624814316	0.00415129213938

Ejercicio 6.3

1. Punto fijo

Figura 4.2: Evidencia del funcionamiento

```
1 -14.705862999999999 100
2 -115.96228492715501 101.256421927
3 -7120.593465380606 7004.63118045
4 -26842648.67975988 26835528.0863
5 -381455336849641.9 3.81455310007e+14
6 -7.703362791137656e+28 7.70336279114e+28
7 -3.141620077550467e+57 3.14162007755e+57
```

2. Newton-Raphson

Figura 4.3: Evidencia del funcionamiento

```
2 0.7141749667496647 3.572595146117408e-05
```

Ejercicio 6.7

Figura 4.4: Evidencia del funcionamiento

1	-0.0232142784842	130.23080948
2	-1.22634747564	0.981070390778
3	0.233951216303	6.24189399405
4	0.396365773727	0.40975928849
5	0.944691165764	0.580428199087

Figura 4.5: Evidencia del funcionamiento

1	2.356928735	0.0607024144942
2	2.54728716043	0.0747298649291
3	2.52633908838	0.00829186871345

Figura 4.6: Evidencia del funcionamiento

1	1.92701799321	0.167607156721
2	1.95147933238	0.0125347672239
3	1.94460445794	0.00353535877774

Ejercicio 6.9

Figura 4.7: Evidencia del funcionamiento

```
2 -14.68 4521.409632
3 -9.210556393189425 1333.3198762137952
4 -5.597514435490806 391.86532674166983
5 -3.2249697167153863 114.94352764786103
```

Figura 4.8: Evidencia del funcionamiento

```
1 -1.28421052631579 3.7254098360655727
2 210.61247159807715 1.0060975046566403
3 -1.283667770416372 165.0708573136199
4 -1.2831253244627547 0.0004227536806229821
```

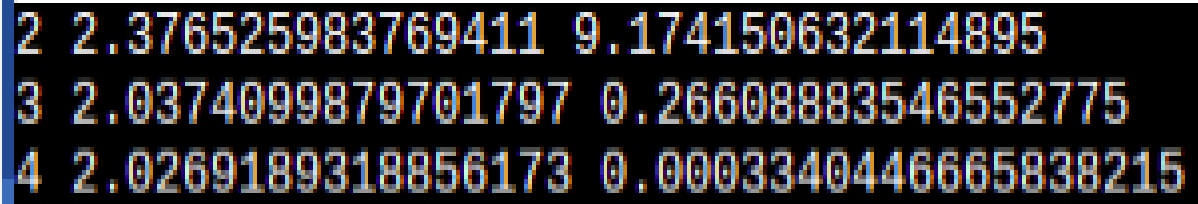
Ejercicio 6.13

Figura 4.9: Evidencia del funcionamiento

```
2 0.7141749667496647 3.572595146117408e-05
```

Ejercicio 6.19

Figura 4.10: Evidencia del funcionamiento



A screenshot of a terminal window with a black background and yellow text. The text is arranged in three lines, each starting with a line number (2, 3, 4) followed by two floating-point numbers separated by a space. The numbers are displayed in a monospaced font.

Line	Value 1	Value 2
2	2.376525983769411	9.174150632114895
3	2.0374099879701797	0.26608883546552775
4	2.0269189318856173	0.0003340446665838215

CAPÍTULO 5

BIBLIOGRAFÍA

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