

UNIVERSIDAD NACIONAL DEL ALTIPLANO  
FACULTAD DE ING. ESTADÍSTICA E INFORMÁTICA  
ESCUELA PROFESIONAL DE ING. ESTADÍSTICA E INFORMÁTICA



PROGRAMACIÓN NUMÉRICA

**i**DOCENTE:

Dr. TORRES CRUZ FRED

**i**ALUMNOS:

- AQUINO SANDOVAL JEAN CARLOS
- PARICAHUA PARI CLIDE NEIL
- QUENAYA LOZA LUIS ANGEL
- TICONA MIRAMIRA ROBERTO ANGEL

**i**SECCIÓN:

”IV SEMESTRE - A”

PUNO - PERÚ

2025



# Introducción

---

La programación numérica es una rama fundamental dentro de las ciencias computacionales y aplicadas, cuyo objetivo es proporcionar métodos y herramientas para resolver problemas matemáticos mediante el uso de algoritmos y programas informáticos. A través de ella, es posible aproximar soluciones a ecuaciones que no pueden resolverse de forma analítica, optimizar funciones complejas, y analizar sistemas dinámicos en diversas áreas del conocimiento.

El propósito de este libro es introducir al lector en los principios y aplicaciones prácticas de la programación numérica, utilizando lenguajes de programación de propósito científico como `Python` y `R`. A lo largo de los capítulos, se desarrollan los fundamentos teóricos y computacionales necesarios para comprender los métodos numéricos más empleados en ingeniería, física, economía y nutrición, entre otros campos.

En la primera unidad, se abordarán los conceptos básicos de la programación numérica, la definición de funciones matemáticas y sus restricciones, así como los métodos clásicos para el encontrar raíces de ecuaciones no lineales, tales como el método de bisección, el método de Newton-Raphson y el método de la secante. Estos métodos serán implementados paso a paso en código, permitiendo observar su comportamiento, eficiencia y convergencia.

En la segunda unidad, se explorarán métodos más avanzados, entre ellos el gradiente descendente y sus aplicaciones en optimización, la interpolación polinómica como técnica para aproximar funciones a partir de datos discretos, y la diferenciación numérica, que permite estimar derivadas de funciones cuando no se dispone de una expresión analítica. Cada tema será acompañado de ejemplos prácticos y ejercicios que ayudarán a reforzar la comprensión teórica mediante la experimentación computacional.

Además, se presentarán recomendaciones sobre buenas prácticas en la escritura de código científico, el uso eficiente de librerías numéricas, y la validación de resultados mediante análisis de error. El enfoque de este libro combina la precisión matemática con la aplicación práctica, fomentando el desarrollo de habilidades analíticas y computacionales en el lector.

En conjunto, este material busca no solo enseñar los fundamentos de la programación numérica, sino también promover un pensamiento crítico y estructurado al enfrentar problemas reales que requieren soluciones computacionales.



# Índice general

---

Introducción	3
Índice de Figuras	7
I Unidad I	9
1. Programación Numérica	11
2. Funciones	13
2.1. Funciones a nuestro alrededor . . . . .	13
2.2. Definición de función . . . . .	13
2.3. Cuatro formas de representar una función . . . . .	14
2.4. Gráficas de funciones . . . . .	15
2.4.1. Gráficas de funciones por localización de puntos . . . . .	15
2.4.2. La prueba de la recta vertical . . . . .	16
2.5. Aplicación . . . . .	17
3. Restricciones	19
3.1. Ejemplo . . . . .	19
4. Método de Newton Raphson	23
4.1. Fundamento teórico . . . . .	23
4.2. Condiciones de convergencia . . . . .	23
4.3. Criterio de parada . . . . .	24
4.4. Algoritmo del método de Newton-Raphson . . . . .	24
4.5. Aplicación del método en Python . . . . .	24
5. Método de Bisección	29

6. Método de la Secante	31
7. Método de Punto Fijo	33
8. Método de Regula Falsi	35
 II Unidad II	 37
9. Gradiente de una función	39
10. Diferenciación Numérica	41
11. Interpolación	43
Conclusiones	45
Bibliografía	47

# Índice de figuras

---

2.1. Diagrama de flechas de $f$ . . . . .	14
2.2. Cuatro formas de representar una función . . . . .	15
2.3. La altura de la gráfica arriba del punto $x$ es el valor de $f(x)$ . . . . .	16
2.4. Funciones lineal y constante . . . . .	16
2.5. Prueba de la recta vertical . . . . .	17
2.6. Graficando funciones con Python . . . . .	18
3.1. Aplicación de restricciones en Python . . . . .	20
4.1. Método de Newthton-Rapshon en Python . . . . .	27





## Parte I

### Unidad I



# 1 Programación Numérica

---

La programación numérica es una disciplina que combina las matemáticas aplicadas y la informática con el objetivo de resolver problemas cuantitativos mediante métodos computacionales. Se centra en el diseño, análisis e implementación de algoritmos que permiten obtener soluciones aproximadas a ecuaciones, sistemas y modelos que, en la mayoría de los casos, no pueden resolverse de forma analítica (Burden & Faires, 2016b; Chapra & Canale, 2015c).

A diferencia de la programación convencional, que busca desarrollar aplicaciones funcionales o sistemas de información, la programación numérica se orienta a la resolución eficiente y precisa de problemas matemáticos. Entre sus principales aplicaciones se encuentran la simulación de fenómenos físicos y biológicos, la modelización económica, la ingeniería de datos, la inteligencia artificial y el análisis estadístico en ciencias de la salud (Press et al., 2007).

El objetivo fundamental de esta área es transformar problemas continuos en representaciones discretas que puedan ser tratadas por un computador. De esta forma, se logra aproximar soluciones a problemas de optimización, integración, derivación, interpolación, ajuste de curvas y resolución de ecuaciones diferenciales.

En términos prácticos, la programación numérica permite al investigador o profesional:

- Resolver ecuaciones no lineales mediante métodos iterativos.
- Aproximar derivadas e integrales de funciones cuando no se dispone de una forma analítica.
- Interpolarse o ajustar funciones a datos experimentales.
- Optimizar funciones de una o varias variables bajo restricciones.
- Analizar errores y estimar la estabilidad numérica de los métodos empleados.

Actualmente, lenguajes como **Python**, **R**, **MATLAB** y **Julia** ofrecen bibliotecas especializadas que facilitan el desarrollo de algoritmos numéricos de alto rendimiento. Estos entornos han hecho posible que la programación numérica sea una herramienta accesible y poderosa para la investigación científica, la ingeniería y la docencia (Chapra & Canale, 2015c).

En síntesis, la programación numérica constituye una base esencial para la solución computacional de problemas científicos y técnicos, integrando el razonamiento matemático con la capacidad de cómputo moderna.



## 2 Funciones

---

### 2.1 Funciones a nuestro alrededor

En casi todos los fenómenos físicos observamos que una cantidad depende de otra. Por ejemplo, la estatura de una persona depende de su edad, la temperatura de la fecha, el costo de enviar un paquete por correo depende de su peso. Usamos el término función para describir esta dependencia de una cantidad con respecto a otra. Esto es, decimos lo siguiente: (Stewart et al., 2001)

- La estatura es una función de la edad.
- La temperatura es una función de la fecha.
- El costo de enviar un paquete por correo depende de su peso.

### 2.2 Definición de función

Una función  $f$  es una regla que asigna a cada elemento  $x$  de un conjunto  $A$  exactamente un elemento, llamado  $f(x)$ , de un conjunto  $B$ .

Para hablar de una función, es necesario darle un nombre. Usaremos letras como  $f, g, h, \dots$  para representar funciones. Por ejemplo, podemos usar la letra  $f$  para representar una regla como sigue:

"  $f$  " es la regla "elevar al cuadrado el número"

cuando escribimos  $f(2)$  queremos decir "aplicar la regla  $f$  al número 2". La aplicación de la regla da  $f(2) = 2^2 = 4$ . Del mismo modo,  $f(3) = 3^2 = 9$ ,  $f(4) = 4^2 = 16$ , y en general  $f(x) = x^2$ . (Stewart et al., 2001)

Por lo general consideramos funciones para las cuales los conjuntos  $A$  y  $B$  son conjuntos de número reales. El símbolo  $f(x)$  se lee "f de x" o "f en x" y se denomina valor de  $f$  en  $x$ , o la imagen de  $x$  bajo  $f$ . El conjunto  $A$  recibe el nombre de dominio de la función. El rango de  $f$  es el conjunto de todos los valores posibles de  $f(x)$  cuando  $x$  varía en todo el dominio. El símbolo que representa un número arbitrario del dominio de una función  $f$  se llama variable independiente. El símbolo que representa un número en el rango de  $f$  se llama variable dependiente. Por tanto, si escribimos  $y = f(x)$ , entonces  $x$  es la variable independiente y  $y$  es la variable dependiente. (Stewart et al., 2001)

Es útil considerar una función como una [máquina](#). Si  $x$  está en el dominio de la función  $f$ , entonces cuando  $x$  entra a la máquina, es aceptada como entrada y la máquina produce una salida  $f(x)$  de acuerdo con la regla de la función. Así, podemos considerar el dominio como el conjunto de todas las posibles entradas y el rango como el conjunto de todas las posibles salidas. (Stewart et al., [2001](#))

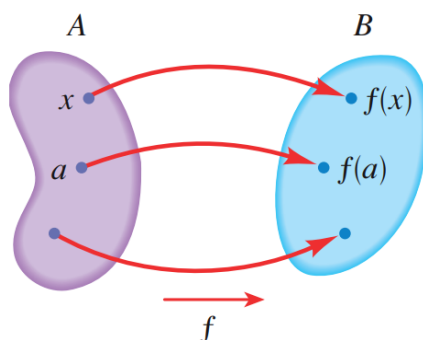


Figura 2.1: Diagrama de flechas de  $f$

### 2.3 Cuatro formas de representar una función

Para entender mejor lo que es una función, podemos describir una función específica en las siguientes cuatro formas: (Stewart et al., [2001](#))

- verbalmente (por descripción en palabras)
- algebraicamente (por una fórmula explícita)
- visualmente (por una gráfica)
- numéricamente (por una tabla de valores)

Una función individual puede estar representada en las cuatro formas, y con frecuencia es útil pasar de una representación a otra para adquirir más conocimientos sobre la función. No obstante, ciertas funciones se describen en forma más natural por medio de un método que por los otros. Un ejemplo de una descripción verbal es la siguiente regla para convertir entre escalas de temperatura: (Stewart et al., [2001](#))

”Para hallar el equivalente Fahrenheit de una temperatura Celsius, multiplicar por  $\frac{9}{5}$  la temperatura Celsius y luego sumar 32”

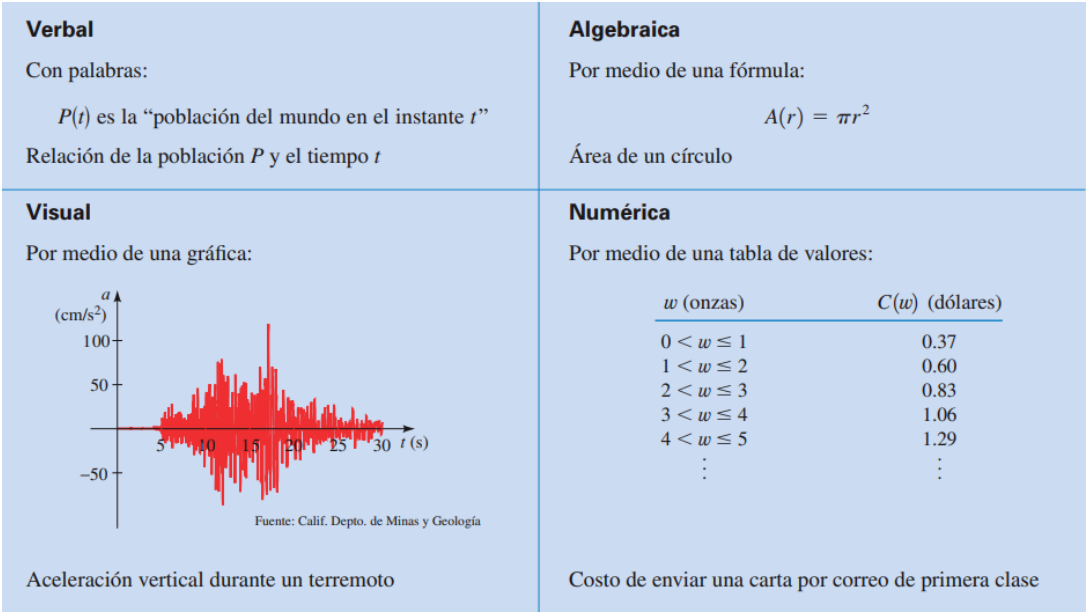


Figura 2.2: Cuatro formas de representar una función

2.4 Gráficas de funciones

2.4.1 Gráficas de funciones por localización de puntos

Para graficar una función  $f$  localizamos los puntos  $(x, f(x))$  en un plano de coordenadas. En otras palabras, localizamos los puntos  $(x, y)$  cuya coordenada  $x$  es una entrada y cuya coordenada  $y$  es la correspondiente salida de la función. (Stewart et al., 2001)

Si  $f$  es una función con dominio  $A$ , entonces la gráfica de  $f$  es el conjunto de pares ordenados

$$(x, f(x)) | x \in A$$

localizados en un plano de coordendas. En otras palabras, la gráfica de  $f$  es el conjunto de todos los puntos  $(x, y)$  tales que  $y = f(x)$ ; esto es, la gráfica de  $f$  es la gráfica de la ecuación  $y = f(x)$ .

La gráfica de una función  $f$  da un retrato del comportamiento o “historia de la vida” de la función. Podemos leer el valor de  $f(x)$  a partir de la gráfica como la altura de la gráfica arriba del punto  $x$ . (Stewart et al., 2001)

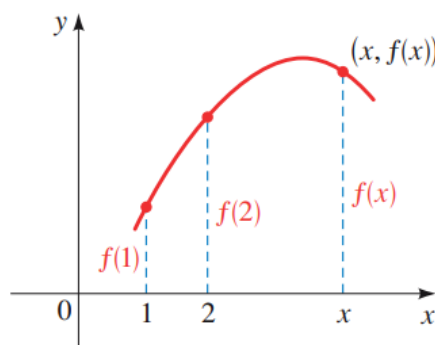


Figura 2.3: La altura de la gráfica arriba del punto  $x$  es el valor de  $f(x)$

Una función  $f$  de la forma  $f(x) = mx + b$  se denomina función lineal porque su gráfica es la gráfica de la ecuación  $y = mx + b$ , que representa una recta con pendiente  $m$  y punto de intersección  $b$  en  $y$ . Un caso especial de una función lineal se presenta cuando la pendiente es  $m = 0$ . La función  $f(x) = b$ , donde  $b$  es un número determinado, recibe el nombre de *funcinconstante* porque todos sus valores son el mismo número, es decir,  $b$ . Su gráfica es la recta horizontal  $y = b$ . (Stewart et al., 2001)

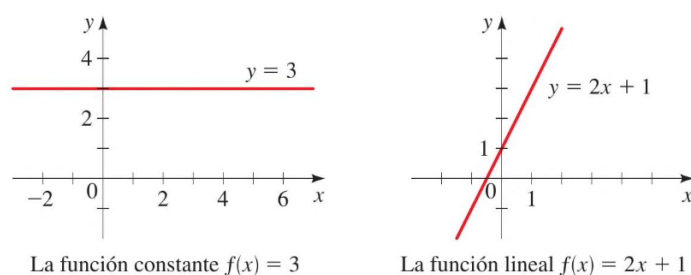


Figura 2.4: Funciones lineal y constante

### 2.4.2 La prueba de la recta vertical

La gráfica de una función es una curva en el plano  $xy$ . Pero surge la pregunta. ¿Cuáles curvas del plano  $xy$  son gráficas de funciones? ESTo se contesta por medio de la prueba siguiente. (Stewart et al., 2001)

Una curva en el plano de coordenadas es la gráfica de una función si y sólo si ninguna recta vertical cruza la curva más de una vez.

Podemos ver la Figura 2.5 para entender por qué la Prueba de la Recta Vertical es verdadera. Si cada recta vertical  $x = a$  cruza la curva sólo una vez en  $(a, b)$ , entonces exactamente un valor funcional está definido por  $f(a) = b$ . Pero si una recta  $x = a$  cruza la curva dos veces, en  $(a, b)$  y en  $(a, c)$ , entonces la curva no puede representar una función porque una función no puede asignar dos valores diferentes a  $a$ .



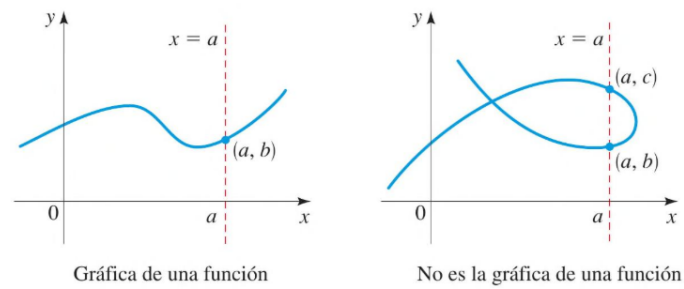


Figura 2.5: Prueba de la recta vertical

## 2.5 Aplicación

Se presentará un código en Python que sea capaz de graficar funciones, según los datos de entrada que se pidan.

```

1 import sympy as sp
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 # === Ingreso de la función por el usuario ===
6 expr_str = input("Ingrese la función en términos de x (ejemplo: sin(x), x**2 + 3*x
   ↪ - 5, exp(-x)*cos(x)): ")
7
8 # === Definición de variable simbólica ===
9 x = sp.Symbol('x')
10
11 # === Conversión del texto a expresión simbólica ===
12 try:
13     expr = sp.sympify(expr_str)
14 except sp.SympifyError:
15     print("Error: la función ingresada no es válida.")
16     exit()
17
18 # === Creación de función numérica evaluable ===
19 f = sp.lambdify(x, expr, modules=['numpy'])
20
21 # === Intervalo de graficación ===
22 x_vals = np.linspace(-10, 10, 400)
23 y_vals = f(x_vals)
24
25 # === Graficar ===
26 plt.figure(figsize=(7,5))
27 plt.plot(x_vals, y_vals, label=f"$f(x) = {sp.latex(expr)}$", color='navy')
28 plt.title("Gráfica de la función ingresada", fontsize=13)
29 plt.xlabel("x")
30 plt.ylabel("f(x)")
31 plt.grid(True, linestyle='--', alpha=0.6)
32 plt.axhline(0, color='black', linewidth=1)
33 plt.axvline(0, color='black', linewidth=1)
34 plt.legend()

```

```
35 plt.show()
```

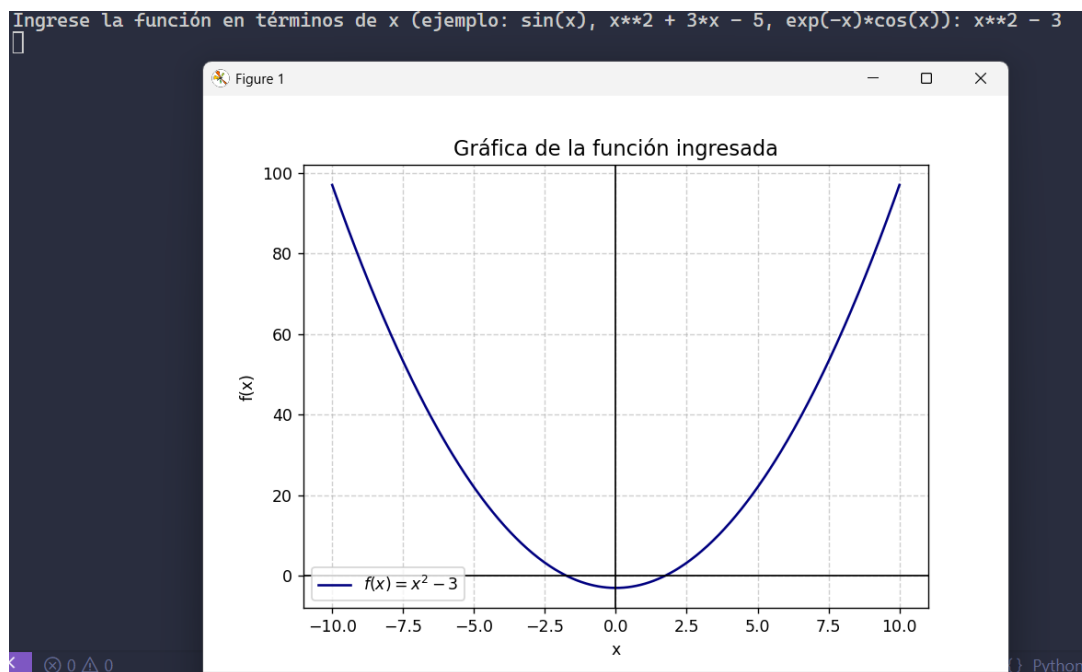


Figura 2.6: Graficando funciones con Python

## 3 Restricciones

---

Las restricciones son condiciones que limitan el conjunto de valores posibles que pueden tomar las variables en un problema matemático. Estas limitaciones definen el espacio factible o región factible, dentro del cual se busca una o más soluciones que cumplan con determinados criterios. Las restricciones pueden ser igualdades (por ejemplo,  $x + y = 10$ ) o desigualdades (por ejemplo,  $x \geq 0$ ), y desempeñan un papel fundamental en los problemas de optimización, programación lineal y análisis numérico. (Anton et al., 2012; Burden & Faires, 2016a; Chapra & Canale, 2015a)

Un sistema de ecuaciones consiste en un conjunto de ecuaciones que comparten las mismas variables. Resolverlo implica encontrar los valores que satisfacen todas las ecuaciones simultáneamente. Estos sistemas pueden clasificarse en lineales y no lineales, y sus métodos de resolución varían desde técnicas algebraicas clásicas (como la sustitución o igualación) hasta procedimientos numéricos avanzados (como el método de Gauss-Seidel o Newton-Raphson). (Burden & Faires, 2016a; Chapra & Canale, 2015a; Riley et al., 2006)

En el contexto de la programación numérica, las restricciones se representan y manipulan mediante código, permitiendo no solo resolver el sistema de ecuaciones sino también visualizar gráficamente la región factible y el punto óptimo. Esta representación es especialmente útil para comprender la interacción entre las ecuaciones y las limitaciones impuestas, facilitando el análisis y la toma de decisiones en problemas aplicados de ingeniería, economía y ciencias computacionales. (Chapra & Canale, 2015a; Stewart et al., 2001)

### 3.1 Ejemplo

Un desarrollador tiene 15 horas semanales para dedicar al desarrollo de software de front-end (x) y back-end (y). Además:

- Debe dedicar al menos 5 horas al desarrollo de front-end para cumplir con los entregables del cliente.
- El tiempo total no puede exceder 15 horas por restricciones de tiempo del sprint.

Formule las restricciones, represéntelas gráficamente e identifique las combinaciones posibles de tiempo a invertir en cada actividad.

Variables

- $x$  = horas dedicadas al front-end
- $y$  = horas dedicadas al back-end

## Restricciones

1. Debe dedicar al menos 5 horas al mes.

$$\blacksquare x \geq 5$$

2. El tiempo total no puede exceder a 15 horas.

$$\blacksquare x + y \leq 15$$

Gráfico generado con python



Figura 3.1: Aplicación de restricciones en Python

## Interpretación del gráfico

En la representación gráfica se observa la región factible determinada por las restricciones planteadas:

- $x \geq 5$ : el desarrollador debe dedicar al menos 5 horas al front-end. Esta condición aparece como una línea vertical en  $x = 5$  y la región válida se encuentra a la derecha de ella.
- $y \geq 0$ : el tiempo dedicado al back-end no puede ser negativo, por lo que la región se limita a la parte superior del eje  $x$ .
- $x + y \leq 15$ : la suma de horas asignadas a front-end y back-end no puede superar las 15 horas semanales. Gráficamente, corresponde a la semirrecta bajo la línea  $x + y = 15$ .

La intersección de estas tres restricciones genera una región triangular factible delimitada por los puntos  $(5, 0)$ ,  $(15, 0)$  y  $(5, 10)$ . Esto significa que cualquier combinación de horas ubicada dentro o sobre este triángulo cumple con las condiciones del problema. Por ejemplo, el desarrollador podría dedicar:

- 5 horas a front-end y 10 horas a back-end,
- 10 horas a front-end y 5 horas a back-end,
- o bien 15 horas únicamente a front-end.

En conclusión, la región sombreada representa todas las combinaciones posibles de tiempo de trabajo entre front-end y back-end que respetan tanto el mínimo requerido en front-end como la restricción máxima de 15 horas semanales.

Código en Python

```

1      # Función para preparar expresiones lineales
2  def preparar_expresion(expr: str) -> str:
3      expr = expr.replace(" ", "")          # quitar espacios
4      expr = expr.replace("^", "**")        # potencia
5      expr = expr.replace("-x", "-1*x")    # caso -x
6      expr = expr.replace("+x", "+1*x")    # caso +x
7      if expr.startswith("x"):             # si empieza con x
8          expr = "1*" + expr
9      expr = expr.replace("x", "*x")       # poner multiplicación
10     expr = expr.replace("**x", "*x")     # corregir si se duplicó
11     return expr
12
13     # Restricciones:
14     # 1) x = 5 (vertical)
15     # 2) x + y = 15 -> y = -x + 15
16     func2 = preparar_expresion("-x+15")
17
18     # Rango de la gráfica
19     xmin, xmax = -5, 20
20     ymin, ymax = -5, 20
21
22     # Recorremos el plano
23     for y in range(ymax, ymin - 1, -1):
24         linea = ""

```

```
25 for x in range(xmin, xmax + 1):
26     # Recta 1:  $x = 5$ 
27     cond1 = (x == 5)
28
29     # Recta 2:  $y = -x + 15$ 
30     try:
31         y2 = eval(func2)
32     except:
33         y2 = None
34     cond2 = (y2 is not None and abs(y - y2) < 0.5)
35
36     # Región factible:  $x \geq 5$ ,  $y \geq 0$ ,  $x+y \leq 15$ 
37     region = (x >= 5 and y >= 0 and x + y <= 15)
38
39     # Qué dibujar
40     if cond1 and cond2:
41         linea += "#"
42     elif cond1:
43         linea += "*"
44     elif cond2:
45         linea += "o"
46     elif x == 0 and y == 0:
47         linea += "+"
48     elif x == 0:
49         linea += "|"
50     elif y == 0:
51         linea += "-"
52     elif region:
53         linea += "."
54     else:
55         linea += " "
56     print(linea)
57
58     # Leyenda
59     print("\nLeyenda del gráfico:")
60     print(" * =  $x = 5$ ")
61     print(" o =  $x + y = 15$ ")
62     print(" # = Intersección")
63     print(" . = Región factible")
64     print(" | = Eje Y")
65     print(" - = Eje X")
66     print(" + = Origen (0,0)")
```

## 4 Método de Newton Raphson

---

El método de Newton-Raphson es una técnica iterativa utilizada para encontrar raíces de ecuaciones no lineales de la forma:

$$f(x) = 0$$

Es uno de los métodos más eficaces y ampliamente utilizados debido a su rapidez de convergencia cuando se cumplen las condiciones necesarias. Fue propuesto originalmente por Isaac Newton y posteriormente generalizado por Joseph Raphson (Chapra & Canale, 2015b).

### 4.1 Fundamento teórico

La idea básica del método consiste en aproximar la función  $f(x)$  mediante su expansión de Taylor alrededor de un punto  $x_i$  y despreciar los términos de orden superior:

$$f(x) \approx f(x_i) + f'(x_i)(x - x_i)$$

Para hallar la raíz, se hace  $f(x) = 0$ , y despejando  $x$  se obtiene la fórmula iterativa:

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

Esta expresión permite calcular una mejor aproximación de la raíz en cada iteración. El proceso continúa hasta que el valor de  $x_{i+1}$  converge a la raíz real con una tolerancia previamente establecida (Chapra & Canale, 2015b).

### 4.2 Condiciones de convergencia

El método de Newton-Raphson presenta convergencia cuadrática, es decir, el error disminuye aproximadamente al cuadrado en cada iteración, siempre que se cumplan las siguientes condiciones (Süli & Mayers, 2003):

- La función  $f(x)$  es continua y derivable en un intervalo que contiene la raíz buscada.
- La derivada  $f'(x)$  no se anula en el entorno de la raíz.
- La estimación inicial  $x_0$  está suficientemente cerca de la raíz real.

Sin embargo, si  $f'(x_i)$  se aproxima a cero o si la estimación inicial está muy alejada, el método puede divergir o generar oscilaciones (Burden & Faires, 2011).

### 4.3 Criterio de parada

El proceso iterativo se detiene cuando se cumple alguna de las siguientes condiciones (Chapra & Canale, 2015b):

- $|f(x_{i+1})| < \varepsilon$
- $|x_{i+1} - x_i| < \varepsilon$

donde  $\varepsilon$  es la tolerancia o el error máximo permitido.

### 4.4 Algoritmo del método de Newton-Raphson

1. Elegir una estimación inicial  $x_0$ .

2. Calcular  $f(x_0)$  y  $f'(x_0)$ .

3. Evaluar:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

4. Repetir el proceso:

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

hasta que se cumpla el criterio de convergencia.

### 4.5 Aplicación del método en Python

El objetivo fue desarrollar en el lenguaje de programación Python un programa que permita al usuario ingresar una función  $f(x)$  y graficarla en un intervalo definido. Esta etapa inicial tiene como propósito ayudar al usuario a identificar visualmente las posibles raíces y decidir si desea aplicar el método de Newton-Raphson.

En caso afirmativo, el programa solicita un valor inicial  $x_1$  basado en la observación de la gráfica y ejecuta el algoritmo iterativo de Newton-Raphson hasta que la diferencia entre dos aproximaciones sucesivas sea menor a una tolerancia predefinida. Finalmente, el programa muestra en pantalla la raíz aproximada encontrada y el número de iteraciones necesarias para alcanzarla.

Este enfoque combina la interpretación gráfica con el análisis numérico, promoviendo una comprensión más completa del comportamiento de la función y de la eficacia del método iterativo.

Entrada

Una cadena de texto que representa una función matemática.



$$f(x) = x^3 - x - 1$$

Salida

- Gráfica para evaluar si realizar o no el método.
- La raíz encontrada.
- Número de iteraciones realizadas hasta encontrar la raíz.

Código

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 # === 1. Ingreso de la función ===
5 func_str = input("Ingrese la función f(x): ") # Ejemplo: x**3 - x - 1
6
7 # Definimos la función y su derivada (usando derivada numérica)
8 def f(x):
9     return eval(func_str)
10
11 def f_prime(x, h=1e-6): # derivada numérica
12     return (f(x + h) - f(x - h)) / (2 * h)
13
14 # === 2. Graficar la función ===
15 xmin = float(input("Ingrese el valor mínimo de x: "))
16 xmax = float(input("Ingrese el valor máximo de x: "))
17
18 x = np.linspace(xmin, xmax, 400)
19 y = f(x)
20
21 plt.figure(figsize=(8, 5))
22 plt.plot(x, y, label=f"f(x) = {func_str}", color='blue')
23 plt.axhline(0, color='black', linestyle='--')
24 plt.axvline(0, color='black', linestyle='--')
25 plt.title("Gráfico de la función ingresada")
26 plt.xlabel("x")
27 plt.ylabel("f(x)")
28 plt.legend()
29 plt.grid(True)
30 plt.show()
31
32 # === 3. Pregunta si desea aplicar Newton-Raphson ===
33 op = input("¿Desea encontrar una raíz con el método de Newton-Raphson? (s/n): ").
34     ↪ lower()
35
36 if op == "s":
37     # === 4. Ingreso de punto inicial ===

```

```
37 x0 = float(input("Basado en la gráfica, ingrese el valor inicial x1: "))
38
39 # Parámetros del método
40 tol = 1e-6
41 max_iter = 100
42
43 # Iteraciones
44 for i in range(1, max_iter + 1):
45     fx = f(x0)
46     fpx = f_prime(x0)
47
48     if fpx == 0:
49         print(f"La derivada es cero en x = {x0}. El método no puede continuar.")
50         break
51
52     x1 = x0 - fx / fpx
53
54 # Verificar convergencia
55 if abs(x1 - x0) < tol:
56     print(f"\n Raíz aproximada encontrada: {x1:.6f}")
57     print(f"Iteraciones realizadas: {i}")
58     break
59
60 x0 = x1
61 else:
62     print("\n No se alcanzó la convergencia después de", max_iter, "iteraciones.")
63
64 else:
65     print("No se aplicó el método de Newton-Raphson.")
```

Ejecución

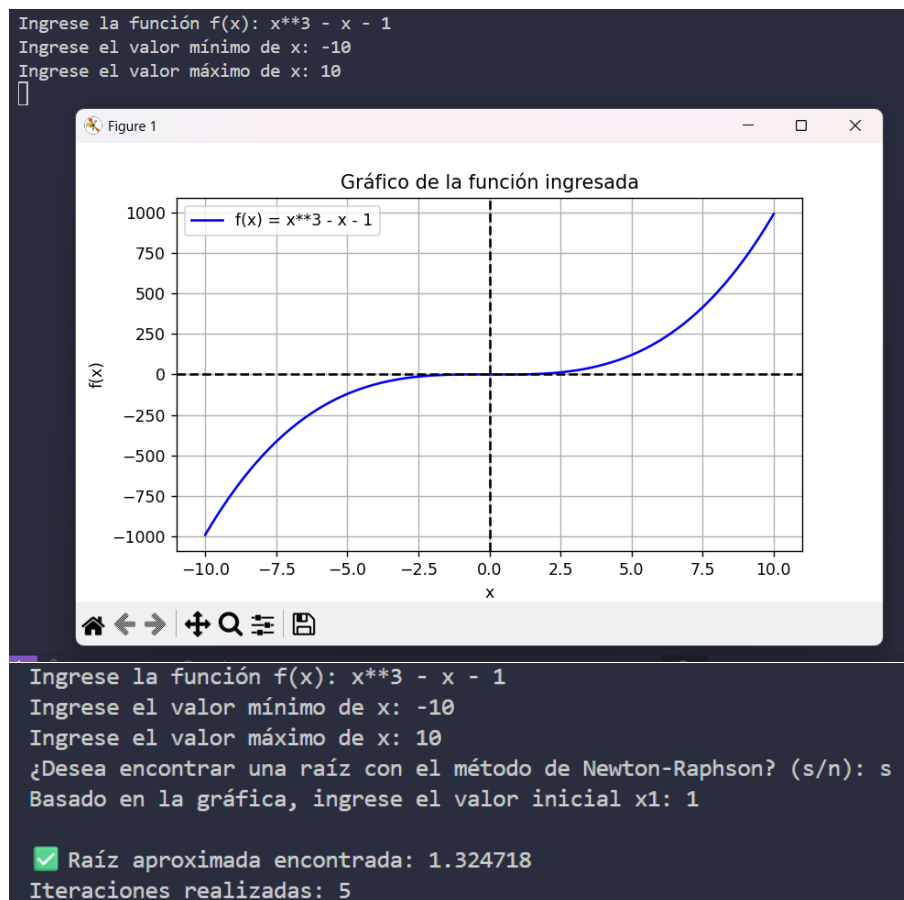


Figura 4.1: Método de Newton-Raphson en Python



## 5 Método de Bisección

---



## 6 Método de la Secante

---





## 7 Método de Punto Fijo

---



## 8 Método de Regula Falsi

---



## Parte II

### Unidad II



## 9 Gradiente de una función

---

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.





## 10 Diferenciación Numérica

---



## 11 Interpolación

---



## Conclusiones

---

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.



## Bibliografía

---





# Bibliografía

---

- Anton, H., Bivens, I., & Davis, S. (2012). Cálculo (10.<sup>a</sup> ed.). Wiley.
- Burden, R. L., & Faires, J. D. (2011). Análisis Numérico (9na ed.). Cengage Learning.
- Burden, R. L., & Faires, J. D. (2016a). Análisis Numérico (10.<sup>a</sup> ed.). Cengage Learning.
- Burden, R. L., & Faires, J. D. (2016b). Análisis numérico (10.<sup>a</sup> ed.). Cengage Learning.
- Chapra, S. C., & Canale, R. P. (2015a). Métodos Numéricos para Ingenieros (7.<sup>a</sup> ed.). McGraw-Hill Education.
- Chapra, S. C., & Canale, R. P. (2015b). Métodos Numéricos para Ingenieros (7ma ed.). McGraw-Hill Education.
- Chapra, S. C., & Canale, R. P. (2015c). Métodos numéricos para ingenieros (7.<sup>a</sup> ed.). McGraw-Hill Education.
- Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). Numerical Recipes: The Art of Scientific Computing (3.<sup>a</sup> ed.). Cambridge University Press.
- Riley, K. F., Hobson, M. P., & Bence, S. J. (2006). Mathematical Methods for Physics and Engineering (3.<sup>a</sup> ed.). Cambridge University Press.
- Stewart, J., Redlin, L., & Watson, S. (2001). Precálculo: Matemáticas para el Cálculo (5.<sup>a</sup> ed.). Thomson.
- Süli, E., & Mayers, D. F. (2003). An Introduction to Numerical Analysis. Cambridge University Press.