UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE QUÍMICA

MÉTODOS NUMÉRICOS APLICADOS A LA SOLUCIÓN DE PROBLEMAS DE INGENIERÍA QUÍMICA UTILIZANDO MATHEMATICA 10.0

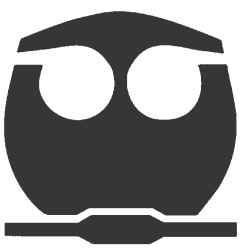
T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO QUÍMICO

PRESENTA

ALBERTO ALFARO-VICTORIA LÓPEZ



CIUDAD UNIVERSITARIA, CD. MX, 2016

JURADO ASIGNADO:

PRESIDENTE:	**Profesora: GUADALUPE JOSEFINA TOLEDO MACÍAS**
VOCAL:	**Profesor: EUGENIO LEÓN FAUTSCH TAPIA**
SECRETARIO:	**Profesor: MANUEL VÁZQUEZ ISLAS**
1er. SUPLENTE:	**Profesor: ERNESTO JOSÉ CALDERÓN CASTILLO**
2do. SUPLENTE:	**Profesor: CARLOS ÁLVAREZ MACIEL**

SITIO DONDE SE DESARROLLÓ EL TEMA:

FACULTAD DE QUÍMICA, CIUDAD UNIVERSITARIA.

ASESOR DEL TEMA:

_____

I.Q. MANUEL VÁZQUEZ ISLAS

SUSTENTANTE:

_____

ALBERTO ALFARO-VICTORIA LÓPEZ

Índice.

Introducción.

La disponibilidad de computadoras de alta velocidad, los lenguajes de programación de alto nivel y el acceso a paquetes matemáticos potentes han impactado de manera importante la educación y la práctica de la ingeniería en los últimos años. Durante las décadas pasadas, la resolución de problemas de ingeniería se llevaba a cabo principalmente con calculadoras de mano. En ocasiones se utilizaban computadoras que no eran accesibles a cualquier usuario y requerían un entrenamiento especial para ser utilizadas.

En ingeniería química y en la investigación científica es común encontrarse con problemas que no tienen solución analítica, es decir, que requieren del uso de técnicas numéricas para resolverse. El acceso cada vez mayor a las computadoras ha facilitado esta labor considerablemente. Sin embargo, la falta de conocimiento en el uso de software matemático y la escasa capacitación de muchos estudiantes en lenguajes de programación han sido un impedimento para hacer frente a estos problemas.

El **objetivo** del presente proyecto es proporcionar información que facilite la resolución de problemas de ingeniería química mediante métodos numéricos. Se escogió el software Mathematica 10.0 debido a que tiene ventajas importantes que se pueden aplicar para conseguir resultados de la manera más eficiente.

Capítulo 1. Fundamentos previos.

1.1. Métodos numéricos.

Los métodos numéricos son una manera de convertir operaciones matemáticas superiores en operaciones aritméticas básicas. Son ampliamente utilizados en ingeniería debido a su capacidad de manipular sistemas de ecuaciones grandes, manejar no linealidades y resolver problemas complejos cuya solución analítica es inexistente o difícil de obtener [1].

Para identificar la diferencia entre una solución analítica y una solución numérica de manera sencilla, podemos considerar la siguiente función:

$$f(x) = x - 5 \tag{1-1}$$

Si se quiere encontrar la raíz, es decir, conocer el valor donde f(x)=0, sumamos 5 de ambos lados:

$$x = 5 \tag{1-2}$$

De este modo se obtiene una solución analítica.

Una manera de llegar al resultado numéricamente podría ser el método de Newton-Raphson, el cual se analizará a profundidad en el Capítulo 3. Este método consiste en tomar un valor inicial y aplicar la fórmula:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \tag{1-3}$$

Si tomamos 2 como estimado inicial y lo sustituimos en la ecuación (1-3):

$$x_1 = 2 - \frac{2-5}{1} \tag{1-4}$$

$$x_1 = 5$$

Después, el nuevo valor calculado se vuelve a introducir en la fórmula de Newton-Raphson.

$$x_2 = 5 - \frac{5-5}{1} \tag{1-4}$$

$$x_2 = 5$$

Como x_2 tuvo el mismo valor que el calculado anteriormente, se dice que el método ha alcanzado la convergencia y por lo tanto, esa es nuestra solución. Sin embargo, no siempre es posible tener el mismo valor. Por este motivo, se define un pequeño número ε que se utiliza para crear un criterio de aceptación de la solución. Si la nueva solución es menor a ese número ε, entonces se acepta y se dan por terminados los cálculos.

$$|valor\ anterior - valor\ nuevo| < \varepsilon \tag{1-5}$$

Por ejemplo, si escogemos $\varepsilon = 10^{-6}$, aceptamos la respuesta, ya que se cumple la condición $(5 - 5) < \varepsilon$.

Hasta ahora, el ejemplo de la ecuación (1-1) se abordó mediante una secuencia de operaciones aritméticas más un criterio de aceptación. De esta forma se llegó a la solución numérica.

En este caso, la obtención de la solución analítica era demasiado sencilla; sólo bastaba sumar 5 en ambos lados para conseguirla. Desgraciadamente, los

problemas a los que se enfrentan los ingenieros químicos no son tan triviales como para resolverse con una simple suma. Esa es la razón por la que deben estudiarse técnicas de análisis numérico. Si a esto se le suman las capacidades de una computadora y un lenguaje de programación, se tiene un conjunto de herramientas que permitirán hacer frente a una gran cantidad de problemas complejos.

La mayoría de los métodos que se verán en este trabajo se resumen en el Apéndice.

1.2. Mathematica 10.0 y el lenguaje Wolfram.

Los métodos numéricos producen una secuencia de aproximaciones luego de realizar el mismo procedimiento una y otra vez. Esto puede resultar extenuante para una persona, pero no para una computadora. La importancia de programar radica en que nos permite delegar tareas repetitivas y agobiantes a una máquina, pero para ello es necesario comprender la lógica y la forma en que la computadora recibe instrucciones.

El software Mathematica 10.0, desarrollado por Wolfram Research, es una de las mejores opciones de software matemático que existen actualmente. Su característica más destacada es el potente lenguaje de programación que utiliza: ***The Wolfram Language***. Este lenguaje, reconocido por su facilidad de uso, sobresale en varias áreas de la computación técnica. Esto debido a que el código es muy intuitivo y fácil de leer, pues permite la entrada de lenguaje natural. Por ejemplo, consideremos la siguiente ecuación:

$$P = \frac{nRT}{V - nb} - \frac{n^2 a}{V^2} \tag{1-5}$$

A la ecuación (1-5) se le conoce como *la ecuación de Van der Waals*, y nos permite calcular la presión de un gas en función de su temperatura y volumen. En la tabla 1.1 se muestra una comparación de cómo se escribiría esta misma ecuación en Wolfram y en otros lenguajes:

Lenguaje	Ecuación de Van der Waals
Java	`P =(n*R*T)/(V-n*b)-(Math.pow(n,2)*a/Math.pow(V,2))`
C++	`P =(n*R*T)/(V-n*b)-(pow(n, 2)*a/pow(V, 2))`
Python	`P =(n*R*T)/(V-n*b)-(n**2)*a/V**2`
Matlab	`P =(n*R*T)/(V-n*b)-(n^2)*a/V^2`
Wolfram	$P = \frac{n\,R\,T}{V - n\,b} - \frac{n^2\,a}{V^2}$

Tabla 1.1. Ecuación de Van der Waals escrita en distintos lenguajes de programación.

Como se puede ver en la tabla 1.1, la ecuación es por mucho, más fácil de leer en el lenguaje Wolfram. Esto no quiere decir que los demás sean inútiles; cada uno es excelente en su respectivo campo. Y Wolfram es el más apropiado para lo que se hará en el presente trabajo. La ecuación podría haberse escrito usando la diagonal invertida para dividir, pero este lenguaje permite insertar barras de fracción, así como símbolos de varios operadores. Al usarlos, la legibilidad del código mejora y se facilita la tarea de localizar errores.

A continuación se dará una breve introducción al uso de Mathematica 10.0 y su lenguaje de programación.

1.2.1. Introduciendo símbolos en Mathematica 10.0.

Hasta ahora se ha visto que el lenguaje Wolfram permite introducir ecuaciones tal como se verían escritas en un cuaderno de apuntes. Esto es porque el software Mathematica trabaja bajo esa filosofía. Los archivos creados en este software tienen extensión .nb, que significa Wolfram Notebook. Los siguientes atajos del teclado pueden usarse para hacer que nuestras ecuaciones se vean presentables.

Atajo	Resultado
Ctrl + /	Barra de fracción
Ctrl + 6	Raíz cuadrada
Ctrl + 2	Exponente
Esc + pi + Esc	π
Esc + ee + Esc	e
Esc + ii + Esc	i

Tabla 1.2. Atajos del teclado para introducir símbolos en Mathematica 10.0.

Los atajos de la tabla 1.2 son útiles y servirán para escribir de forma clara la mayoría de las ecuaciones que se usarán en este trabajo. Además, se pueden introducir símbolos de suma, producto, derivadas, integrales, matrices, letras griegas, y muchos otros. Esto se hace haciendo clic sobre la ficha *Palettes*, del menú superior. Ahí se desplegarán una serie de opciones que nos permitirán seleccionar qué símbolo insertar.

En la siguiente sección se crearán unos cuantos programas simples que servirán de fundamento para crear otros más elaborados.

1.2.2. Algunos programas simples.

Imprimir una línea de texto

Uno de los programas más sencillos que hay es el de imprimir una línea de texto. Aunque es muy sencillo, esto se usará continuamente para mostrar mensajes, así como los valores de las variables que calculemos.

```
Print["¡Hola mundo!"];
¡Hola mundo!
```

Programa 1.1. Imprimir una línea de texto. En la segunda línea se muestra el resultado de correrlo.

La función Print[ ] se usa para mostrar cualquier texto en la pantalla. El programa 1.1 muestra un uso muy simple de ésta. Dentro de los corchetes se inserta el argumento de la función, que en este caso es el texto que va a mostrar. Como se trata de una cadena de texto, debe ponerse entre comillas. También es necesario destacar que en Wolfram, todas las instrucciones de un programa deben finalizar con punto y coma. Finalmente para ejecutar el programa, se pulsa Shift + Enter o la tecla Intro del teclado numérico.

También es posible añadir comentarios. Éstos son líneas de texto que el programa ignora pero nos ayudan a recordar qué es lo que hace y documentarlo. Los comentarios se insertan entre los símbolos (* *). Por ejemplo, podemos añadir un comentario al programa anterior:

```
(* Este programa es muy simple *)
Print["¡Hola mundo!"];
```

Programa 1.2. Ejemplo del uso de los comentarios.

El programa 1.2 contiene un comentario en la primera línea. Éstos no afectan su funcionamiento y son muy útiles como guía y para recordar de qué trata.

Trabajando con variables.

Las variables se utilizan para almacenar información y son indispensables en cualquier lenguaje. El programa 1.3 muestra un uso que se les puede dar:

```
x1 = 0.4; (* Fracción mol 1 *)
x2 = 1 - x1; (* Fracción mol 2 *)

Print["La fracción mol x2 vale ", x2];


La fracción mol x2 vale 0.6
```

Programa 1.3. Uso de variables para calcular una fracción mol.

En la primera línea se declara la variable x1 y se le da un valor de 0.4. Después se declara otra variable x2 asignándole el resultado de restar 1-x1. Luego se imprime el resultado de esta operación. Nótese que la variable no entra en las comillas junto con el texto que se va a imprimir; sino que debe ir afuera y separada por una coma.

Con lo que hasta el momento se ha visto, podemos hacer un programa sencillo que utilice la ecuación del gas ideal.

Problema.

Dos moles de un gas se encuentran en un recipiente de 10 litros a una temperatura de 273.15 K. Calcule la presión utilizando la ecuación del gas ideal.

```
V = 10;  (* Volumen (litros) *)
R = 0.082; (* Constante *)
n = 2; (* moles *)
T = 273.15; (* K *)

P = n R T / V;

Print["La presión vale ", P, " atm"];
```

Programa 1.4. Uso de la ecuación del gas ideal.

El programa 1.4 resuelve este problema sencillo. Primero se introducen los datos conocidos, luego la ecuación, y al final la función Print[ ] imprime el resultado.

Luego de correr este programa, la pantalla muestra lo siguiente:

```
La presión vale 4.47966 atm
```

Nota: En el lenguaje Wolfram, la multiplicación se puede expresar con un asterisco o un espacio. En este caso hay un espacio entre n, R y T.

1.2.3. Operadores relacionales, lógicos y sentencias if-else.

El uso de los operadores relacionales y lógicos hará que nuestros programas sean capaces de tomar decisiones. La tabla 1.3 muestra estos operadores y cómo se introducen en Mathematica.

Operador	Símbolo
Igual que	==
Mayor que	>
Menor que	<
Distinto de	!=
Disyunción	||
Conjunción	&&

Tabla 1.3 Operadores relacionales y lógicos.

Los operadores de la tabla 1.3 se suelen usar en conjunto con una sentencia if-else, que evalúa si una condición es cierta y lleva a cabo una acción con base en la respuesta que se obtenga. Un ejemplo del uso de una sentencia de este tipo es el programa 1.5.

```
edad = 10;

If[edad < 18,
  Print["Menor de edad"],
  Print["Mayor de edad"];
 ];
```

Programa 1.5. Ejemplo del uso de una sentencia if-else.

En la primera línea se declara la variable *edad*. Enseguida una sentencia If evalúa si esta variable es menor de 18. Si se cumple, imprime el mensaje “Menor de edad”, y en caso contrario imprime “Mayor de edad”. Es importante recalcar que toda sentencia If debe finalizar con punto y coma.

1.2.4. Ciclos For.

Los ciclos For son de mucha utilidad en análisis numérico, ya que nos permiten ordenar al programa que repita un mismo procedimiento una y otra vez. Estos ciclos se usarán en casi todo este trabajo y son de suma importancia en programación.

Problema.

Realice la suma de todos los números desde 1 hasta 100.

Para resolver esto, podríamos sumar manualmente 1 + 2 + 3 + 4 + 5 + … + 100, pero sería una tarea repetitiva y agobiante. Una segunda opción sería escribir un programa que lo haga usando un ciclo For.

```
Suma = 0;

For[i = 0, i < 101, i++,
 Suma = Suma + i;
]

Print["La suma es ", Suma];
```

Programa 1.6. Suma de todos los números desde 1 hasta 100.

Al inicio del For se declara el contador i, que inicia en el valor de cero. Después se indica hasta dónde va a llegar, que en este caso es hasta que i sea menor que 101. La parte i++ significa que el contador i aumentará de uno en uno. Luego se pone el procedimiento que se repetirá, que será ir sumando cada valor de i. El resultado que se obtiene al correr el programa es:

```
La suma es 5050
```

1.2.5. Funciones.

En la sección 1.2.2 se presentó la función Print[ ], que tiene la capacidad de mostrar un mensaje en pantalla. El lenguaje Wolfram posee alrededor de 5000 funciones incorporadas. Todas tienen nombres que inician en mayúsculas, así como sus corchetes, dentro de los cuales van los argumentos. Muchas de ellas son útiles en matemáticas, por ejemplo, la función Sin[ ] calcula el seno del argumento; la función Abs[ ] devuelve el valor absoluto.

También es posible que el usuario cree sus propias funciones. Por ejemplo, el programa 1.7 contiene una función que calcula el volumen molar de un gas con los datos de temperatura y presión:

```
(* Función que calcula el volumen molar *)
V[P_, T_] := 0.082 * T / P;

vmolar = V[2, 273];

Print["El volumen molar es", vmolar];
```

Programa 1.7. Ejemplo del uso de funciones.

Para declarar la función, se pone su nombre, seguido de corchetes. Dentro de corchetes se ponen las variables que utilizará seguidas de un guion bajo. El signo de igual debe estar precedido por el signo de dos puntos. Y a la derecha, la expresión que se va a evaluar. En el programa 1.7, la función V[ ] lleva los argumentos P y T. Después, la variable *vmolar* contendrá el resultado de evaluar esta función en 2 atm y 273 K. El resultado se muestra al final con la función Print[], que ya es predeterminada del lenguaje Wolfram.

```
El volumen molar es 11.193
```

1.3. Consideraciones adicionales.

En este capítulo se analizó qué son los métodos numéricos y la importancia que tienen. También se dio una introducción rápida al uso del software Mathematica 10.0 y el lenguaje de programación Wolfram. Se desarrollaron programas sencillos en los cuales se muestra el uso de variables, comentarios, operadores relacionales, ciclos, funciones predeterminadas de Mathematica, así como funciones creadas por el usuario, entre otras cosas. Esto servirá de base para comenzar a crear programas de análisis numérico en los capítulos siguientes.

El lenguaje de programación Wolfram es sumamente extenso y posee capacidades no sólo útiles en ingeniería, sino en ciencias de la vida, finanzas, geografía, estadística, procesamiento de imágenes, etc. En el presente capítulo se proporcionaron los fundamentos para comenzar a utilizarlo.

Si se desea profundizar en las capacidades de Wolfram, puede consultarse su documentación completa, la cual se encuentra en el siguiente sitio:

http://reference.wolfram.com/language/

Capítulo 2. Interpolación.

En ingeniería química es común encontrarse con problemas en los que se debe estimar el valor intermedio de una variable a partir de un grupo de puntos conocidos. Regularmente las fuentes de información contienen los datos en forma tabular y frecuentemente el valor buscado no se encuentra entre los datos tabulados, lo cual representa un inconveniente. Aproximar el valor mediante una línea recta es una técnica rápida y sencilla. Sin embargo, no se recomienda hacerlo porque conlleva un porcentaje de error muy elevado, pues los datos no varían de forma lineal. Si se quiere obtener un valor más exacto de la función es necesario aplicar técnicas de interpolación más exactas, como la interpolación polinomial.

Se sabe que para un conjunto discreto de puntos, hay sólo un polinomio que pasa por todos ellos. Este polinomio único puede ser determinado con los métodos de Newton y de Lagrange, los cuales se analizarán en este capítulo.

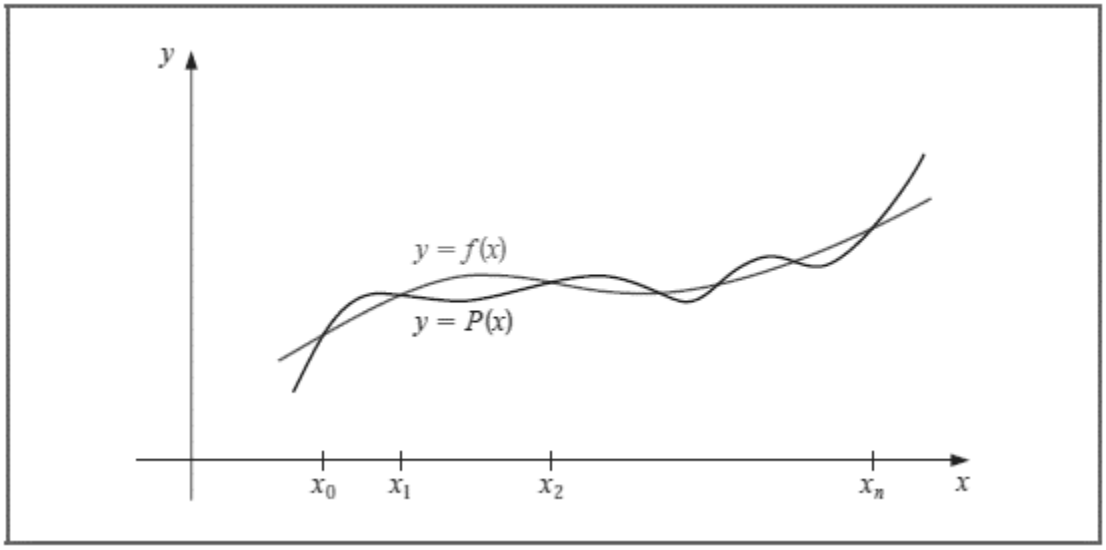


Figura 2.1. Representación de la interpolación polinomial. Dada una función f(x), de la cual se conocen sus valores en x_0, x_1, …, x_n, existe un polinomio P(x) que pasa por esos puntos

2.1. Newton.

Se tiene una función de la siguiente forma:

i	0	1	2	3	…	N
x	x_0	x_1	x_2	x_3	…	x_n
f(x)	$f(x_0)$	$f(x_1)$	$f(x_2)$	$f(x_3)$	…	$f(x_n)$

Tabla 2.1. Función en forma tabular [2].

Y se desea aproximarla con un polinomio que pasa por los puntos (0) y (1), el cual es de la forma:

$$p(x) = a_0 + a_1(x - x_0) \tag{2-1}$$

Donde x_0 es la abscisa del punto (0) y a_0, a_1 son constantes por determinar. Para encontrar el valor de a_0 se hace $x=x_0$, de donde $a = p(x_0) = f(x_0)$, y con el fin de encontrar el valor de a_1 se hace $x = x_1$, donde:

$$a_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \tag{2-2}$$

Aquí introducimos lo que se conoce como notación de diferencia dividida. La diferencia dividida cero de la función f con respecto a x_0 se denota por $f[x_0]$, que es simplemente la función f evaluada en el punto x_0.

$$f(x_0) = f[x_0] \tag{2-3}$$

La primera diferencia dividida de f con respecto a x_1 y x_0 se denota por $f[x_1,x_0]$ y está dada por la ecuación $(2\text{-}2)$.

Al sustituir los valores de las constantes en la ecuación (2-1) queda el siguiente polinomio de primer grado en términos de diferencias divididas:

$$p(x) = f[x_0] + (x - x_0) * f[x_0, x_1] \tag{2-4}$$

Y si ahora se desea aproximar la función tabular con un polinomio de segundo grado que pase por los puntos (0), (1) y (2) y que tenga la forma:

$$p_2(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) \tag{2-5}$$

Donde x_0 y x_1 vuelven a ser las abscisas de los puntos (0) y (1) y a_0, a_1 y a_2 son constantes. Para encontrar a_2 se procede como en la forma anterior; o sea:

Si $x = x_2$

$$a_2 = \frac{f(x_2) - f(x_0) - (x_2 - x_0) * \frac{f(x_1) - f(x_0)}{x_1 - x_0}}{(x_2 - x_0)(x_2 - x_1)} \tag{2-6}$$

Desarrollando algebraicamente el numerador y el denominador se llega a la siguiente expresión:

$$a_2 = \frac{\frac{f(x_2) - f(x_1)}{x_2 - x_1} \_ \frac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_2 - x_1} \tag{2-7}$$

que es la segunda diferencia dividida respecto a x_0, x_1 y x_2.

Sustituyendo las constantes a_0, a_1 y a_2 en la ecuación (3) obtenemos:

$$p_2(x) = f[x_0] + (x - x_0) * f[x_0, x_1] + (x - x_0)(x - x_1) * f[x_0, x_1, x_2] \tag{2-8}$$

que es un polinomio de segundo grado en términos de diferencias divididas.

En general, para un polinomio de grado n que pasa por los puntos (0), (1), ..., (n), y que está escrito de la forma

$$\begin{aligned} p_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots \\ + a_n(x - x_0)(x - x_1)(\ldots)(x - x_{n-1}) \end{aligned} \tag{2-9}$$

La ecuación (4) se conoce como la *fórmula de diferencia dividida interpolante de Newton*.

Los coeficientes están dados por:

$$a_0 = f(x_0) = f[x_0] \tag{2-10}$$

$$a_1 = f[x_0, x_1] \tag{2-11}$$

$$a_2 = f[x_0, x_1, x_2] \tag{2-12}$$

Por inducción, las constantes requeridas son:

$a_k = f[x_0, x_1, x_2, \dots, x_k]$

para k = 0, 1, …, n.

La determinación de diferencias divididas para cuatro puntos de datos tabulados se representa en la Tabla 2.2.

x	f(x)	Primeras diferencias divididas	Segundas diferencias divididas	Tercera diferencia dividida
x_0	$f[x_0]$	$f[x_0, x_1] = \frac{f[x_1] - f[x_0]}{x_1 - x_0}$	$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}$	$f[x_0, x_1, x_2, x_3] = \frac{f[x_1, x_2, x_3] - f[x_0, x_1, x_2]}{x_3 - x_1}$
x_1	$f[x_1]$	$f[x_1, x_2] = \frac{f[x_2] - f[x_1]}{x_2 - x_1}$	$f[x_1, x_2, x_3] = \frac{f[x_2, x_3] - f[x_1, x_2]}{x_3 - x_1}$	
x_2	$f[x_2]$	$f[x_2, x_3] = \frac{f[x_3] - f[x_2]}{x_3 - x_2}$		
x_3	$f[x_3]$			

Tabla 2.2. Cálculo de diferencias divididas.

La aproximación polinomial de Newton puede expresarse sintéticamente como [1]:

$$p_n(x) = \sum_{k=0}^{n} a_k \prod_{i=0}^{k-1} (x - x_i) \qquad (2\text{-}13)$$

La ecuación (2-13) es fácil de introducir en un lenguaje de programación como Wolfram, ya que permite la notación de producto y suma. El algoritmo en pseudocódigo [3] quedaría de la siguiente forma:

ENTRADA	Números x_0, x_1, …, x_n; valores $f(x_0)$, $f(x_1)$, …, $f(x_n)$ como $F_{0,0}$, $F_{1,0}$, $F_{n,0}$
SALIDA	Números $F_{0,0}$, $F_{1,1}$, …, $F_{n,n}$ donde:

$$p(x) = \sum_{i=0}^{n} F_{i,i} \prod_{j=0}^{i-1} (x - x_j)$$

Paso 1 Desde i = 1, 2, …, n

Desde j = 1, 2, …, i

tomar $F_{i,j} = \frac{F_{i,j-1} - F_{i-1,j-1}}{x_i - x_{i-j}}$

Paso 2 SALIDA ($F_{0,0}$, $F_{1,1}$, …, $F_{n,n}$);

Imprimir($F_{i,i}$ es f[x_0,x_1, …, x_i])

2.2. Lagrange.

Otra forma de encontrar el polinomio de interpolación es mediante la aproximación polinomial de Lagrange. Para esto, partimos nuevamente de una función tabular como la que se presenta en la tabla 2.1.

En este caso construimos primeramente, para cada k = 0,1,…,n, una función $L_k(x)$ con las dos propiedades siguientes:

a) $L_k(x_i) = 0 \;\; si \;\; i \neq k$

b) $L_k(x_k) = 1$

Para satisfacer la condición a) es necesario que el numerador contenga el término:

$$(x - x_0)(x - x_1) \dots (x - x_{k-1})(x - x_{k+1}) \dots (x - x_n)$$

Asimismo, para satisfacer la condición b), el denominador debe ser el mismo término pero evaluado en $x = x_k$. Entonces:

$$L_k(x) = \frac{(x - x_0) \dots (x - x_{k-1})(x - x_{k+1}) \dots (x - x_n)}{(x_k - x_0) \dots (x_k - x_{k-1})(x_k - x_{k+1}) \dots (x_k - x_n)} \qquad (2\text{-}14)$$

La ecuación (2-14) puede reescribirse utilizando notación de producto de la siguiente forma:

$$L_k(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{(x - x_i)}{x_k - x_i} \qquad (2\text{-}15)$$

De este modo, el polinomio de interpolación de Lagrange puede representarse como:

$$p_n(x) = \sum_{i=0}^{n} L_i(x)\, f(x_i) \qquad (2\text{-}16)$$

La ecuación (2-16) es una reformulación del polinomio de Newton que evita el cálculo de diferencias divididas.

A continuación se muestra el pseudocódigo para interpolar usando polinomios de Lagrange:

ENTRADA	Números $x_0, x_1, \ldots, x_n$; valores $f(x_0), f(x_1), \ldots, f(x_n)$
SALIDA	p(x), valor de la función en el punto x:
Paso 1	Desde i = 0, 1, …, (n + 1) $L_k(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{(x - x_i)}{x_k - x_i}$
Paso 2	$p_n(x) = \sum_{i=0}^{n} L_i(x)\, f(x_i)$
Paso 3	Imprimir[p(x)]

2.3. Problema.

A la temperatura de 10°C, la densidad ρ(c) de la solución acuosa de ácido sulfúrico varía con la concentración de acuerdo con la siguiente tabla:

C (%)	5	20	40	70
ρ (g/cm3)	1.0344	1.1453	1.3103	1.6323

Tabla 2.3. Variación de la densidad con la concentración a 10°C

Con esta información, obtenga ρ cuando la concentración es del 50%

2.3.1. Solución por Newton.

El siguiente código de Mathematica resuelve el problema:

Wolfram Mathematica | STUDENT EDITION

```
x[0] = 5;  x[1] = 20;  x[2] = 40;  x[3] = 70;
f[x[0]] = 1.0344;  f[x[1]] = 1.1453;  f[x[2]] = 1.3103;
f[x[3]] = 1.6923; n = 3; X = 50;
For[i = 0, i < n, i++,
 (* Se calculan las primeras diferencias divididas *)
 ρ[i, 1] = (f[x[i + 1]] - f[x[i]]) / (x[i + 1] - x[i]);
]
(* Segundas diferencias divididas y tercera diferencia *)
For[j = 2, j < 4, j++,
 For[i = 1, i < n, i++,
  ρ[i, j] = (ρ[i, j - 1] - ρ[i - 1, j - 1]) / (x[i + 1] - x[i - j + 1]);
 ]
]
a[0] = f[x[0]];
a[1] = ρ[0, 1];
a[2] = ρ[1, 2];
a[3] = ρ[2, 3];
ρ[X_] = Sum[a[k] Product[(X - x[i]), {i, 0, k-1}], {k, 0, n}];

Print["\nCuando C= ", X, "%, la densidad es: ", ρ[X], " g/cm^3\n"];
```

Programa 2.1. Interpolación de Newton.

El resultado que se obtiene al correr el programa es el siguiente:

```
Cuando C= 50%, la densidad es: 1.41368 g/cm^3
```

que es el valor de ρ interpolado a partir del conjunto discreto de puntos que se nos proporciona.

2.3.2. Solución por Lagrange.

El programa 2.2 resuelve el mismo problema por el método de Lagrange.

```
x[0] = 5;  x[1] = 20;  x[2] = 40;  x[3] = 70;
f[x[0]] = 1.0344;  f[x[1]] = 1.1453;  f[x[2]] = 1.3103;
f[x[3]] = 1.6923;
n = 3;  X = 50;

For[i = 0, i < (n + 1), i++,
 L[i] = ∏_{j=0}^{n} (If[j ≠ i, (X - x[j])/(x[i] - x[j]), 1]);
]
f[X_] = ∑_{i=0}^{n} L[i] f[x[i]];
Print["\nCuando C = ", X, "% la densidad es ", f[X], " g/cm^3\n"];
```

Programa 2.2. Interpolación de Lagrange.

El resultado después de correr este programa es el mismo que el obtenido por el método de Newton.

```
Cuando C = 50% la densidad es 1.41368 g/cm^3
```

2.3.3. Comparación de los métodos utilizados.

En ambos casos el resultado obtenido es 1.41368 g/cm^3. Esto se debe a que tanto Newton como Lagrange, son formas distintas de llegar al mismo polinomio de interpolación.

De modo que la tabla completa, con el nuevo valor obtenido, quedaría de esta forma:

C (%)	5	20	40	**50**	70
ρ (g/cm3)	1.0344	1.1453	1.3103	**1.41368**	1.6323

Tabla 2.4. Variación de la densidad con la concentración, con el nuevo valor interpolado.

Se puede ver que el nuevo valor obtenido por interpolación sigue la misma tendencia. Asimismo, Perry [7] reporta que la densidad en esas condiciones es de 1.4029 g/cm^3. El porcentaje de error en este caso es de 0.77%. Al ser menor al 1%,

podría considerarse que es aceptable. Sin embargo, para cálculos más exactos se puede agregar otro punto y seguir el mismo procedimiento.

Para demostrar esto, se modificó el programa 2.2 agregando un punto más, cuando la concentración es de 30%. El programa modificado quedó de la siguiente forma:

```
x[0] = 5;  x[1] = 20; x[2] = 30;  x[3] = 40;  x[4] = 70;
f[x[0]] = 1.0344;  f[x[1]] = 1.1453; f[x[2]] = 1.2255;
f[x[3]] = 1.3103;
f[x[4]] = 1.6923;
n = 4;  X = 50;

For[i = 0, i < (n + 1), i++,
 L[i] = Product_{j=0}^{n} (If[j ≠ i, (X - x[j])/(x[i] - x[j]), 1]);
]
f[X_] = Sum_{i=0}^{n} L[i] f[x[i]];
Print["\nCuando C = ", X, "% la densidad es ", f[X], " g/cm^3\n"];
```

Programa 2.3. Se agregó un nuevo punto para obtener un resultado más exacto.

Ahora se muestra el resultado obtenido al correr el programa con un punto más:

```
Cuando C = 50% la densidad es 1.40651 g/cm^3
```

El nuevo resultado es mucho más cercano al valor experimental que reporta Perry en el Manual del Ingeniero Químico [7]:

Interpolación con cuatro puntos	Interpolación con cinco puntos	Valor experimental (Perry [7])	%Error con cuatro puntos	%Error con cinco puntos
1.41368	1.40651	1.4029	0.77	0.25

Tabla 2.5. Comparación al introducir un nuevo punto.

El porcentaje de error se reduce considerablemente al introducir un nuevo punto. La ventaja de escribir un programa que funcione es que éste puede ser reutilizado o

modificado, ya sea para resolver un problema distinto o para mejorarlo agregando más puntos.

2.4. Interpolación bidimensional.

Hasta ahora se ha realizado interpolación cuando la función depende únicamente de una variable. Sin embargo, los métodos utilizados pueden extenderse para interpolar en dos dimensiones, como se muestra en la figura 2.2:

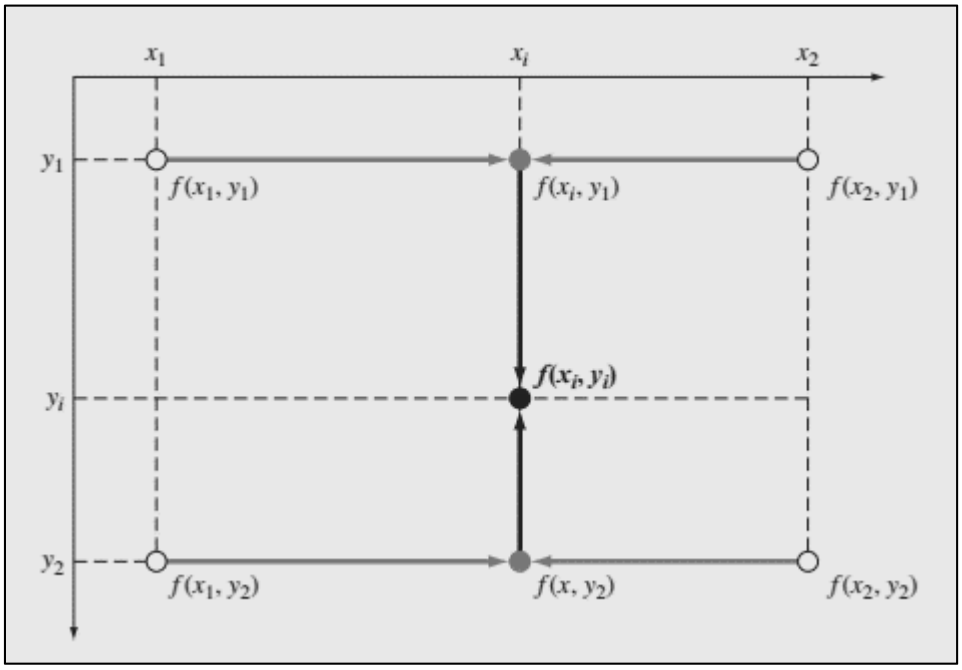


Figura 2.2. Representación de la interpolación bidimensional.

La interpolación bidimensional consiste en determinar valores para funciones de dos variables $z = f(x_i, y_i)$. Esto puede realizarse aplicando primeramente interpolación de Lagrange a lo largo del eje x para determinar valores en x_i. Posteriormente, estos valores se usan para interpolar linealmente a lo largo del eje y y conseguir el resultado final en el punto (x_i,y_i). El procedimiento se puede resumir en la siguiente tabla:

		y_1	y_2	y_3
		$L_1(y)$	$L_2(y)$	$L_3(y)$
x_1	$L_1(x)$	$f(x_1,y_1)$	$f(x_1,y_2)$	$f(x_1,y_3)$
x_2	$L_2(x)$	$f(x_2,y_1)$	$f(x_2,y_2)$	$f(x_2,y_3)$
x_3	$L_3(x)$	$f(x_3,y_1)$	$f(x_3,y_2)$	$f(x_3,y_3)$

Tabla 2.6. Interpolación bidimensional.

El pseudocódigo para realizar interpolación bidimensional se muestra a continuación:

ENTRADA Números $x_0, x_1, \ldots, x_n$; $y_0, y_1, \ldots, y_n$;
valores $f(x_0,y_0), f(x_0,y1), \ldots, f(x_0,y_n)$;
$f(x_1,y_0), f(x_1,y_1), \ldots, f(x_0,y_n)$;
$f(x_n,y_0), f(x_n,y_1), \ldots, f(x_n,y_n)$

SALIDA $p(x_i, y_i)$, valor de la función en el punto (x_i, y_i):

Paso 1 Desde i = 0, 1, …, (n - 1)

Si i ≠ j

$$L_i(x) = \prod_{j=0}^{n-1} \frac{(x - x_j)}{x_i - x_j}$$

$$L_i(y) = \prod_{j=0}^{n-1} \frac{(y - y_j)}{y_i - y_j}$$

Paso 2

$$p_n(x,y) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} [L_i(x) * L_j(y) * f(x_i, y_j)]$$

Paso 3 Imprimir[$p(x_i, y_i)$]

2.5. Problema de interpolación bidimensional.

La densidad ρ = ρ(T,C) de la solución acuosa de ácido sulfúrico varía con la temperatura y la concentración de acuerdo con la siguiente tabla:

		T (°C)		
		30	60	100
C(%)	20	1.1335	1.1153	1.0885
	40	1.2953	1.2732	1.2446
	70	1.6014	1.5753	1.5417

Tabla 2.7. Variación de la densidad con la temperatura y concentración.

Con esta información, calcule ρ en C = 50% y T= 50°C

2.5.1. Solución por Lagrange.

En el problema de la Sección 2.3, la densidad sólo dependía de la concentración. Sin embargo, ahora depende de la concentración y de la temperatura, por lo que es necesario aplicar interpolación en dos dimensiones. El programa 2.4 se escribió para solucionar este problema.

```
x[0] = 20; y[0] = 30;
x[1] = 40; y[1] = 60;
x[2] = 70; y[2] = 100;
X = 60; Y = 50; n = 3;

f[x[0], y[0]] = 1.1335;  f[x[0], y[1]] = 1.1153;  f[x[0], y[2]] = 1.0885;
f[x[1], y[0]] = 1.2953;  f[x[1], y[1]] = 1.2732;  f[x[1], y[2]] = 1.2446;
f[x[2], y[0]] = 1.6014;  f[x[2], y[1]] = 1.5753;  f[x[2], y[2]] = 1.5417;
```

Programa 2.4. Interpolación bidimensional. (Parte 1 de 2)

```
For[i = 0, i < n, i++,

  ℒx[i] = ∏_{j=0}^{n-1} (If[j ≠ i, (X - x[j])/(x[i] - x[j]), 1]);

  ℒy[i] = ∏_{j=0}^{n-1} (If[j ≠ i, (Y - y[j])/(y[i] - y[j]), 1]);

]

f[X_, Y_] = ∑_{i=0}^{n-1} (∑_{j=0}^{n-1} ℒx[i] * ℒy[j] * f[x[i], y[j]]);
Print["\nCuando C = ", X, "% y T = ", Y,
  "°C la densidad es ", f[X, Y], " g/cm^3\n"];
```

Programa 2.4. Interpolación bidimensional (Parte 2 de 2).

Luego de correr este programa, el resultado que aparece en pantalla es el siguiente:

```
Cuando C = 60% y T = 50°C la densidad es 1.47411 g/cm^3
```

La densidad calculada es 1.47411 g/cm^3. A las mismas condiciones, Perry [7] reporta que el valor experimental es de 1.4735 g/cm^3.

Densidad experimental	Densidad calculada	%Error
1.4735	1.47411	0.041

Tabla 2.8. Valor experimental vs Valor calculado.

Con el programa 2.4 fue posible calcular la densidad con un porcentaje de error de apenas 0.041%. El valor calculado es muy cercano al experimental.

Capítulo 3. Ecuaciones algebraicas no lineales.

Aunque existen muchas formas de encontrar las raíces de ecuaciones algebraicas analíticamente, es común que los problemas de ingeniería química conduzcan a ecuaciones cuya solución analítica es imposible o muy complicada. Para resolver este tipo de problemas se puede recurrir a métodos numéricos o métodos gráficos. Estos últimos suelen ser imprecisos e ineficientes, pues hay que graficar la función y enseguida ver dónde cruza el eje de las abscisas. Los métodos numéricos son una mejor opción debido a su facilidad de implementación y a la exactitud que pueden llegar a alcanzar. Además de eso, tienen la ventaja de que una vez programados, el mismo código puede reusarse para resolver cualquier problema del mismo tipo.

3.1. Métodos abiertos.

Los métodos abiertos son aquellos que no requieren que se conozca un intervalo donde la función cambie de signo. La mayoría de ellos sólo necesitan un valor inicial para comenzar a realizar los cálculos. O en caso de que necesiten dos valores, no es necesario que éstos encierren la raíz buscada.

Estos métodos suelen converger rápido y son fácilmente programables.

3.1.1. Método de punto fijo.

De los métodos abiertos, el más sencillo es el de punto fijo, a veces llamado también *método de iteraciones sucesivas*.

Un punto fijo para una función es un número en el cual el valor de la función no cambia cuando es evaluada en él.

$$g(p) = p \tag{3-1}$$

La ecuación (3-1) indica que el número p es un punto fijo de la función g.

Para encontrar la solución a alguna ecuación no lineal usando este método, primero debe tenerse la función igualada a cero, esto es:

$$f(x) = 0 \tag{3-2}$$

Enseguida se realiza alguna transformación algebraica de la función $\mathrm{f(x)}$, de modo que la variable que queremos conocer quede igualada a una expresión que dependa de ella misma:

$$x = g(x) \tag{3-3}$$

Hecho esto, el siguiente paso es evaluar la función en un estimado inicial; el resultado de esto se vuelve a introducir en la función, hasta que ya no haya cambio, que es cuando se alcanza la convergencia.

$$x_{i+1} = g(x_i) \tag{3-4}$$

El pseudocódigo para este método se muestra a continuación:

ENTRADA	Estimado inicial $\mathrm{x_0}$, función $\mathrm{x = g(x)}$, número máximo de iteraciones $\mathrm{i_{max}}$, error ε
SALIDA	Solución aproximada x
Paso 1	Desde $\mathrm{i = 0, 1, ..., (i_{max})}$
	$\mathrm{x_{i+1} = g(x_i)}$
Paso 2	Hacer:
	$\mathrm{Error = \lvert x_{i+1} - x_i \rvert}$
Paso 3	Si Error < ε
	Imprimir[La solución es $\mathrm{x_{i+1}}$]
	Detener;

3.1.2. Método de Aitken. [6]

El método de Aitken, también conocido como el *acelerador de Aitken*, es una mejora del método de punto fijo que logra que converja más rápido.

Consideremos la ecuación (3-4), la cual se usa para construir el algoritmo que genera un conjunto de aproximaciones sucesivas $\mathrm{x_1}$, $\mathrm{x_2}$,..., $\mathrm{x_n}$. Para algunas ecuaciones, esta secuencia convergerá a una raíz, mientras que en otras no lo hará.

Las condiciones de convergencia para una secuencia de este tipo se pueden obtener examinando el error de cada aproximación en la secuencia:

$$\varepsilon_i = r - x_i \tag{3-5}$$

De la ecuación (3-4) obtenemos:

$$dx_{i+1} = F'(x_i)dx_i \tag{3-6}$$

$$F'(x) = \frac{dF}{dx} \tag{3-7}$$

A medida que i crece y los errores ε_i disminuyen, estas diferenciales proporcionan una buena aproximación a los errores. Entonces:

$$\varepsilon_{i+1} \cong F'(x)\varepsilon_i \tag{3-8}$$

La ecuación (3-8) sugiere que conforme x_i se va acercando a la raíz r, los cocientes de los errores subsecuentes $\varepsilon_2/\varepsilon_1$, $\varepsilon_3/\varepsilon_2$, etc. se aproximan a una constante. Entonces:

$$\frac{\varepsilon_2}{\varepsilon_1} \cong \frac{\varepsilon_3}{\varepsilon_2} \tag{3-9}$$

Podemos reescribir la ecuación (3-9) como:

$${\varepsilon_2}^2 \cong \varepsilon_3\varepsilon_1 \tag{3-10}$$

Sustituyendo (3-5) en (3-10)

$$(r - x_2)^2 \cong (r - x_3)(r - x_1) \tag{3-11}$$

Como la solución de la ecuación (3-11) es aproximada, no se puede obtener un valor exacto de la raíz r a partir de las aproximaciones sucesivas x_1, x_2 y x_3. Entonces, la solución a la ecuación (3-11) la llamaremos x_4. Donde x_4 será ordinariamente superior a x_1, x_2 y x_3.

Al resolver la ecuación (3-11) da:

$$x_4 = \frac{(x_3x_1 - {x_2}^2)}{(x_3 - 2x_2 + x_1)} \tag{3-12}$$

Que se puede reescribir de la siguiente forma:

$$x_4 = x_3 - \frac{(x_3 - x_2)^2}{(x_3 - 2x_2 + x_1)} \qquad \textbf{(3-13)}$$

El método de Aitken consiste en usar las ecuaciones (3-4) y (3-13) de manera repetida hasta alcanzar la convergencia. Cuando $(i+1)$ no es múltiplo de 4, se usa la ecuación (3-4), pero cuando sí lo es, se utiliza la (3-13). Este procedimiento se describe en el siguiente pseudocódigo:

ENTRADA	Estimado inicial x_0, función $x = g(x)$, número máximo de iteraciones i_{max}, error ε
SALIDA	Solución aproximada x
Paso 1	Desde $i = 0, 1, ..., (i_{max})$
	Si Módulo[$(i+1) / 4$] = 0
	$$x_{i+1} = x_i - \frac{(x_i - x_{i-1})^2}{(x_i - 2x_{i-1} + x_{i-2})}$$
	Si no:
	$$x_{i+1} = g(x_i)$$
Paso 2	Hacer:
	$$\text{Error} = \lvert x_{i+1} - x_i \rvert$$
Paso 3	Si Error < ε
	Imprimir[La solución es x_{i+1}]
	Detener;

El algoritmo es muy parecido al del método de punto fijo. Sólo que en este caso usamos el operador módulo para obtener el residuo de $(i+1)/4$. Si el residuo es cero, significa que $(i+1)$ es múltiplo de 4 y debemos utilizar la ecuación del acelerador de Aitken.

3.1.3. Método de Newton-Raphson.

De los métodos de aproximaciones sucesivas, el de Newton-Raphson es de los que presentan las mejores características de eficiencia, pues casi siempre converge a la solución y lo hace en pocas iteraciones [8].

Al igual que los anteriores, este método comienza con un estimado inicial x_0, y se aproxima a la raíz luego de la aplicación de una fórmula recurrente.

Para encontrar x_{i+1}, se determina la derivada de la función en el punto x_i. La derivada proporciona la pendiente de la recta tangente en ese punto. Así que el nuevo valor x_{i+1} se encontrará en la intersección de esa tangente con el eje de las abscisas. En la figura 3.1 se ilustra este procedimiento.

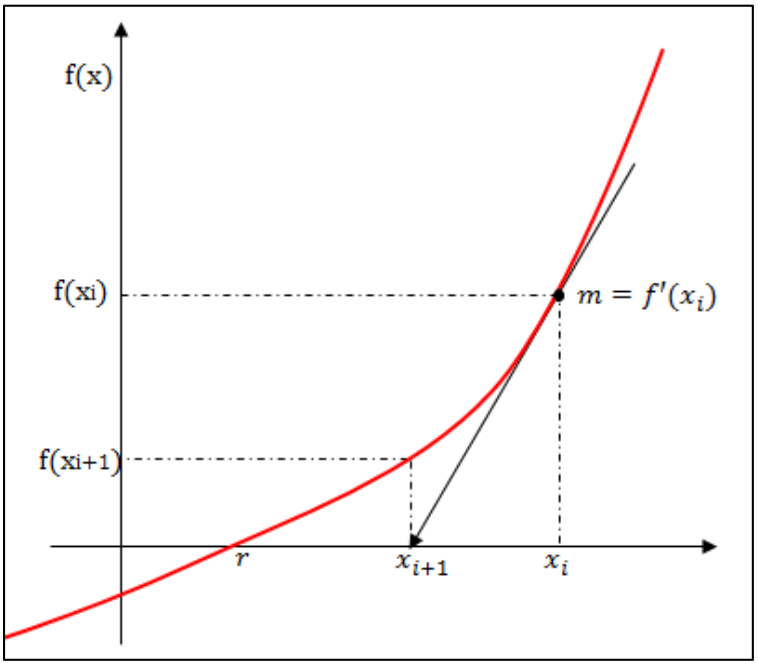


Figura 3.1. Interpretación gráfica del método de Newton-Raphson.

Supongamos que queremos aproximar la solución a $f(x)=0$ con un estimado inicial x_i. Este estimado puede estar muy lejos de la raíz, así que necesitamos encontrar una mejor aproximación. Esto lo hacemos con la ecuación de la recta tangente a la función $f(x)$ en el punto x_i:

$$y = f(x_i) + f'(x_i)(x - x_i) \tag{3-14}$$

En la figura 3.1 podemos ver que la línea tangente cruza el eje x mucho más cerca de la solución que x_i. El punto donde la tangente cruza lo llamaremos x_{i+1}, y usaremos este punto para calcular nuestra nueva aproximación a la raíz.

El punto x_{i+1} es fácil de determinar. Ya conocemos sus coordenadas, que son $(x_{i+1},0)$, y también sabemos que forma parte de la recta tangente. Así que podemos sustituirlo en la ecuación (3-14) para obtener:

$$0 = f(x_i) + f'(x_i)(x_{i+1} - x_i) \tag{3-15}$$

$$x_{i+1} - x_i = -\frac{f(x_i)}{f'(x_i)} \tag{3-16}$$

Despejando x_{i+1} nos queda:

$$\boldsymbol{x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}} \qquad \textbf{(3-17)}$$

El método de Newton-Raphson consiste en aplicar la ecuación (3-17) de manera repetida hasta lograr la convergencia. El algoritmo en pseudocódigo se muestra a continuación:

ENTRADA	Estimado inicial x_0, número máximo de iteraciones i_{max}, error ε
SALIDA	Solución aproximada x
Paso 1	Desde $i = 0, 1, ..., (i_{max})$ $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$
Paso 2	Hacer: $\text{Error} = \lvert x_{i+1} - x_i \rvert$
Paso 3	Si Error < ε Imprimir[La solución es x_{i+1}] Detener;

3.1.4. Método de la secante.

El método de Newton-Raphson es una técnica muy efectiva para la resolución de ecuaciones algebraicas no lineales. Sin embargo, un problema potencial de éste puede ser la evaluación de la derivada. El método de la secante modifica ligeramente el método de Newton-Raphson aproximando la derivada por diferencias divididas hacia atrás.

$$f'(x_i) \cong \frac{f(x_{i-1}) - f(x_i)}{x_{i-1} - x_i} \qquad (3\text{-}18)$$

Sustituyendo la ecuación (3-18) en (3-17):

$$\boldsymbol{x_{i+1} = x_i - \frac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)}} \qquad \textbf{(3-19)}$$

La ecuación (3-19) es la fórmula del método de la secante. Como puede verse, requiere de dos estimados iniciales. La interpretación gráfica de este método se presenta en la figura 3.2.

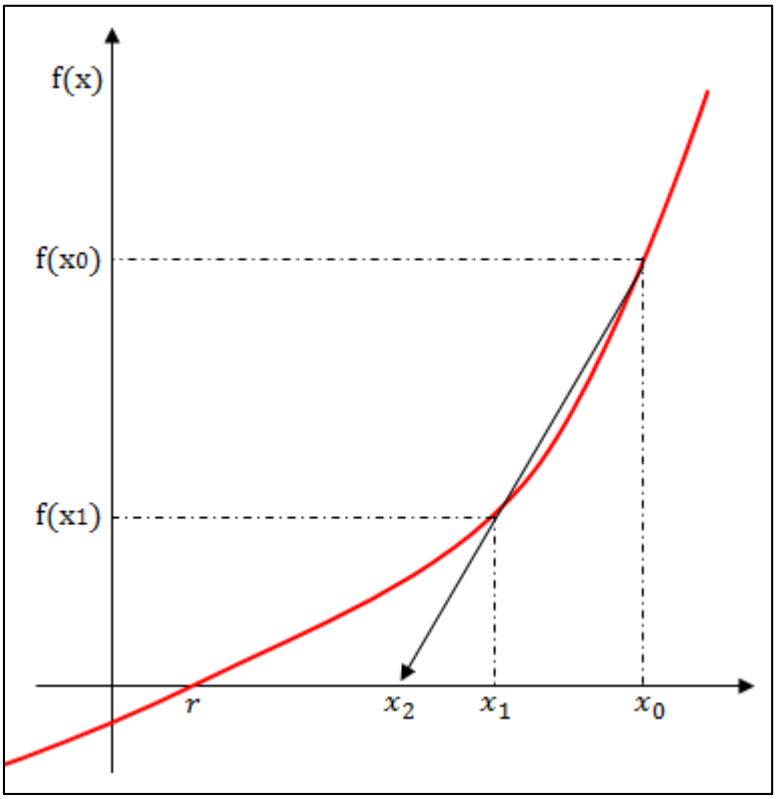


Figura 3.2. Interpretación gráfica del método de la secante.

El pseudocódigo para este método es el siguiente:

ENTRADA	Estimados iniciales x_0 y x_1, número máximo de iteraciones i_{max}, error ε
SALIDA	Solución aproximada x
Paso 1	Desde $i = 1, 2, ..., (i_{max})$ $$x_{i+1} = x_i - \frac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)}$$
Paso 2	Hacer: $$Error = \lvert x_{i+1} - x_i \rvert$$
Paso 3	Si Error < ε Imprimir[La solución es x_{i+1}] Detener;

3.1.5. Método de Wegstein. [19]

Definimos:

$$g(x) = f(x) + x \tag{3-18}$$

Como se vio en el método de punto fijo, si x es raíz, $g(x) = x$, ya que $f(x) = 0$.

El método de Wegstein es similar al de la secante. Se inicia con dos estimados x_0 y x_1, éste último se obtiene del método de punto fijo. Enseguida se traza la recta secante que pasa por los puntos $(x_{i-1}, g(x_{i-1}))$ y $(x_i, g(x_i))$. Después se debe localizar el punto de intersección de la secante con la recta de 45° que pasa por el origen, es decir, la recta $y = x$. Este punto de intersección es (x_{i+1}, x_{i+1}).

El procedimiento se ilustra en la figura 3.3.

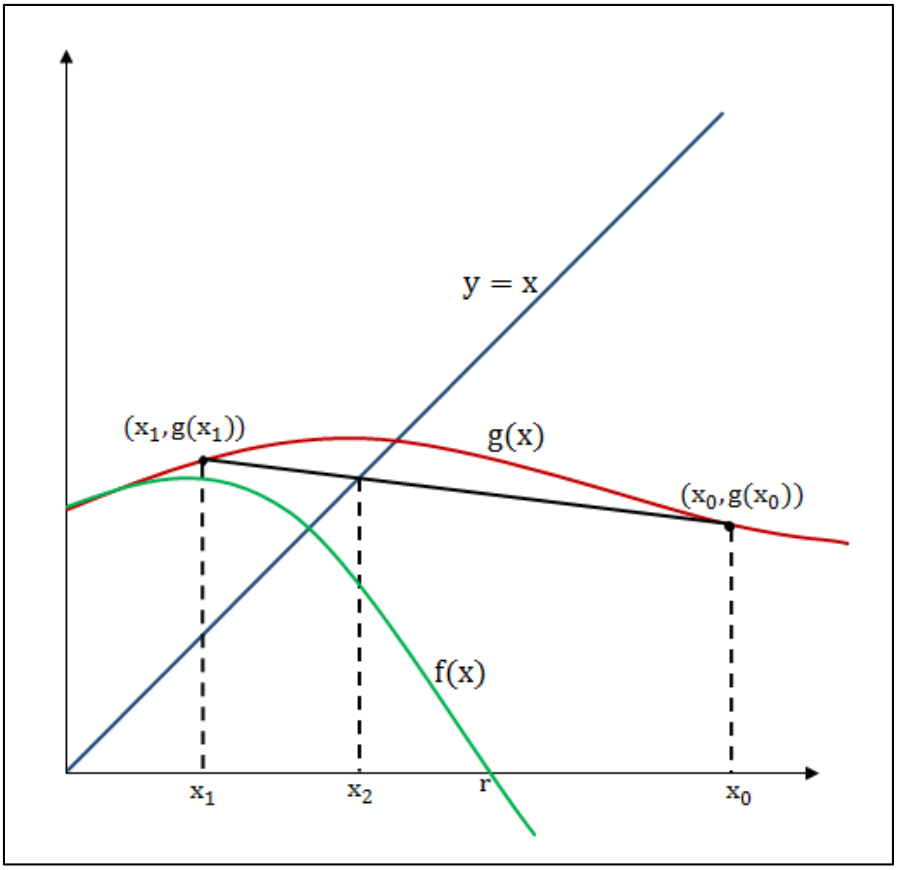


Figura 3.3. Representación gráfica del método de Wegstein.

Como tenemos tres puntos por los que pasa la recta secante, podemos expresar su pendiente de distintas formas.

$$W = tan\theta \tag{3-19}$$

$$tan\theta = \frac{g(x_i) - g(x_{i-1})}{x_i - x_{i-1}} \tag{3-20}$$

$$W = \frac{x_{i+1} - g(x_i)}{x_i - x_{i-1}} \tag{3-21}$$

Resolviendo (3-21) para x_{i+1}:

$$x_{i+1} = \frac{W}{W-1} * x_i - \frac{1}{W-1} * g(x_i) \tag{3-22}$$

Sea:

$$q = \frac{W}{W-1} \tag{3-23}$$

$$q = \frac{W - 1 + 1}{W - 1} \tag{3-24}$$

$$q = \frac{W - 1}{W - 1} + \frac{1}{W - 1} \tag{3-25}$$

$$q = 1 + \frac{1}{W - 1} \tag{3-26}$$

Entonces:

$$q - 1 = \frac{1}{W - 1} \tag{3-27}$$

Sustituyendo (3-27) en (3-22) llegamos a la ecuación de iteración para el método de Wegstein:

$$x_{i+1} = x_i * q + (1 - q) * g(x_i) \tag{3-28}$$

Donde q está dada por la ec. (3-23) $q = W/(W\text{-}1)$; y W por:

$$W = \frac{g(x_i) - g(x_{i-1})}{x_i - x_{i-1}} \tag{3-29}$$

El pseudocódigo es el siguiente:

ENTRADA	Estimados iniciales x_0 y x_1, número máximo de iteraciones i_{max}, error ε
SALIDA	Solución aproximada x
Paso 1	Desde $i = 1, 2, ..., (i_{max})$
	Hacer:
	$W = \frac{g(x_i) - g(x_{i-1})}{x_i - x_{i-1}}$
Paso 2	$q = \frac{W}{W - 1}$
Paso 3	$x_{i+1} = x_i * q + (1 - q) * g(x_i)$
Paso 4	$Error = \lvert x_{i+1} - x_i \rvert$
Paso 5	Si Error < ε
	Imprimir[La solución es x_{i+1}]; Detener;

3.2. Problema de métodos abiertos: La ecuación de Van der Waals

Los métodos anteriormente estudiados los aplicaremos a la solución del siguiente problema del libro *Álgebra superior* [5], escrito por profesores de la Facultad de Química.

Problema.

La ecuación de estado de Van der Waals es:

$$P = \frac{RT}{V-b} - \frac{a}{V^2} \tag{3-30}$$

Donde P es la presión, V el volumen molar, T la temperatura, R la constante universal de los gases, y las constantes *a* y *b* están dadas por:

$$a = \frac{9R^2Tc^2}{8Pc} \qquad\qquad b = \frac{RT_c}{3P_c}$$

Donde Pc y Tc son presión crítica y temperatura crítica, respectivamente.

Calcule el volumen molar del metanol a 97°C y presión de 1 atmósfera. Utilice el volumen ideal como estimado inicial y $\varepsilon = 10^{-6}$.

3.2.1. Solución por el método del punto fijo.

Antes de comenzar, es necesario reescribir la ecuación de Van der Waals como una función del volumen molar, quedando de la siguiente forma:

$$f(V) = \frac{ab}{V^2 - V(Pb + RT) + a}$$

A continuación se muestra el programa que se usó para solucionar este problema:

```
imax = 100;
R = 0.08205;(* Constante universal de los gases*)
T = 97;(* T *);   TK = T + 273.15; (* T en K *)
P = 1; (* P en atm *)
ϵ = 1 * 10^-6;
Tc = 512.6; (* Temperatura crítica en K *)
Pc = 79.9; (* Presión crítica en atm*)
V[0] = R * TK / P; (* Estimado inicial *)
```

Programa 3.1. Solución del problema mediante el método de punto fijo (Parte 1 de 2)

```
F[V_] := (a * b) / (V^2 - (P * b + R * TK) * V + a);

Print["i", "\t V"];
For[i = 0, i ≤ imax, i++,
  V[i + 1] = F[V[i]];
  error = Abs[V[i + 1] - V[i]];
  Print[i, "\t", V[i]];
  If[error ≤ ϵ,
   Break[];
  ];
 ];
```

Programa 3.1. Solución del problema mediante el método de punto fijo (Parte 2 de 2)

Los resultados que se obtienen al correr el programa son los siguientes:

i	V
0	30.3708
1	0.223225
2	0.240945
3	0.248241
4	0.251372
5	0.252739
6	0.253341
7	0.253607
8	0.253725
9	0.253777
10	0.2538
11	0.25381
12	0.253815
13	0.253817

El método converge en la iteración 13, dando un valor de 0.253817 L/mol.

3.2.2. Solución por el método de Aitken.

El programa 3.1 se modifica para que aplique la fórmula del acelerador de Aitken cuando (i+1) sea múltiplo de 4.

```
imax = 100;
R = 0.08205;(* Constante universal de los gases*)
T = 97;(* T*);   TK = T + 273.15; (* T en K *)
P = 1; (* P en atm *)
ϵ = 1 * 10^-6;
Tc = 512.6; (* Temperatura crítica en K *)
Pc = 79.9; (* Presión crítica en atm*)
V[0] = R * TK / P; (* Estimado inicial *)

a = 9/8 * R^2 * Tc^2 / Pc;
b = 1/3 * R * Tc / Pc;

F[V_] = a * b / (V^2 - (P * b + R * TK) * V + a);

Print["i", "\t V"];
For[i = 0, i ≤ imax, i++,
  If[Mod[i + 1, 4] == 0,
   V[i + 1] = V[i] - (V[i] - V[i - 1])^2 / (V[i] - 2 V[i - 1] + V[i - 2]);
   Print[i, "\t", V[i + 1], "\tAitken"],
   V[i + 1] = F[V[i]];
   Print[i, "\t", V[i]];
  ];
  error = Abs[V[i + 1] - V[i]];

  If[error ≤ ϵ,
   Break[];
  ];
 ];
```

Programa 3.2. Solución del problema mediante el método de Aitken.

Los resultados que se obtienen al correr este otro programa son los siguientes:

i	V	
0	30.3708	
1	0.223225	
2	0.240945	
3	0.253347	Aitken
4	0.253347	
5	0.25361	
6	0.253726	
7	0.253818	Aitken
8	0.253818	

El método de Aitken converge al valor de 0.253818, pero lo hace en menos iteraciones. Se puede ver que el acelerador fue aplicado en la iteración 3 y 7.

3.2.3. Solución por el método de Newton-Raphson.

Para poder aplicar el método de Newton-Raphson es necesario dejar la función $f(V)$ igualada a cero. Partimos de la ecuación (3-30) y multiplicamos por $(V\text{-}b)V^2$:

$$P(V-b)V^2 = RT\,V^2 - a(V-b) \tag{3-31}$$

O bien

$$PV^3 - PbV^2 = RT\,V^2 - aV - ab \tag{3-32}$$

Reordenando:

$$PV^3 - (Pb + RT)V^2 + aV - ab = 0 \tag{3-33}$$

La ecuación (3-33) se usó en el programa 3.3 para resolver el problema por el método de Newton-Raphson.

```
imax = 100;
R = 0.08205;(* Constante universal de los gases*)
T = 97;(* T °C *);   TK = T + 273.15; (* T en K *)
P = 1; (* P en atm *)
ϵ = 1 * 10^-6;
Tc = 512.6; (* Temperatura crítica en K *)
Pc = 79.9; (* Presión crítica en atm*)
V[0] = R * TK / P; (* Estimado inicial *)

a = 9/8 * R^2 * Tc^2 / Pc;
b = 1/3 * R * Tc / Pc;

F[V_] := P * V^3 - (P * b + R * TK) * V^2 + a * V - a * b;
Print["i", "\t V"];
For[i = 0, i ≤ imax, i++,
  V[i + 1] = V[i] - F[V[i]] / F'[V[i]];
  error = Abs[V[i + 1] - V[i]];
  Print[i, "\t", V[i]];
  If[error ≤ ϵ,
   Break[];
  ];
 ];
```

Programa 3.3. Solución del problema por el método de Newton-Raphson.

Y los resultados que arroja este programa son los siguientes:

i	V
0	30.3708
1	29.7407
2	29.713
3	29.713

El método de Newton-Raphson converge en menos iteraciones, aunque a un valor de 29.713 L/mol. El porqué de este cambio se explicará en la sección 3.3.

3.2.4. Solución por el método de la secante.

En el programa 3.3 se modifica para aplicar el método de la secante. Se debe sustituir la fórmula de Newton-Raphson por la ecuación (3-19) y añadir otro estimado inicial, que es el obtenido del método del punto fijo.

```
imax = 100;
R = 0.08205;(* Constante universal de los gases*)
T = 97;(* T en °C *);   TK = T + 273.15; (* T en K *)
P = 1; (* P en atm *)
ϵ = 1 * 10^-6;
Tc = 512.6; (* Temperatura crítica en K *)
Pc = 79.9; (* Presión crítica en atm*)
a = 9/8 * R^2 * Tc^2 / Pc;
b = 1/3 * R * Tc / Pc;

F[V_] := P * V^3 - (P * b + R * TK) * V^2 + a * V - a * b; (* Función *)

V[0] = R * TK / P; (* Estimado inicial *)
V[1] = F[V[0]]; (* Esto es el método del punto fijo *)

Print["i", "\t V", "\n0\t", V[0]];
For[i = 1, i ≤ imax, i++,
  V[i + 1] = V[i] - (V[i - 1] - V[i]) * F[V[i]] / (F[V[i - 1]] - F[V[i]]);
  error = Abs[V[i + 1] - V[i]];
  Print[i, "\t", V[i]];
  If[error ≤ ϵ,
   Break[];
  ];];
```

Programa 3.4. Solución del problema mediante el método de la secante.

Los resultados al aplicar el método de la secante se muestran a continuación:

i	V
0	30.3708
1	590.227
2	30.3691
3	30.3674
4	29.7404
5	29.7142
6	29.713
7	29.713

3.2.5. Solución por el método de Wegstein.

Para el método de Wegstein modificamos el programa 3.4 apegándonos al pseudocódigo de la sección 3.1.5. Dentro del ciclo definimos las variables W y q. El programa es el siguiente:

```
imax = 100;
R = 0.08205;(* Constante universal de los gases*)
T = 97; (* T °C *);   TK = T + 273.15; (* T en K *)
P = 1; (* P en atm *)
ϵ = 1 * 10^-6;
Tc = 512.6; (* Temperatura crítica en K *)
Pc = 79.9; (* Presión crítica en atm*)
a = 9/8 * R^2 * Tc^2 / Pc;
b = 1/3 * R * Tc / Pc;
F[V_] := P * V^3 - (P * b + R * TK) * V^2 + a * V - a * b; (* Función *)

V[0] = R * TK / P; (* Estimado inicial *)
V[1] = F[V[0]]; (* Esto es el método del punto fijo *)
Print["i", "\t V", "\n0\t", V[0]];
```

Programa 3.5. Solución del problema por el método de Wegstein. (Parte 1 de 2).

```
For[i = 1, i ≤ imax, i++,
  W = (F[V[i]] - F[V[i - 1]]) / (V[i] - V[i - 1]);
  q = W / (W - 1);
  V[i + 1] = V[i] * q + (1 - q) * F[V[i]];
  error = Abs[V[i + 1] - V[i]];
  Print[i, "\t", V[i]];
  If[error ≤ ϵ,
   Break[];
  ];];
```

Programa 3.5. Solución del problema mediante el método de Wegstein. (Parte 2 de 2).

A continuación se muestran los resultados de este programa:

i	V
0	30.3708
1	590.227
2	30.3692
3	30.3676
4	29.7723
5	29.7486
6	29.7476
7	29.7476

3.3. Comparación de los métodos abiertos.

Como se mencionó al inicio de este capítulo, para aplicar los métodos abiertos no es necesario conocer un intervalo en el cual se encuentre la raíz. Esto es útil en el caso del problema que se resolvió, pues al ser una ecuación cúbica existen 3 raíces y no se tiene una idea muy clara de dónde puedan localizarse.

Asimismo, se vio que el método de punto fijo y el de Aitken convergieron a una solución de 0.253818, mientras que el resto lo hicieron a 29.713 L/mol. Esto se debe a que estos otros métodos son más sensibles al estimado inicial, que en este caso fue el volumen ideal.

El significado físico de las raíces es explicable desde el punto de vista termodinámico. De acuerdo con J.M Smith, H.C. Van Ness y M.M. Abbott [9], cuando el sistema se encuentra por debajo del punto crítico, encontramos 3 raíces, pero sólo la mayor (29.713 L/mol) y la menor (0.253818 L/mol) tienen sentido. La primera de ellas es el volumen del vapor, mientras que la menor es el volumen del líquido. No obstante, la raíz mayor no es posible obtenerla con los primeros dos métodos

La tabla 3.1 muestra una comparación entre el método de punto fijo y el de Aitken.

i	Punto fijo	Aitken
0	30.3708	30.3708
1	0.223225	0.223225
2	0.240945	0.240945
3	0.248241	0.253347
4	0.251372	0.253347
5	0.252739	0.25361
6	0.253341	0.253726
7	0.253607	0.253818
8	0.253725	0.253818
9	0.253777	-
10	0.2538	-
11	0.25381	-
12	0.253815	-
13	0.253817	-

Tabla 3.1. Comparación entre el método de punto fijo y el método de Aitken. En verde se indica que el valor fue obtenido con la ecuación del acelerador de Aitken.

Al método de punto fijo le toma 13 iteraciones llegar a la raíz buscada, mientras que el método de Aitken acelera la convergencia y lo consigue en sólo 8. No obstante, los otros métodos muestran mejores características de eficiencia, pues son capaces de calcular más raíces si se cambia el estimado inicial y lo hacen en menos iteraciones.

La tabla 3.2 muestra una comparación de los métodos usados para resolver el problema.

Método	Iteraciones requeridas
Punto fijo	13
Aitken	8
Newton-Raphson	3
Secante	7
Wegstein	7

Tabla 3.2. Comparación entre los métodos de Newton-Raphson, de la secante y de Wegstein.

Los tres últimos métodos convergen en menos iteraciones que los que se analizaron anteriormente, siendo el de Newton-Raphson el que lo hace más eficientemente, pues localiza la raíz apenas en la tercera iteración.

El lenguaje de programación Wolfram, que utiliza Mathematica, permite el cálculo de la derivada con sólo proporcionar la función que se quiere derivar. Sin embargo, no en todos los lenguajes se tiene la facilidad de hacer esto. Si la derivada es complicada y nos encontramos programando en un lenguaje sin las capacidades para el cálculo numérico que tiene Wolfram, podemos optar por cualquiera de los otros métodos, ya sea el de la secante o el método de Wegstein. Estos últimos nos evitan el cálculo de la derivada. Por otro lado, si la derivada no representa ninguna dificultad o el lenguaje nos permite determinarla, Newton-Raphson puede ser lo más apropiado.

3.4. Métodos cerrados.

Los métodos numéricos cerrados se basan en el teorema del valor intermedio, o específicamente en el teorema de Bolzano. Sea f es una función continua definida en el intervalo $[a, b]$, con $f(a)$ y $f(b)$ de signo opuesto. Entonces existe al menos un número p en (a, b) con $f(p) = 0$. [3]

Este tipo de métodos requieren dos valores iniciales de x. El resultado de evaluar la función en dichos valores, debe dar signo contrario. De esta manera se garantiza que hay al menos una raíz en ese intervalo. Para localizar la raíz, es necesario dividir el intervalo en un subintervalo más pequeño. El proceso se repite y la aproximación a la raíz mejora a medida que los subintervalos se hacen más pequeños.

3.4.1. Método de la bisección.

Empezamos con los valores iniciales del intervalo (x_I, x_D), y definimos el punto medio entre ellos como x_M. Este método divide el intervalo a la mitad, es decir:

$$x_M = \frac{x_I + x_D}{2} \tag{3-34}$$

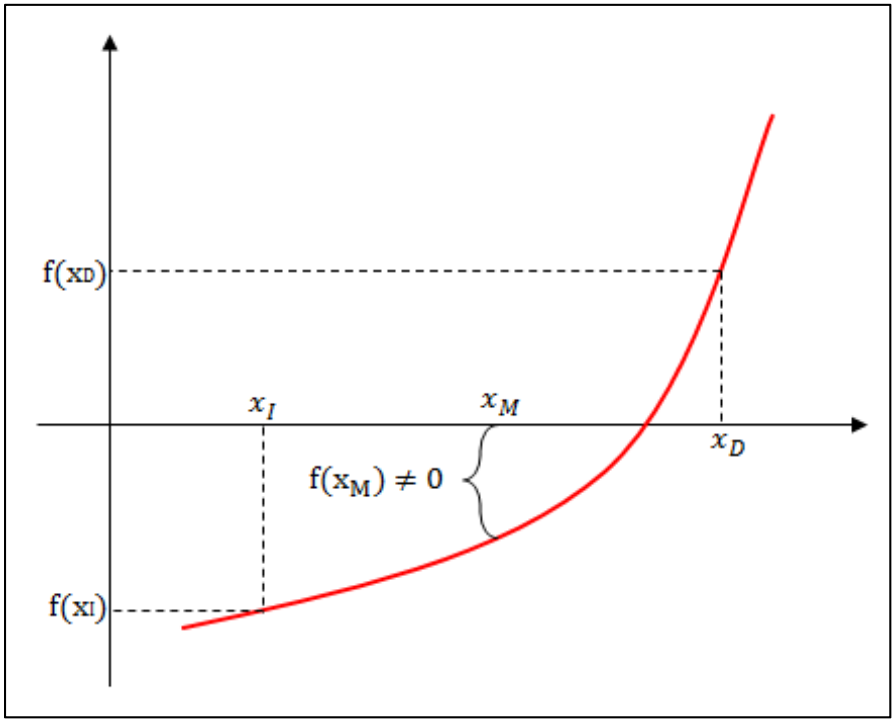


Figura 3.4. Representación gráfica del método de la bisección

Una vez aplicada la ecuación (3-35), existen tres posibilidades [1]:

1. Si $f(x_I)*f(x_D) < 0$, entonces la raíz se encuentra en el subintervalo izquierdo. Por lo tanto, hacemos $x_D = x_M$.
2. Si $f(x_I)*f(x_D) > 0$, entonces la raíz se encuentra en el subintervalo derecho. Por lo tanto, hacemos $x_I = x_M$.
3. Si $f(x_I)*f(x_D) = 0$, la raíz es x_M y se termina el cálculo.

Nota: También se puede añadir $\frac{|X_D - X_I|}{2} < \varepsilon$ como criterio para terminar el proceso.

El funcionamiento de este método se detalla en el pseudocódigo siguiente:

ENTRADA	Extremos del intervalo x_I y x_D, número máximo de iteraciones i_{max}, error ε
SALIDA	Solución aproximada x_M
Paso 1	Desde $i = 0, 1, ..., (i_{max})$
	Hacer:
	$x_M = \frac{x_I - x_D}{2}$
Paso 2	Si $f(x_M) = 0$ o $\mid (x_D - x_I) / 2 \mid < \varepsilon$
	Imprimir (x_M)
	Detener;
Paso 3	Si $f(x_I)*f(x_M) > 0$
	$x_I = x_M$;
	Si no
	$x_D = x_M$

3.4.2. Método de la posición falsa.

Este método, también conocido como *regula falsi*, funciona de manera similar al de la bisección. Sólo que en lugar de dividir el intervalo a la mitad, se hace una mejor

aproximación uniendo los puntos $(x_I, f(x_I))$ y $(x_D, f(x_D))$ con una línea recta. El punto donde esta línea se intersecta con el eje de las abscisas será el valor de x_M.

Para calcular la intersección con el eje x, se utilizan triángulos semejantes

$$\frac{f(x_I)}{x_M - x_I} = \frac{f(x_D)}{x_M - x_D} \tag{3-35}$$

Despejando x_M se llega a:

$$x_M = x_D - \frac{f(x_D)(x_I - x_D)}{f(x_I) - f(x_D)} \tag{3-36}$$

La figura 3.5 es la representación gráfica de este método.

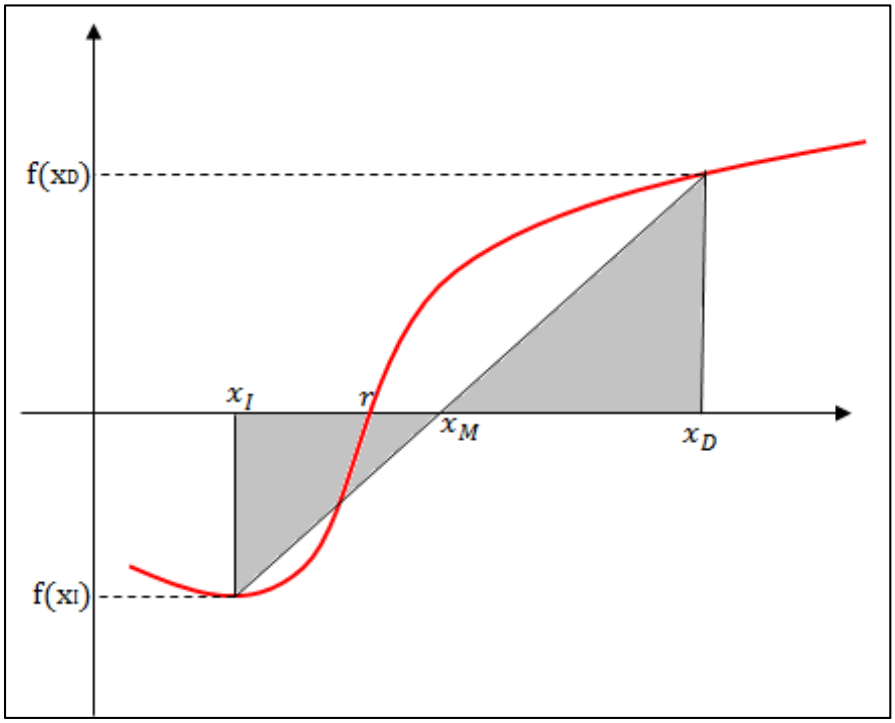


Figura 3.5. Representación gráfica del método de la posición falsa. En gris se indican los triángulos semejantes en los que se basa el método.

El pseudocódigo del método de la posición falsa es casi igual al de la bisección. Sólo se sustituye la ecuación (3-34) por la (3-36) y se cambia el criterio para terminar.

ENTRADA	Extremos del intervalo x_I y x_D, número máximo de iteraciones i_{max}, error ε
SALIDA	Solución aproximada x_M
Paso 1	Desde $i = 0, 1, ..., (i_{max})$
	Hacer: $$x_M = x_D - \frac{f(x_D)(x_I - x_D)}{f(x_I) - f(x_D)}$$
Paso 2	Si $f(x_M) < \varepsilon$
	Imprimir (x_M)
	Detener;
Paso 3	Si $f(x_I)*f(x_M) > 0$
	$x_I = x_M$;
	Si no
	$x_D = x_M$

3.5. Problema de métodos cerrados: Cálculo del factor de fricción

Para aplicar estos dos métodos, se volvió a elegir un problema del libro *Álgebra Superior* [5], de los profesores del Departamento de Matemáticas de la Facultad de Química.

Problema.

La ecuación de Colebrook puede escribirse como:

$$\frac{1}{\sqrt{f}} + 2\log\left(\frac{(k/D)}{3.7D} + \frac{2.51}{Re\sqrt{f}}\right) = 0 \qquad (3\text{-}37)$$

Usando esta ecuación, calcule el factor de fricción en una tubería de hierro galvanizado a partir de la siguiente información:

Rugosidad absoluta	k = 7.62 x 10^{-6} m
Diámetro de tubería	D = 0.0508 m
Número de Reynolds	2 200 507

3.5.1. Primera parte de la solución.

Definimos nuestra función:

$$F(f) = \frac{1}{\sqrt{f}} + 2\log\left(\frac{(k/D)}{3.7D} + \frac{2.51}{Re\sqrt{f}}\right)$$

Antes de comenzar a aplicar los métodos cerrados, es conveniente desarrollar un programa que localice el intervalo donde la función cambia de signo.

```
(* Variables *)
Unprotect[Re]; (* Para poder usar Re como variable *)
imax = 80;
k = 7.62 * 10^-6;(* Rugosidad absoluta (m)*)
d = 0.0508;(* Diámetro de tubería *)
Re = 2 200 507;(* Número de Reynolds *)
tol = 1 * 10^-7;
(* k/D = Rugosidad relativa*)
f[0] = 0.001; (* Estimado inicial para buscar intervalo *)

F[f_] := 1/Sqrt[f] + 2 * Log[10, (k / d)/(3.7 * d) + 2.51/(Re * Sqrt[f])]; (* Función *)

(* Obtener los dos estimados para el intervalo *)

For[i = 0, i < 6, i++,
 f[i + 1] = f[i] + 0.01;
 If[F[f[i + 1]] * F[f[i]] < 0,
  (* el límite inferior será el valor menor*)
  xI = Min[f[i + 1], f[i]];
  (* el límite superior será el valor mayor*)
  xD = Max[f[i + 1], f[i]];
  Print["Intervalo encontrado.
La raíz está entre f = ", xI, " y f = ", xD, "\ni\t f"];
  Break[];
 ];
];
```

Programa 3.6. Programa que localiza el intervalo en el que la función cambia de signo.

3.5.2. Solución por el método de la bisección.

Si al programa 3.6 le agregamos las siguientes líneas de código, el problema quedará resuelto por el método de la bisección.

```
(* Empieza el método de la bisección *)
For[i = 0, i < imax, i++,
  f = (xI + xD)/2;
  If[F[f] == 0 || Abs[(xD - xI)/2] < tol,
    Break[];
   ]
   If[(F[xI] * F[f]) > 0,
    xI = f,
    xD = f;
   ];
  Print[i, "\t", f];
 ];
Print["\nEl factor de fricción es ", f]
```

Programa 3.7. Método de la bisección. (Utilizar en conjunto con el programa 3.6).

Los resultados obtenidos con este método se muestran a continuación:

```
Intervalo encontrado.
La raíz está entre f = 0.021 y f = 0.031
i      f
0      0.026
1      0.0285
2      0.02725
3      0.026625
4      0.0263125
5      0.0261563
6      0.0260781
7      0.0261172
```

(Continúa)

```
8     0.0260977
9     0.0261074
10     0.0261123
11     0.0261147
12     0.0261135
13     0.0261129
14     0.0261132
15     0.0261131
```

Este método convergió en la iteración 15 al valor de 0.0261131.

3.5.3. Solución por el método de la posición falsa.

De manera similar, le agregamos las siguientes líneas de código al programa 3.6 y el problema estará resuelto por el método de la posición falsa.

```
(* Empieza el método de la posición falsa *)
For[i = 0, i < imax, i++,

  f = xD - (xD - xI) * F[xD] / (F[xD] - F[xI]);


  If[Abs[F[f]] < tol,
    Break[];
   ]
   If[(F[xI] * F[f]) > 0,
    xI = f,
    xD = f;
   ];
  Print[i, "\t", f];
 ];
Print["\nEl factor de fricción es ", f]
```

Programa 3.8. Método de la posición falsa. (Utilizar en conjunto con el programa 3.6).

Los resultados obtenidos con este método son los siguientes:

```
Intervalo encontrado.
La raíz está entre f = 0.021 y f = 0.031
i      f
0     0.026834
1     0.0262206
2     0.0261291
3     0.0261154
4     0.0261134
5     0.0261131
6     0.026113
7     0.026113

El factor de fricción es 0.026113
```

Este método convergió al valor de 0.026113 en la iteración 7.

3.6. Comparación de los métodos cerrados.

Los métodos numéricos cerrados son una buena opción cuando tenemos una idea de dónde puede estar la raíz. En esta ocasión se resolvió el problema del factor de fricción. Como sabemos que f es un número adimensional que sólo puede estar entre 0 y 1, comenzamos a buscar el intervalo evaluando la función en un número ligeramente mayor a 0, que fue 0.001 (véase programa 3.6). Este tipo de problemas son muy apropiados para resolverse con métodos cerrados.

Tanto el método de la bisección como el de la posición falsa funcionan de manera muy parecida. Sin embargo, este último es el que muestra la mayor eficiencia, ya que logra converger en menos iteraciones. Esto sucede porque el método de la bisección no toma en cuenta la magnitud de $f(x_I)$ y de $f(x_D)$ para determinar el x_M. Simplemente divide el intervalo a la mitad en cada iteración, mientras que el de la *regula falsi* traza una línea recta, generando una "posición falsa" de la raíz en la intersección con el eje de las abscisas.

La tabla 3.3 muestra una comparación de ambos métodos.

i	Bisección	Regula falsi
0	0.026	0.026834
1	0.0285	0.0262206
2	0.02725	0.0261291
3	0.026625	0.0261154
4	0.0263125	0.0261134
5	0.0261563	0.0261131
6	0.0260781	0.026113
7	0.0261172	0.026113
8	0.0260977	--
9	0.0261074	--
10	0.0261123	--
11	0.0261147	--
12	0.0261135	--
13	0.0261129	--
14	0.0261132	--
15	0.0261131	--

Tabla 3.3. Comparación entre el método de la bisección y el de la posición falsa.

Con ambos métodos se llegó al mismo valor, aunque el de la posición falsa fue más eficiente. Según el autor Chapra [1], hay casos especiales en los que puede suceder lo contrario. Por eso es recomendable evaluar la función en el valor calculado y verificar que el resultado sea cercano a cero. En este caso:

$$F(0.026113) = -2.13099\ x\ 10^{-8}$$

La función arroja un valor muy cercano a cero, por lo cual es aceptable la solución. Esto último se incluyó como criterio de paro en el método de la posición falsa para asegurar que el resultado fuera correcto.

Capítulo 4. Sistemas de ecuaciones algebraicas lineales.

El álgebra de matrices, también conocida como álgebra lineal, es una rama de las matemáticas ampliamente aplicada en la solución de problemas de ingeniería. En muchas ocasiones, los problemas de ingeniería química tienen que ser modelados como un sistema de ecuaciones algebraicas lineales.

Considérese el siguiente sistema de n ecuaciones con n incógnitas x_1, x_2, ..., x_n.

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{aligned}$$

De los subíndices en los coeficientes de este sistema, a_{ij}, la i corresponde a la ecuación en la que está el coeficiente, y la j corresponde a la incógnita por la que está multiplicado. Esto puede ser expresado en forma simple como:

$$ax = b \tag{4-1}$$

Para resolver este sistema usando el álgebra lineal, primero se escribe la matriz aumentada, la cual es un arreglo rectangular de todos los coeficientes expresado de la siguiente forma:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{pmatrix} \tag{4-2}$$

A la diagonal que contiene los elementos a_{11}, a_{22}, a_{33}, ..., a_{nn}, se le llama diagonal principal de la matriz. Los métodos que se estudiarán en este capítulo requieren que no haya ceros en dicha diagonal, por lo que se pueden intercambiar las filas si así se requiere.

4.1. Eliminación gaussiana.

Este método consta de dos pasos fundamentales.

a) La eliminación hacia adelante de incógnitas.
b) La sustitución hacia atrás.

La eliminación hacia adelante consiste en ir eliminando incógnitas hasta conseguir una matriz triangular superior, que es aquel sistema que sólo tiene ceros debajo de la diagonal principal.

Considérese la siguiente matriz:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{22} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{pmatrix} \tag{4-2}$$

Con el fin de eliminar la incógnita $x1$ de la segunda fila, se multiplica la primera fila por a_{21}/a_{11} y este resultado se le resta a esa misma fila. El proceso se repite para eliminar x_1 de la tercera fila (ahora multiplicando por a_{31}/a_{11}) dando como resultado lo siguiente:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ 0 & a'_{22} & a'_{22} & b'_2 \\ 0 & a'_{32} & a'_{33} & b'_3 \end{pmatrix} \tag{4-3}$$

El superíndice prima, indica que los coeficientes ya fueron modificados una vez. Lo siguiente será eliminar la incógnita x_2 del tercer renglón. Esto se logra multiplicando la primera fila por a'_{32}/a_{12} y restando el resultado a la tercera. El resultado sería el siguiente:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ 0 & a'_{22} & a'_{22} & b'_2 \\ 0 & 0 & a''_{33} & b''_3 \end{pmatrix} \tag{4-4}$$

La matriz (4-4) ya es una matriz triangular superior, pues sólo tiene ceros debajo de la diagonal principal. Ahora es posible aplicar la sustitución hacia atrás para terminar de resolver el sistema.

De la última fila, se despeja x_3:

$$x_3 = \frac{b_3''}{a_{33}''} \tag{4-5}$$

Enseguida se sustituye x_3 en la segunda fila y se despeja x_2 para dar:

$$x_2 = \frac{b_2' - a_{23}' x_3}{a_{22}'} \tag{4-6}$$

Al sustituir el resultado de (4-6) en la primera fila despejar x_1 obtenemos:

$$x_3 = \frac{b_1 - a_{12} x_2 - a_{13} x_3}{a_{11}} \tag{4-7}$$

En general, este procedimiento se puede representar mediante la fórmula:

$$x_i = [b_i^{(i-1)} - \sum_{j=i+1}^{n} a_{ij}^{(i+1)} x_j] \,/\, a_{ii}^{i-1} \tag{4-8}$$

Para $i = (n\text{-}1), (n\text{-}2), \ldots, 1$.

El pseudocódigo es el siguiente:

ENTRADA	Matriz aumentada
SALIDA	Soluciones $x_1, x_2, \ldots, x_n$
Paso 1	Desde $k = 1, (n - 1)$
Paso 2	Desde $i = (k+1), n$
	$factor = a_{i,k} / a_{k,k}$
Paso 3	Desde $j = (k+1), n$
	$a_{i,j} = a_{i,j} - factor * a_{k,j}$
	Fin Desde (3)
	$b_i = b_i - factor * b_k$
	Fin Desde (2)
	Fin Desde (1)
Paso 4	$x_n = b_n / a_{n,n}$
Paso 5	Desde $i = (n - 1), 1, -1$
	$sum = b_i$
	Desde $j = (i + 1), n$

```
                    sum = sum  - ai,j * xj
                Fin Desde (5)
                xi = sum / ai,i
            Fin Desde (4)
```

4.2. Método de Gauss-Jordan.

Este método es muy parecido al de eliminación gaussiana. La diferencia está en que en el método de Gauss-Jordan se busca obtener una matriz identidad.

Considérese nuevamente la matriz diagonal superior (4-4) obtenida mediante eliminación gaussiana. Para convertirla en una matriz identidad podemos dividir el tercer renglón entre a''_{33}. Enseguida eliminamos x_3 del renglón 1 restándole el renglón 3 multiplicado por a_{13}.

$$\begin{pmatrix} a_{11} & a_{12} & 0 & b_1 \\ 0 & a'_{22} & a'_{22} & b'_2 \\ 0 & 0 & 1 & \frac{b''_3}{a''_{33}} \end{pmatrix} \qquad (4\text{-}9)$$

El proceso continúa hasta que se logra obtener la matriz identidad.

$$\begin{pmatrix} 1 & 0 & 0 & b_1^n \\ 0 & 1 & 0 & b_2^n \\ 0 & 0 & 1 & b_3^n \end{pmatrix} \qquad (4\text{-}10)$$

Es posible que haya problemas si algún elemento de la diagonal principal es cero. En este capítulo se escribió un programa que consta de tres subrutinas para el método de Gauss-Jordan y eliminación gaussiana. La primera de ellas intercambia filas si uno de los elementos pivote es cero, la segunda resta filas y la tercera normaliza.

El programa se muestra en la siguiente página.

```
(* Subrutina 1: Intercambio de filas *)
inter[ma_, l_] :=
 Module[{},
  j = l; n = Length[ma];
  While[ma[[j, l]] == 0 ∧ j ≤ n, j = j + 1];
  exchange =
   Table[If[i == k ∧ i ≠ j ∧ i ≠ l, 1,
     If[(i == j ∧ k == l) ∨ (i == l ∧ k == j), 1, 0]],
    {i, 1, n}, {k, 1, n}];
  mat = exchange.ma]

(* Subrutina 2: Resta de filas *)
resta[ma_, l_, p_, fac_] :=
 Module[{},
  n = Length[ma];
  res = Table[If[i == p ∧ k == l, 1, 0],
    {i, 1, n}, {k, 1, n}];
  mat = ma - fac res.ma]

(* Subrutina 3: Normalización *)
normal[ma_, l_] :=
 Module[{},
  j = l; n = Length[ma];
  exc = Table[If[i ≠ k, 0, If[i == l, 1/ma[[l, l]], 1]],
    {i, 1, n}, {k, 1, n}];
  mat = exc.ma]
```

Programa 4.1. Subrutinas para el método de Gauss-Jordan y eliminación gaussiana. La primera intercambia filas, la segunda resta filas y la tercera normaliza.

En la siguiente sección se utilizará el programa 4.1 en conjunto con otros para resolver problemas comunes.

4.3. Problemas de sistemas de ecuaciones algebraicas lineales.

4.3.1. Balances de materia en estado estacionario.

Uno de los principios científicos más importantes en la ingeniería química es la ley de la conservación de la materia. Para mostrar una aplicación de los métodos estudiados en esta sección, se eligió el problema 2.4 del libro Resolución de problemas en Ingeniería Química y Bioquímica con PolyMath, Excel y MatLab, de Cutlip y Shacham [11].

Problema.

Una mezcla de xileno, estireno, tolueno y benceno se separa con un sistema de columnas de destilación según la disposición de la figura 4.1.

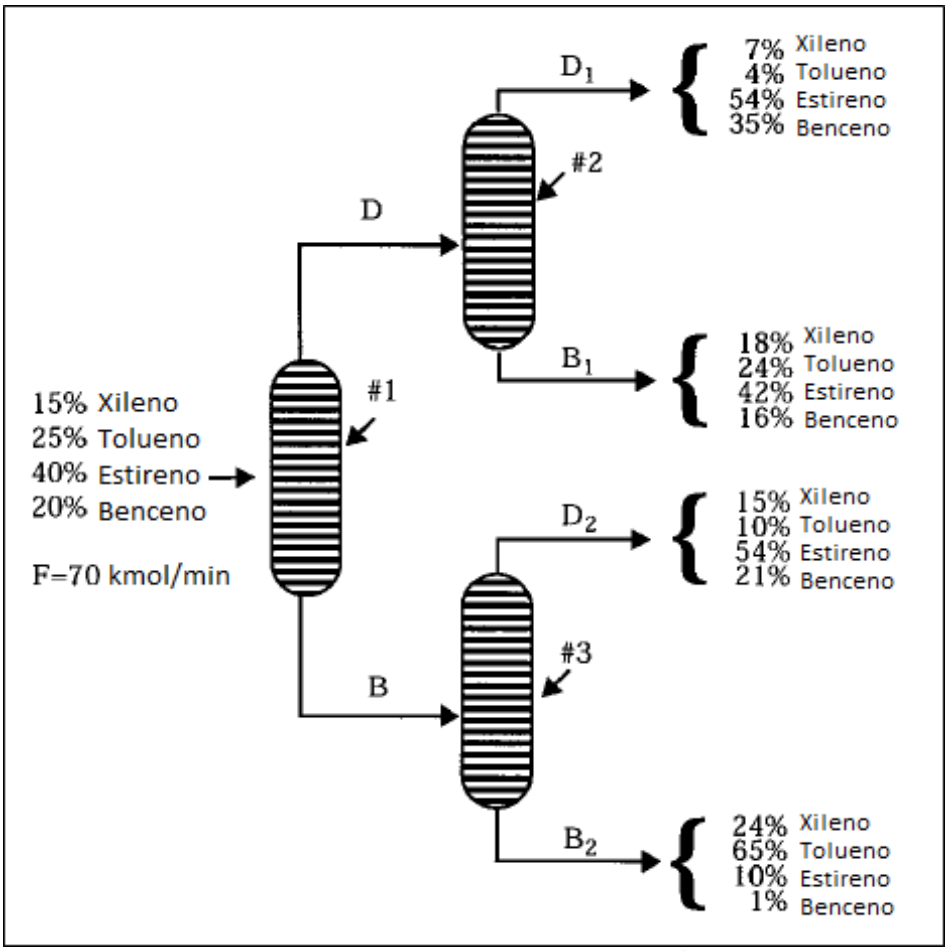


Figura 4.1. Secuencia de separación

Calcule las velocidades de flujo molar de D_1, D_2, B_1 y B_2.

4.3.1.1. Solución por eliminación gaussiana.

Los balances de materia para cada especie son:

Xileno	$0.07D_1 + 0.18B_1 + 0.15D_2 + 0.24B_2 = 0.15(70)$
Estireno	$0.04D_1 + 0.24B_1 + 0.10D_2 + 0.65B_2 = 0.25(70)$
Tolueno	$0.54D_1 + 0.42B_1 + 0.54D_2 + 0.10B_2 = 0.40(70)$
Benceno	$0.35D_1 + 0.16B_1 + 0.21D_2 + 0.01B_2 = 0.20(70)$

El programa 4.2 resuelve un sistema por eliminación gaussiana. Es importante usarlo en conjunto con el programa 4.1 que se escribió en la sección anterior, de lo contrario no funcionará.

```
gauss[matr_] :=
 Module[{},
  ma = matr;
  n = Length[ma];
  For[m = 1, m < n, m++,
   inter[ma, m];
   ma = mat;
   Print[MatrixForm[ma]];
   For[s = m + 1, s ≤ n, s++,
    resta[ma, m, s, ma[[s, m]] / ma[[m, m]]];
    ma = mat]];
  Print[MatrixForm[ma]];
  x[n] = ma[[n, n + 1]] / ma[[n, n]];
  Print["x[", n, "]=", x[n]];
  For[j = n - 1, j ≥ 1, j--,
   x[j] = (ma[[j, n + 1]] - Sum[ma[[j, h]] x[h],
         {h, j + 1, n}]) / ma[[j, j]];
   Print["x[", j, "]=", x[j]]]
 ]
```

Programa 4.2. Solución del problema por eliminación gaussiana. Usar junto con el programa 4.1. (Parte 1 de 2).

```
F = 70;
gauss[( 0.07  0.18  0.15  0.24  0.15*F
        0.04  0.24  0.10  0.65  0.25*F
        0.54  0.42  0.54  0.10  0.40*F
        0.35  0.16  0.21  0.01  0.20*F )];
```

Programa 4.2. Solución del problema por eliminación gaussiana. Usar junto con el programa 4.1. (Parte 2 de 2).

Y los resultados en pantalla son los siguientes:

```
( 0.07  0.18  0.15  0.24  10.5
  0.04  0.24   0.1  0.65  17.5
  0.54  0.42  0.54   0.1   28.
  0.35  0.16  0.21  0.01   14. )

( 0.07    0.18       0.15       0.24     10.5
  0.    0.137143  0.0142857  0.512857    11.5
  0.   -0.968571 -0.617143 -1.75143    -53.
  0.     -0.74      -0.54     -1.19     -38.5 )

( 0.07    0.18       0.15       0.24     10.5
  0.    0.137143  0.0142857  0.512857    11.5
  0.       0.      -0.51625   1.87063   28.2188
  0.       0.     -0.462917  1.57729   23.5521 )

( 0.07    0.18       0.15       0.24      10.5
  0.    0.137143  0.0142857  0.512857     11.5
  0.       0.      -0.51625   1.87063    28.2188
  0.       0.         0.     -0.100081  -1.75141 )

x[4]=17.5

x[3]=8.75

x[2]=17.5

x[1]=26.25
```

$D_1 = 26.25\ kmol/min$ $B_1 = 17.5\ kmol/min$

$D_2 = 8.75\ kmol/min$ $B_2 = 17.5\ kmol/min$

El programa muestra la matriz desde el principio hasta convertir nuestro sistema de ecuaciones en una matriz diagonal superior. Después aplica la sustitución hacia atrás y calcula el valor de cada incógnita.

4.3.1.2. Solución por el método de Gauss-Jordan.

Para resolver el problema por Gauss-Jordan, se modificó el programa 4.2 para convertir en ceros los elementos por arriba de la diagonal, así como normalizar dividiendo entre los elementos pivote.

```
GaussJordan[matr_] :=
 Module[{},
  ma = matr;
  n = Length[ma];
  For[m = 1, m < n, m++,
   inter[ma, m];
   ma = mat;
   normal[ma, m];
   ma = mat;
   Print[MatrixForm[ma], "\n"];
   For[s = m + 1, s ≤ n, s++,
    resta[ma, m, s, ma[[s, m]]/ma[[m, m]]];
    ma = mat]];
  normal[ma, n];
  ma = mat;
  For[m = n, m > 1, m--,
   For[s = m - 1, s ≥ 1, s--,
    resta[ma, m, s, ma[[s, m]]];
    ma = mat]];
  Print[MatrixForm[ma]]]

F = 70;

GaussJordan[{{0.07, 0.18, 0.15, 0.24, 0.15*F},
             {0.04, 0.24, 0.10, 0.65, 0.25*F},
             {0.54, 0.42, 0.54, 0.10, 0.40*F},
             {0.35, 0.16, 0.21, 0.01, 0.20*F}}];
```

Programa 4.3. Solución del problema por el método de Gauss-Jordan. Es necesario utilizarlo en conjunto con el programa 4.1.

Estos son los resultados de correr el programa 4.3:

```
⎛  1.   2.57143  2.14286  3.42857  150.  ⎞
⎜0.04    0.24     0.1      0.65    17.5 ⎟
⎜0.54    0.42     0.54      0.1    28.  ⎟
⎝0.35    0.16     0.21     0.01    14.  ⎠

⎛1.   2.57143    2.14286    3.42857    150.    ⎞
⎜0.      1.      0.104167   3.73958   83.8542  ⎟
⎜0.  -0.968571  -0.617143  -1.75143    -53.    ⎟
⎝0.    -0.74      -0.54      -1.19     -38.5   ⎠

⎛1. 2.57143   2.14286    3.42857    150.    ⎞
⎜0.    1.     0.104167   3.73958   83.8542  ⎟
⎜0.    0.        1.     -3.62349  -54.661   ⎟
⎝0.    0.    -0.462917   1.57729   23.5521  ⎠

⎛1. 0. 0. 0. 26.25⎞
⎜0. 1. 0. 0.  17.5⎟
⎜0. 0. 1. 0.  8.75⎟
⎝0. 0. 0. 1.  17.5⎠
```

4.3.2. Balanceo de ecuaciones químicas.

Otra aplicación típica de los sistemas de ecuaciones algebraicas lineales es el balanceo de ecuaciones químicas [5]. Consideremos la siguiente reacción:

$$x_1 HCl + x_2 KMnO_4 \rightarrow x_3 Cl_2 + x_4 KCl + x_5 MnCl_2 + x_6 H_2O$$

Para balancearla es necesario encontrar el valor de los coeficientes estequiométricos.

4.3.2.1. Solución por el método de Gauss-Jordan.

Realizamos el balance atómico:

	HCl	$KMnO_4$	Cl_2	KCl	$MnCl_2$	H_2O	
H	x_1					$-2x_6$	= 0
Cl	x_1		$-2x_3$	$-x_4$	$-2x_5$		= 0
K		x_2		$-x_4$	$-x_5$		= 0
Mn		x_2					= 0
O		$4x_2$				$- x_6$	= 0

Tabla 4.1. Balance atómico.

La matriz aumentada para este sistema es:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & -2 & 0 \\ 1 & 0 & -2 & -1 & -2 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & -1 & 0 \end{pmatrix}$$

Una de las grandes ventajas de programar es que el código que se escribe puede ser reutilizado para resolver cualquier problema. Esta vez, volveremos a usar el programa 4.3 para balancear esta ecuación. Sólo es necesario borrar la matriz del problema anterior, que se encuentra en la última línea, y sustituirla por la de este otro. La modificación quedaría así:

```
GaussJordan[{{1, 0, 0, 0, 0, -2, 0},
             {1, 0, -2, -1, -2, 0, 0},
             {0, 1, 0, -1, 0, 0, 0},
             {0, 1, 0, 0, -1, 0, 0},
             {0, 4, 0, 0, 0, -1, 0}}];
```

Y los resultados de correr el mismo programa 4.3, pero con los datos de este problema, son los siguientes:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & -2 & 0 \\ 1 & 0 & -2 & -1 & -2 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & -1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & -2 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -2 & -1 & -2 & 2 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & -1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & -2 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & \frac{1}{2} & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & -1 & 0 \end{pmatrix}$$

(Continúa)

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & -2 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & \frac{1}{2} & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & -1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & -2 & 0 \\ 0 & 1 & 0 & 0 & 0 & -\frac{1}{4} & 0 \\ 0 & 0 & 1 & 0 & 0 & -\frac{5}{8} & 0 \\ 0 & 0 & 0 & 1 & 0 & -\frac{1}{4} & 0 \\ 0 & 0 & 0 & 0 & 1 & -\frac{1}{4} & 0 \end{pmatrix}$$

La última matriz ya es el resultado luego de aplicar el método de Gauss-Jordan. En este caso tenemos un grado de libertad. Entonces:

$$x_1 = 2x_6 \qquad x_4 = \frac{1}{4}x_6$$

$$x_2 = \frac{1}{4}x_6 \qquad x_5 = \frac{1}{4}x_6$$

$$x_3 = \frac{5}{8}x_6$$

Asignamos a x_6 el valor de 8 para tener sólo números enteros.

$$16HCl + 2KMnO_4 \rightarrow 5Cl_2 + 2KCl + 2MnCl_2 + 8H_2O$$

De este modo la reacción queda balanceada.

Capítulo 5. Sistemas de ecuaciones algebraicas no lineales.

En el capítulo anterior se presentaron métodos para resolver sistemas de ecuaciones algebraicas lineales, los cuales son útiles en balances de materia, en el balanceo de ecuaciones químicas, entre muchas otras aplicaciones.

Hay ocasiones en que los problemas conducen a sistemas de ecuaciones no lineales, en los cuales no es posible aplicar eliminación gaussiana ni Gauss-Jordan. Los métodos que se analizarán en este capítulo nos permitirán aproximar la solución de dichos sistemas utilizando técnicas similares a las estudiadas en la sección 3.1.

5.1. Método de Newton-Raphson multivariable.

El método de Newton-Raphson puede generalizarse a un caso de múltiples variables para resolver n ecuaciones algebraicas.

$$\begin{aligned} f_1(x_1,..,x_n) &= \boldsymbol{f_1}(\boldsymbol{x}) = 0 \\ &\dots \\ f_n(x_1,..,x_n) &= \boldsymbol{f_n}(\boldsymbol{x}) = 0 \end{aligned} \tag{5-1}$$

$$\boldsymbol{x} = [x_1,..,x_n]^T \tag{5-2}$$

Donde x es un vector de n dimensiones. El sistema (5-1) puede representarse en forma en forma vectorial como:

$$\boldsymbol{f}(\boldsymbol{x}) = 0 \tag{5-3}$$

Asimismo, el jacobiano de la función (5-3) es:

$$\boldsymbol{J_f}(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_n} \\ \dots & \dots & \dots \\ \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_n}{\partial x_n} \end{pmatrix} \tag{5-4}$$

Para obtener la fórmula de Newton-Raphson considérese la serie de Taylor:

$$f_i(\boldsymbol{x} + \delta\boldsymbol{x}) = f_i(\boldsymbol{x}) + \sum_j \frac{\partial f_i}{\partial x_j}\delta x_j + O(\delta\boldsymbol{x}^2) \qquad (5\text{-}5)$$

$$(i = 1, \dots, n)$$

Ignoramos los términos de $\delta\mathbf{x}^2$ y superiores, y tomamos $\mathrm{fi}(\mathbf{x}+\delta\mathbf{x})$ como cero (este término es la intersección de la tangente) y obtenemos:

$$\sum_j \frac{\partial f_i}{\partial x_j}\delta x_j = -f_i(\boldsymbol{x}) \qquad (5\text{-}6)$$

$$(i = 1, \dots, n)$$

Resolviendo el sistema lineal para $\delta\mathbf{x}^2$ queda:

$$\delta\boldsymbol{x} = -\boldsymbol{J}_f^{-1}(\boldsymbol{x})\boldsymbol{f}(\boldsymbol{x}) \qquad (5\text{-}8)$$

La raíz se puede encontrar a partir de cualquier punto de inicio como:

$$\boldsymbol{x} + \delta\boldsymbol{x} = \boldsymbol{x} - \boldsymbol{J}_f^{-1}(\boldsymbol{x})\boldsymbol{f}(\boldsymbol{x}) \qquad (5\text{-}9)$$

Si las ecuaciones son no lineales, este resultado es sólo una aproximación de la raíz, el cual se puede mejorar iterativamente:

$$\boldsymbol{x_{n+1}} = \boldsymbol{x_n} - \boldsymbol{J}_f^{-1}(\boldsymbol{x_n})\boldsymbol{f}(\boldsymbol{x_n}) \qquad (5\text{-}10)$$

La ecuación (5-10) es la ecuación del método de Newton-Raphson multivariable. Los valores están dados como vectores.

Caso de dos variables.

Para este caso, podemos simplificar tomando el primer elemento del vector solución como h y el segundo solución como j. Así, podemos cambiar la notación como:

$$x^{k+1} = x^k + h \qquad (5\text{-}11)$$

$$y^{k+1} = y^k + j \qquad (5\text{-}12)$$

El pseudocódigo para este caso se muestra a continuación:

ENTRADA	Estimados iniciales x_0, y_0 funciones $F1(x, y) = 0$ y $F2(x, y) = 0$, número máximo de iteraciones i_{max}, error ε
SALIDA	Solución aproximada x, y.
Paso 1	Hacer: $J = \{$ $DF1x = Der(F1, x), \quad DF1y = Der(F1, y);$ $DF2x = Der(F2, x), \quad DF2y = Der(F2, y);$ $\}$
Paso 2	$V = \{ \ -F1;$ $-F2 \ \}$
Paso 3	Desde $i = 0, 1, ..., (i_{max})$ $Sol = \ Solve[J, V]$
Paso 4	Hacer: $h = Sol(1); \quad j = Sol(2)$
Paso 5	$x_{i+1} = x_i + h$ $y_{i+1} = y_i + j$
Paso 6	$Dist = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}$
Paso 7	Si $\ Dist < \varepsilon$ Imprimir[La solución es x_{i+1}, y_{i+1}] Detener;

5.2. Método de Broyden.

En la sección 3.1.4 se analizó el método de la secante, el cual cambia la derivada en el método de Newton Raphson por una aproximación mediante diferencias divididas. El método de Broyden es una generalización del de la secante para sistemas de varias variables.

Partimos de la ecuación vectorial:

$$x^{k+1} = x^k + h^k \tag{5-13}$$

Donde:

$$h^k = -(J^k)^{-1} f^k \tag{5-14}$$

Sustituyendo (5-14) en (5-13):

$$x^{k+1} = x^k - (J^k)^{-1} f^k \tag{5-15}$$

El método de Broyden consiste en sustituir el inverso del jacobiano de la ecuación (5-15) por una matriz $(A^k)^{-1}$ que está dada por:

$$(A^k)^{-1} = (A^{k-1})^{-1} + \frac{[\Delta x^k - (A^{k-1})^{-1}\Delta f^k](\Delta x^k)^T (A^{k-1})^{-1}}{(\Delta x^k)^T (A^{k-1})^{-1}\Delta f^k} \tag{5-16}$$

La ecuación 5-16 es la fórmula del método de Broyden. El primer valor de $(A^{k-1})^{-1}$ es el jacobiano calculado con los valores iniciales. El primer valor de Δx y Δf se determina como el estimado inicial más el valor de h y j calculado con Newton-Raphson multivariable.

Los valores obtenidos de $(Ak)^{-1}$ se deben sustituir en la ecuación (5-15) para obtener el nuevo vector solución. El proceso iterativo continúa hasta alcanzar la convergencia.

A continuación se muestra el pseudocódigo para este método:

ENTRADA	Estimados iniciales x_0, y_0 funciones $F1(x, y) = 0$ y $F2(x, y) = 0$, número máximo de iteraciones i_{max}, error ε
SALIDA	Solución aproximada x, y.

Paso 1	Hacer: $i = 0$
	J= {
	$DF1x = Der(F1, x_i)$, $DF1y = Der(F1, y_i)$;
	$DF2x = Der(F2, x_i)$, $DF2y = Der(F2, y_i)$;
	}

Paso 2	$V = \{ -F1(x_i, y_i)$; $-F2\ (x_i, y_i)\}$
Paso 3	Hacer:

	$A_i = \text{inverse}(J)$
Paso 4	$Sol = \text{Solve}[J, V]$
Paso 5	$h = Sol(1); \quad j = Sol(2)$
Paso 6	$x_{i+1} = x_i + h; \qquad y_{i+1} = y_i + j;$
Paso 7	$X_i = \{x_i; \qquad X_{i+1} = \{x_{i+1};$ $y_i\} \qquad y_{i+1}\}$
Paso 8	Desde $i = 0, 1, \ldots, (i_{max})$
	$\Delta X_{i+1} = X_{i+1} - X_i$
	$\Delta XT_{i+1} = \text{Transponer}(\Delta X_{i+1})$
Paso 9	$\Delta f_{i+1} = \begin{pmatrix} F1(x_{i+1}, y_{i+1}) \\ F2(x_{i+1}, y_{i+1}) \end{pmatrix} - \begin{pmatrix} F1(x_i, y_i) \\ F2(x_i, y_i) \end{pmatrix}$
Paso 10	$(A^{i+1}) = (A^i) + \frac{[\Delta X^{i+1} - A^i * \Delta f^{i+1}](\Delta XT^{i+1})(A^i)}{(\Delta XT^{i+1})(A^i)\Delta f^{i+1}}$
Paso 11	$X_{i+2} = X_{i+1} - (A^{i+1}) * \begin{pmatrix} F1(x_{i+1}, y_{i+1}) \\ F2(x_{i+1}, y_{i+1}) \end{pmatrix}$
Paso 12	$x_{i+2} = X_{i+2}(1) \quad // \text{Extraer elemento 1}$
Paso 13	$y_{i+2} = X_{i+2}(2) \quad // \text{Extraer elemento 2}$
Paso 14	$Dist = \sqrt{(x_{i+2} - x_{i+1})^2 + (y_{i+2} - y_{i+1})^2}$
Paso 15	Si $\quad Dist < \varepsilon$
	Imprimir[La solución es x_{i+2}, y_{i+2}]
	Detener;

5.3. Problema de sistemas de ecuaciones algebraicas no lineales. Equilibrio químico. [13]

Considere las siguientes reacciones simultáneas con constantes de equilibrio $K_1 = 2.667$ y $K_2 = 3.200$:

$$A + B \rightleftarrows C + D$$

$$A + C \rightleftarrows 2E$$

La concentración inicial es $[\mathrm{A}] = 2\ \mathrm{mol/L}$ y $[\mathrm{B}] = 1\ \mathrm{mol/L}$. Calcule las concentraciones de cada especie al equilibrio.

5.3.1. Solución por el método de Newton-Raphson multivariable.

Primero se realizan los balances molares

$$A + B \rightleftarrows \mathrm{C} + \mathrm{D}$$

$$2\text{-x} \quad 1\text{-x} \quad \mathrm{x} \quad \mathrm{x}$$

$$A + C \rightleftarrows 2\mathrm{E}$$

$$2\text{-x-y} \quad \mathrm{x\text{-}y} \quad 2\mathrm{y}$$

$$[A] = 2 - x - y$$
$$[B] = 1 - x$$
$$[C] = x - y$$
$$[D] = x$$
$$[E] = 2y$$

Las expresiones para la constante de equilibrio son:

$$K_1 = \frac{[C][D]}{[A][B]}$$

$$K_2 = \frac{[E]^2}{[A][C]}$$

Sustituyendo datos en las ecuaciones de las constantes de equilibrio e igualando a cero:

$$F_1(x,y) = \frac{x^2(x-y)}{(2-x-y)} - 2.667 = 0$$

$$F_2(x,y) = \frac{4y^2}{(2-x-y)(x-y)} - 3.200 = 0$$

El problema a resolver es un sistema de dos ecuaciones no lineales con dos incógnitas. Para calcular el valor de x y de y se escribió el programa 5.1, que se encuentra en la siguiente página:

```
imax = 15;
tol = 10^-6;
x[0] = 0.75; (* Estimados iniciales *)
y[0] = 0.5;
F1[x_, y_] := x (x - y) / ((2 - x - y) (1 - x)) - 2.667;
F2[x_, y_] := 4 y^2 / ((2 - x - y) (x - y)) - 3.2;

DF1x = D[F1[x[i], y[i]], x[i]];
DF2x = D[F2[x[i], y[i]], x[i]];
DF1y = D[F1[x[i], y[i]], y[i]];
DF2y = D[F2[x[i], y[i]], y[i]];

J = ( DF1x  DF1y
      DF2x  DF2y );

(* Vector de funciones *)
V = ( -F1[x[i], y[i]]
      -F2[x[i], y[i]] );

Print["i\t x\t\t y"];

For[i = 0, i ≤ imax, i++,
 Sol = LinearSolve[J, V];
 h = Sol[[1]][[1]];
 j = Sol[[2]][[1]];

 x[i + 1] = x[i] + h;
 y[i + 1] = y[i] + j;

 Dist = √((x[i + 1] - x[i])^2 + (y[i + 1] - y[i])^2);
 If[Dist ≤ tol,
  Break[]
 ];
 Print[i, "\t", N[x[i]], "\t", N[y[i]]];
]
```

Programa 5.1. Solución mediante el método de Newton-Raphson

Los resultados en pantalla después de correr el programa se muestran a continuación:

i	x	y
0	0.75	0.5
1	0.906764	0.501933
2	0.867657	0.468745
3	0.841656	0.461237
4	0.834554	0.4599
5	0.834164	0.45982

$$x = 0.834164 \qquad y = 0.45982$$

Una vez conocidos x y y, se sustituyen en las ecuaciones de balances molares. Las concentraciones calculadas al equilibrio son:

$[A] = 0.706016\ mol/L$
$[B] = 0.165836\ mol/L$
$[C] = 0.374344\ mol/L$
$[D] = 0.834164\ mol/L$
$[E] = 0.91964\ mol/L$
$Total = 3\ mol/L$

Tabla 5.1. Concentraciones al equilibrio con el método de Newton-Raphson multivariable.

El total es 3 mol/L, con lo cual se comprueba que el resultado es correcto, pues son los mismos moles totales que había inicialmente y no se viola la ley de la conservación de la materia.

5.3.2. Solución por el método de Broyden.

El programa 5.2 se escribió para solucionar el problema por el método de Broyden.

```
imax = 15;
i = 0;
tol = 10^-6;
```

Programa 5.2. Solución mediante el método de Broyden. (Parte 1 de 2).

```
F1[x_, y_] := x (x - y) / ((2 - x - y) (1 - x)) - 2.667;
F2[x_, y_] := 4 y^2 / ((2 - x - y) (x - y)) - 3.2;

i = 0;
(* Derivadas parciales *)
DF1x = D[F1[x[i], y[i]], x[i]];
DF2x = D[F2[x[i], y[i]], x[i]];
DF1y = D[F1[x[i], y[i]], y[i]];
DF2y = D[F2[x[i], y[i]], y[i]];
(* Fin. Derivadas parciales*)

(* Matriz de derivadas parciales *)
J = ( DF1x  DF1y
      DF2x  DF2y );

(* Vector de funciones *)
V = ( -F1[x[i], y[i]]
      -F2[x[i], y[i]] );
x[0] = 0.75; (* Estimados iniciales *)
y[0] = 0.5;
A[0] = Inverse[J];
Sol = LinearSolve[J, V];
h = Sol[[1]][[1]];
j = Sol[[2]][[1]];
x[1] = x[0] + h;
y[1] = y[0] + j;
X[0] = ( x[0]
         y[0] );  X[1] = ( x[1]
                           y[1] );

Print["i\t X[i]\n\n0\t", MatrixForm[X[0]]];
Print["1\t", MatrixForm[N[X[1]]]];

For[i = 0, i ≤ imax, i++,
 ΔX[i + 1] = X[i + 1] - X[i];
 ΔXT[i + 1] = Transpose[ΔX[i + 1]];
```

Programa 5.2. Solución mediante el método de Broyden. (Parte 2 de 3).

```
Δf[i + 1] = {{F1[x[i + 1], y[i + 1]]}, {F2[x[i + 1], y[i + 1]]}} - {{F1[x[i], y[i]]}, {F2[x[i], y[i]]}};

(* Fórmula del método de Broyden *)
A[i + 1] = A[i] + ((ΔX[i + 1] - (A[i].Δf[i + 1])).(ΔXT[i + 1].A[i]))/((ΔXT[i + 1].A[i]).Δf[i + 1])[[1]][[1]];

X[i + 2] = X[i + 1] - A[i + 1].{{F1[x[i + 1], y[i + 1]]}, {F2[x[i + 1], y[i + 1]]}};


x[i + 2] = X[i + 2][[1]][[1]];
y[i + 2] = X[i + 2][[2]][[1]];

Dist = √((x[i + 2] - x[i + 1])^2 + (y[i + 2] - y[i + 1])^2);

If[Dist ≤ tol,
 Break[]
];

Print[i + 2, "\t", MatrixForm[N[X[i + 2]]]];

]
```

Programa 5.2. Solución mediante el método de Broyden. (Parte 3 de 3).

Los resultados después de correr el programa 5.2 son:

i	X[i]
0	$\begin{pmatrix} 0.75 \\ 0.5 \end{pmatrix}$
1	$\begin{pmatrix} 0.906764 \\ 0.501933 \end{pmatrix}$
2	$\begin{pmatrix} 0.793804 \\ 0.464048 \end{pmatrix}$
3	$\begin{pmatrix} 0.815682 \\ 0.462134 \end{pmatrix}$

(Continúa)

4	$\begin{pmatrix} 0.839604 \\ 0.460957 \end{pmatrix}$
5	$\begin{pmatrix} 0.833684 \\ 0.46036 \end{pmatrix}$
6	$\begin{pmatrix} 0.834231 \\ 0.460085 \end{pmatrix}$
7	$\begin{pmatrix} 0.834201 \\ 0.459928 \end{pmatrix}$
8	$\begin{pmatrix} 0.834164 \\ 0.459823 \end{pmatrix}$
9	$\begin{pmatrix} 0.834163 \\ 0.45982 \end{pmatrix}$

$\mathrm{x} = 0.834163 \qquad \mathrm{y} = 0.45982$

Y al sustituirlos en las ecuaciones de balance molar, se obtiene lo siguiente:

$[A] = 0.706017\ mol/L$
$[B] = 0.165837\ mol/L$
$[C] = 0.374343\ mol/L$
$[D] = 0.834163\ mol/L$
$[E] = 0.91964\ mol/L$
$Total = 3\ mol/L$

Tabla 5.2. Concentraciones al equilibrio con el método de Broyden.

5.4. Comparación de los métodos utilizados.

	Newton-Raphson multivariable		Broyden	
i	x	y	x	y
0	0.75	0.5	0.75	0.5
1	0.96764	0.501933	0.906764	0.501933
2	0.867657	0.468745	0.793804	0.464048
3	0.841656	0.461237	0.815682	0.462134
4	0.834554	0.4599	0.839604	0.460957
5	0.834164	0.45982	0.833684	0.46036
6	--	--	0.834231	0.460085
7	--	--	0.834201	0.459928
8	--	--	0.834164	0.459823
9	--	--	0.834163	0.45982

Tabla 5.3. Comparación entre el método de Newton-Raphson multivariable y el método de Broyden.

La tabla 5.3 muestra una comparación entre el método de Newton-Raphson multivariable y el método de Broyden. Como puede verse, ambos llegan a la misma solución. Sin embargo, el método de Newton Raphson lo hace en menos iteraciones.

El método de Broyden, además de converger en más iteraciones que el de Newton, utilizó más líneas de código (esto se puede ver comparando el programa 5.1 contra el 5.2). Por tal motivo, se podría decir que es menos eficiente. Aun así, recomendaría utilizarlo en caso de que no se pueda calcular el inverso del jacobiano del sistema, que fue la razón por la que lo desarrolló Broyden, según él mismo dice en uno de sus artículos [10].

Por otro lado, algunos sistemas de ecuaciones algebraicas no lineales pueden llevar a distintos vectores solución, por lo que hay que tener precauciones al momento de seleccionar los estimados iniciales. El sistema que se resolvió en este capítulo también tiene raíces menores que cero. Sin embargo se descartan porque una concentración con signo negativo no tiene sentido. Tampoco se pueden aceptar valores que violen la ley de la conservación de la materia. Siempre debe tenerse en cuenta la naturaleza del problema, así como las restricciones físicas a las que está sujeto.

Capítulo 6. Integración numérica.

Es frecuente encontrar funciones que no se pueden integrar de manera analítica. En este capítulo se analizarán métodos que nos permitirán obtener una aproximación del área bajo la curva, aun cuando la función sea difícil de integrar analíticamente o sólo se tenga en forma tabular.

6.1. Regla del trapecio.

La primera de estas técnicas es la regla del trapecio, que pertenece a las fórmulas de integración de Newton-Cotes. Este tipo de fórmulas se basan en sustituir una función complicada o datos tabulados por un polinomio fácil del integrar, es decir:

$$\int_a^b f(x)dx \cong \int_a^b f_n(x)dx \tag{6-1}$$

Con:

$$f_n(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1} + a_n x^n \tag{6-2}$$

Donde n es el grado del polinomio.

La regla del trapecio corresponde al caso donde el polinomio de la ecuación (6-2) es de primer grado, es decir, una línea recta. Esta línea puede ser representada como:

$$f_1(x) = f(a) + \frac{f(b) - f(a)}{a - a}(x - a) \tag{6-3}$$

Donde a y b son los límites inferior y superior, respectivamente. Podemos integrar la ecuación 6-3 analíticamente.

$$I = \int_a^b [f(a) + \frac{f(b) - f(a)}{a - a}(x - a)] \tag{6-4}$$

$$I = (b - a)\frac{f(a) + f(b)}{2} \tag{6-5}$$

La interpretación gráfica de la regla del trapecio se muestra en la figura 6.1.

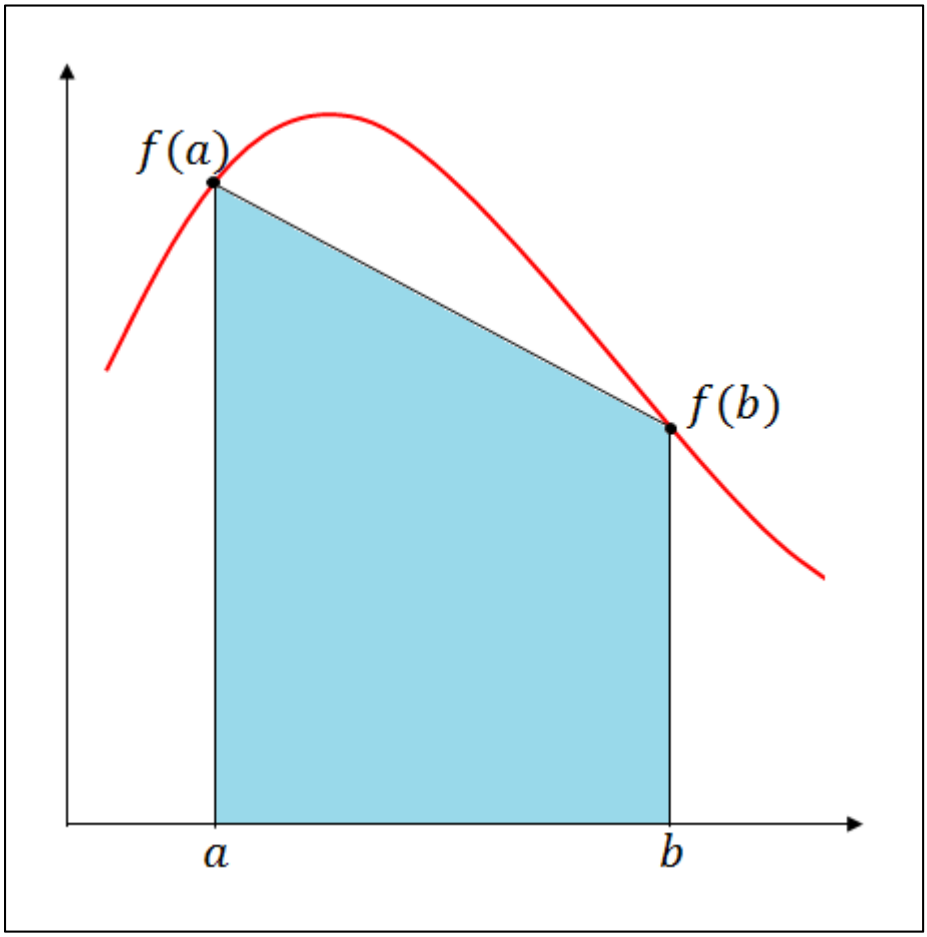


Figura 6.1. Interpretación gráfica de la regla del trapecio.

Como puede verse en la figura 6.1, aproximar el área bajo la curva con un segmento de línea recta puede conducir a errores importantes. La ecuación (6-5) se puede mejorar dividiendo el intervalo de integración en varios segmentos para enseguida aplicar el método a cada uno. Al final se suma el resultado de cada segmento y se obtiene un resultado más exacto de la integral en todo el intervalo. El ancho de los subintervalos en los que se divida estará dado por:

$$h = \frac{b - a}{n} \tag{6-6}$$

Donde n es la cantidad de subintervalos.

Si designamos a y b como x_1 y x_n, respectivamente, la integral completa es:

$$I = \int_{x_0}^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + \ldots + \int_{x_{n-1}}^{x_n} f(x)dx \qquad (6\text{-}7)$$

Aplicando la regla del trapecio a las integrales de (6-7):

$$I \cong h\frac{f(x_0)+f(x_1)}{2} + h\frac{f(x_1)+f(x_2)}{2} + \cdots + h\frac{f(x_{n-1})+f(x_n)}{2} \qquad (6\text{-}8)$$

Agrupando términos:

$$I \cong \frac{h}{2}\left[f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)\right] \qquad (6\text{-}9)$$

La ecuación (6-9) es la regla del trapecio de aplicación múltiple y proporciona un resultado más exacto al aproximar el área bajo la curva mediante varios segmentos. En la figura 6.2 se puede ver la representación gráfica de este método.

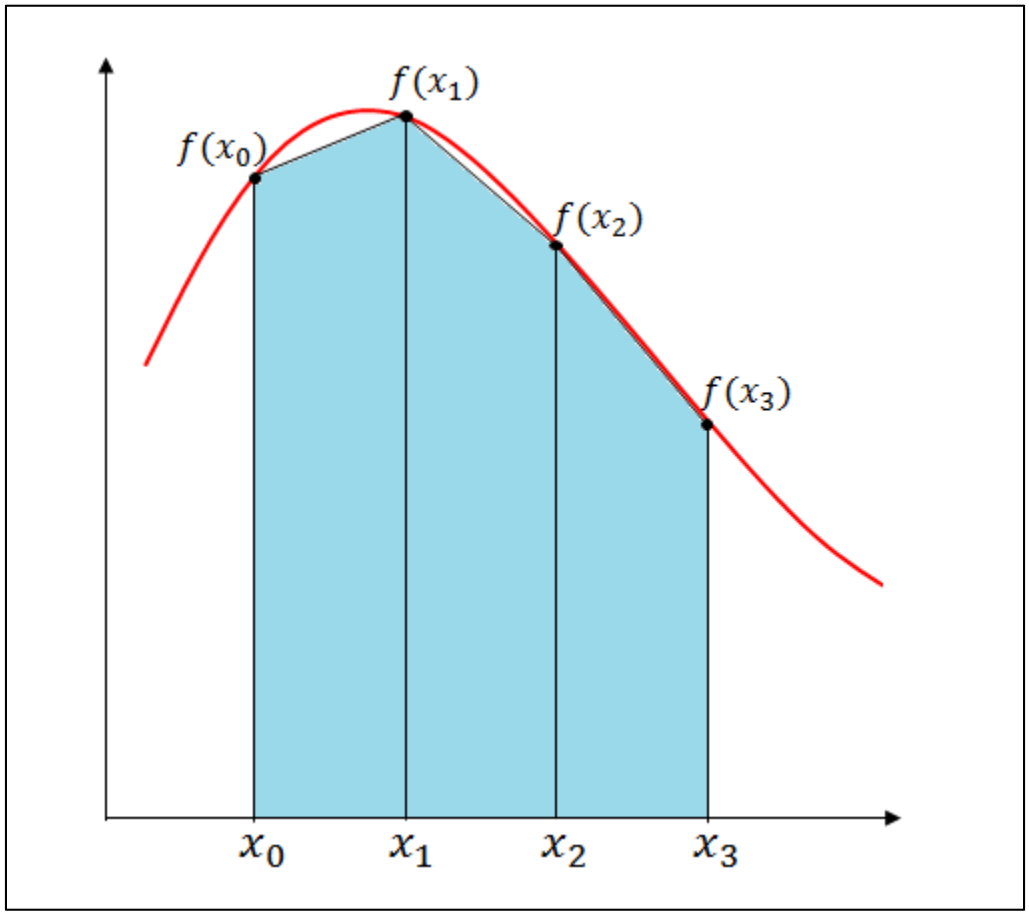


Figura 6.2. Interpretación gráfica de la regla del trapecio de aplicación múltiple.

El pseudocódigo para este método es el siguiente:

ENTRADA	Límites a y b, función $\mathrm{F(x)}$, tamaño de subintervalo h
SALIDA	Área aproximada A
Paso 1	Hacer:
	$\mathrm{n = (b-a) \,/\, h}$
	$\mathrm{Suma = 0}$
Paso 2	Desde $\mathrm{i = 1, 2, ..., n}$
	Hacer:
	$\mathrm{Suma = Suma + F(a + i * h)}$
	Fin Desde
Paso 3	$\mathrm{A = (h/2) * [F(a) + 2 * Suma + F(b)]}$
Paso 4	Imprimir (“La integral es ”, A)

6.2. Regla de Simpson.

Otra de las fórmulas de integración de Newton-Cotes es la regla de Simpson, la cual calcula la integral con mayor exactitud que la regla del trapecio. Este método consiste en unir los puntos con un polinomio de grado superior.

En el capítulo 2 se analizaron los polinomios de interpolación. Si sustituimos un polinomio de Lagrange de segundo grado en la ecuación (6-1), la integral estará dada por:

$$\int_a^b f(x)dx \cong \left[\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}f(x_0) + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}f(x_1) + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}f(x_2)\right] \tag{6-10}$$

Y se puede integrar analíticamente, dando como resultado [1]:

$$I \cong \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)] \tag{6-11}$$

La ecuación (6-11) se conoce como la regla de Simpson 1/3.

Al igual que con la regla del trapecio, es posible mejorar la regla de Simpson dividiendo el intervalo de integración en varios segmentos del mismo tamaño. La integral total se puede representar como:

$$I = \int_{x_0}^{x_2} f(x)dx + \int_{x_2}^{x_4} f(x)dx + \ldots + \int_{x_{n-2}}^{x_n} f(x)dx \tag{6-12}$$

Aplicando la regla de Simpson 1/3 a cada integral:

$$I \cong 2h\frac{f(x_0) + 4f(x_1) + f(x_2)}{6} + 2h\frac{f(x_2) + 4f(x_3) + f(x_4)}{6} + \cdots + 2h\frac{f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)}{6} \tag{6-13}$$

Agrupando términos:

$$I \cong \frac{h}{3}\left[f(x_0) + 4\sum_{\substack{i=1 \\ \Delta i=2}}^{n-1} f(x_i) + 2\sum_{\substack{i=2 \\ \Delta i=2}}^{n-2} f(x_i) + f(x_n)\right] \tag{6-14}$$

A la ecuación (6-14) se le conoce como la regla de Simpson 1/3 de aplicación múltiple. El término $\Delta \mathrm{i} = 2$ significa que en cada suma, el subíndice i va incrementando de dos en dos.

Con la regla de Simpson, es posible aproximar el área de manera más exacta, ya que en lugar de usar segmentos de línea recta como lo hace la regla del trapecio, se utilizan secciones de parábolas que se aproximan a la función con tres puntos. Cada parábola se forma con dos subintervalos, por esta razón el intervalo total de integración debe dividirse en un número par de subintervalos. De lo contrario el método no funcionará.

La figura 6.3 muestra una visualización de este método.

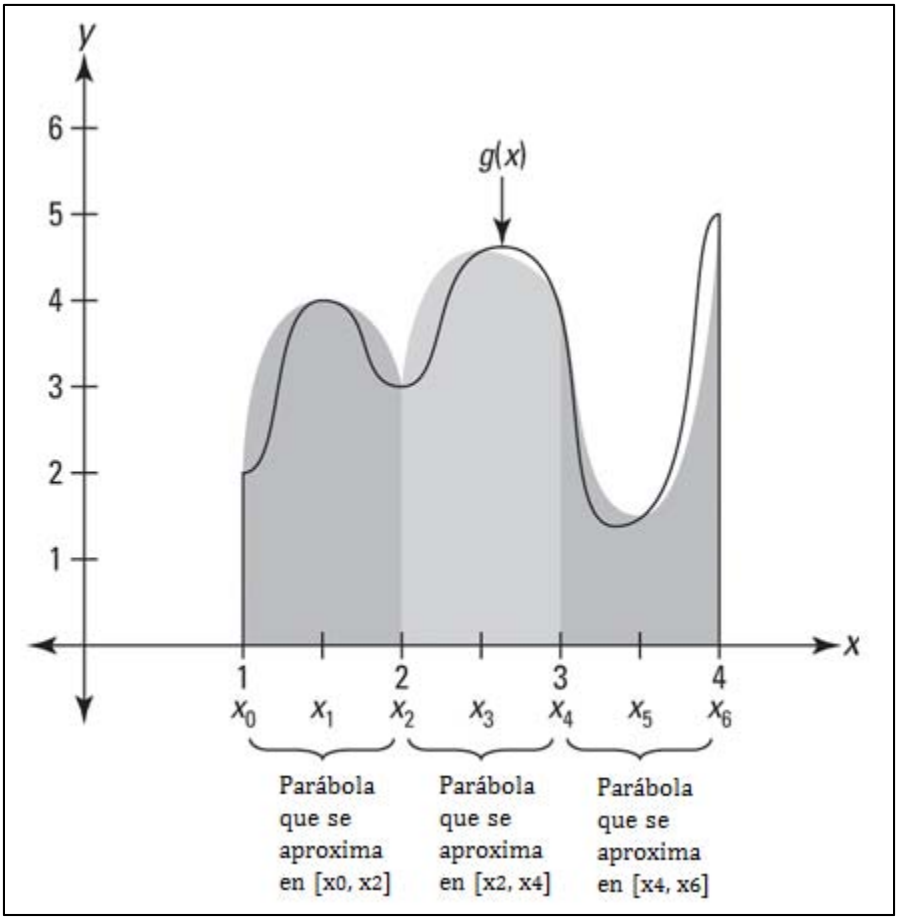


Figura 6.3. Interpretación gráfica de la regla de Simpson 1/3 de aplicación múltiple.

El pseudocódigo para la regla de Simpson es el siguiente:

ENTRADA	Límites a y b, función $F(x)$, tamaño de subintervalo h;
	$\text{Con n \% 2} = 0$
SALIDA	Área aproximada A
Paso 1	Hacer:
	$n = (b\text{-}a) / h$
Paso 2	Función: $\text{Suma1}( ) =$
	$S1 = 0$
	Desde $i = 1, 3, ..., (n\text{-}1), \Delta i = 2$
	$S1 = S1 + F(a + i * h)$
	Fin Función
Paso 3	Función: $\text{Suma2}( ) =$

	S2= 0
	Desde i = 2, 4, ..., (n-1), Δi = 2
	S2 = S2 + F(b + i * h)
	Fin Función
Paso 4	Función: Simpson( ) =
	Correr Suma1(), Suma2();
	A =(h / 3) * [F(a) + 4 * S1 + 2* S2 + F(b)]
	Fin Función
Paso 5	Correr Simpson
Paso 6	Imprimir ("La integral es ", A)

6.3. Cuadratura de Gauss.

Los métodos que se analizaron anteriormente, conocidos como ecuaciones de Newton-Cotes, permiten estimar el área bajo la curva basándose en valores igualmente espaciados de la función. Por ende, los puntos utilizados eran predeterminados o fijos.

La cuadratura de Gauss es una técnica que elimina la restricción de usar puntos fijos. El objetivo es determinar los coeficientes de una ecuación de la forma:

$$I \cong w_1 F(z_1) + w_2 F(z_2) \qquad (6\text{-}15)$$

Donde los términos w son coeficientes desconocidos. Sin embargo, los argumentos de la función z_1 y z_2 no están fijos en los extremos, sino que son incógnitas. De este modo, se tiene una ecuación con cuatro incógnitas, por lo que se requieren cuatro condiciones para determinarlas.

Para obtener las primeras dos condiciones, suponemos que la ecuación (6-15) calcula con exactitud la integral de una constante y de una función lineal. Las dos restantes se obtienen suponiendo que también ajusta la integral de una función parabólica y de una cúbica.

Las ecuaciones a resolver son entonces:

$$w_1 f(z_1) + w_2 F(z_2) = \int_{-1}^{1} dz = 2 \tag{6-16}$$

$$w_1 f(z_1) + w_2 F(z_2) = \int_{-1}^{1} z\, dz = 0 \tag{6-17}$$

$$w_1 f(z_1) + w_2 F(z_2) = \int_{-1}^{1} z^2\, dz = \frac{2}{3} \tag{6-18}$$

$$w_1 f(z_1) + w_2 F(z_2) = \int_{-1}^{1} z^3\, dz = 0 \tag{6-19}$$

Este sistema puede resolverse para dar:

$$w_1 = w_2 = 1 \qquad z_1 = -\frac{1}{\sqrt{3}} = -0.5773503 \qquad z_2 = \frac{1}{\sqrt{3}} = 0.5773503$$

Sustituyendo las soluciones en la ecuación (6-15):

$$I \cong F\left(-\frac{1}{\sqrt{3}}\right) + F\left(\frac{1}{\sqrt{3}}\right) \tag{6-20}$$

La ecuación (6-20) es la fórmula de Gauss-Legendre de dos puntos, la cual tiene una exactitud de tercer grado. Los límites se definen de -1 a 1 para que la fórmula sea lo más general posible.

Para cambiar los límites de integración, se supone que una variable z está relacionada linealmente con la variable original.

$$x = a_0 + a_1 z \tag{6-21}$$

Si el límite inferior x = a corresponde a z = -1, sustituimos estos valores en la ecuación (6-21):

$$a = a_0 + a_1(-1) \tag{6-21}$$

Lo mismo se realiza si el límite superior $\mathrm{x} = \mathrm{b}$ corresponde a $\mathrm{z} = 1$:

$$b = a_0 + a_1(1) \tag{6-22}$$

Resolviendo simultáneamente (6-21) y (6-22) para a_0 y b_0:

$$a_0 = \frac{b+a}{2} \tag{6-23}$$

$$a_1 = \frac{b-a}{2} \tag{6-24}$$

Sustituyendo (6-23) y (6-24) en (6-21):

$$x = \frac{(b+a)+(b-a)z}{2} \tag{6-25}$$

Diferenciando obtenemos:

$$dx = \frac{b-a}{2}dz \tag{6-26}$$

Las ecuaciones (6-25) y (6-26) pueden sustituirse por x y dx en la función que se vaya a integrar. Dichas sustituciones transforman el intervalo de integración sin alterar el valor de la integral.

Existen fórmulas de más puntos para calcular el área bajo la curva con mayor exactitud que la ecuación (6-20). De manera general se escriben como:

$$I \cong w_1F(z_1) + w_2F(z_2) + \cdots + w_nF(z_n) \tag{6-27}$$

donde: n = número de puntos.

La tabla 6.1 muestra los valores de w y de z para fórmulas de Gauss-Legendre de hasta cinco puntos [2].

Número de puntos	Coeficientes w_i	Abscisas z_i
2	$w_1 = w_2 = 1$	$-z_1 = z_2 = 0.5773022692$
3	$w_2 = 0.888888\ldots$ $w_1 = w_3 = 0.55555\ldots$	$-z_1 = z_3 = 0.7745966692$ $z_2 = 0$
4	$w_1 = w_4 = 0.3478548451$ $w_2 = w_3 = 0.6521451549$	$-z_1 = z_4 = 0.8611363116$ $-z_2 = z_3 = 0.3399810436$
5	$w_1 = w_5 = 0.2369268851$ $w_2 = w_4 = 0.4786286705$ $w_3 = 0.568888\ldots$	$-z_1 = z_5 = 0.9061798459$ $-z_2 = z_4 = 0.5384693101$ $z_3 = 0$

Tabla 6.1. Coeficientes y abscisas para el método de cuadratura de Gauss

La exactitud obtenida con las fórmulas de Gauss-Legendre depende del número de puntos elegido. Con dos puntos, se tendrá exactitud de tercer grado; con tres, se tendrá exactitud de cuarto grado y así sucesivamente.

6.4. Problemas de integración numérica.

La integración numérica es ampliamente aplicada en diversos campos de la ingeniería. El modelado de reactores químicos por lo regular implica el cálculo de integrales que no se pueden resolver analíticamente.

En esta sección se verán dos problemas en los cuales se pueden aplicar las técnicas de integración analizadas.

6.4.1. Cálculo del volumen de reactor a partir de datos tabulares. [14]

Ejemplo 2-3 del libro *Elementos de ingeniería de las reacciones químicas*, de H. Scott Fogler [14].

La reacción que describen los datos de la tabla 6.2 debe correrse en un PFR. La velocidad de flujo molar alimentada de A es de $0.4\ \mathrm{mol/s}$. Calcule el volumen de un PFR que se necesita para alcanzar una conversión del 80%.

X	0.0	0.1	0.2	0.4	0.6	0.7	0.8
$\frac{F_{A0}}{-r_A}\ (m^3)$	0.89	1.08	1.33	2.05	3.54	5.06	8.0

Tabla 6.2. Datos procesados para el reactor PFR.

6.4.1.1. Solución por la regla del trapecio.

La forma diferencial del balance molar para un PFR es:

$$F_{A0}\frac{dX}{dV} = -r_A$$

Reacomodando e integrando:

$$V = \int_0^{0.8} \frac{F_{A0}\,dX}{-rA}$$

Para calcular el volumen se debe evaluar la integral numérica. El programa 6.1 realiza este cálculo usando la regla del trapecio.

```
(* Fogler. Ejemplo 2-3 a) *)
X = {0, 0.1, 0.4, 0.6, 0.8};
F = {0.89, 1.08, 2.05, 3.54, 8};
n = Length[X] - 1;

a = X[[1]];
b = X[[n + 1]];
h = (b - a)/(Length[X] - 1);
A = h/2 (F[[1]] + 2 * Sum[F[[j]], {j, 2, n}] + F[[n + 1]]);

Print["El volumen es ", A, " dm^3"];
```

Programa 6.1. Cálculo del volumen del reactor mediante la regla del trapecio.

El resultado que aparece en pantalla después de correr el programa es el siguiente:

```
El volumen es 2.223 dm^3
```

6.4.1.2. Solución por la regla de Simpson.

Para resolver el mismo problema mediante la regla de Simpson, se escribió el programa 6.2.

```
X = {0, 0.1, 0.4, 0.6, 0.8};
F = {0.89, 1.33, 2.05, 3.54, 8};
n = Length[X] - 1;
a = X[[1]];
b = X[[n + 1]];

h = (b - a)/n;
```

Programa 6.2. Cálculo del volumen del reactor mediante regla de Simpson. (Parte 1 de 2).

```
Suma1[] := (
    S1 = 0;
    For[i = 2, i < (n + 1), i += 2,
     S1 = S1 + F[[i]];
    ]
   );

Suma2[] := (
    S2 = 0;
    For[i = 3, i ≤ (n), i += 2,
     S2 = S2 + F[[i]];
    ]
   );

RunSimpson[] := (
    Suma1[]; Suma2[];
    V = h/3 * (F[[1]] + 4 * S1 + 2 * S2 + F[[n + 1]]);
    Print["El volumen es ", V, "dm^3"];
   );
RunSimpson[];
```

Programa 6.2. Cálculo del volumen del reactor mediante regla de Simpson. (Parte 2 de 2).

Y el resultado después de correrlo es:

```
El volumen es 2.16467dm^3
```

El cual es más cercano al valor de 2.165 dm^3 que reporta el autor.

6.4.2. Cálculo de la constante cinética. [13]

Se tiene la siguiente reacción en fase líquida:

$$2A \rightleftarrows B$$

La cual tiene una constante de equilibrio $K = 5$ y concentraciones iniciales $[C_{A0}]=0.6$ y $[C_{B0}]=0.05$.

En 30 minutos, se alcanza un 50% de conversión. ¿Cuál es el valor de la constante cinética de la reacción hacia adelante?

6.4.2.1. Solución por la regla del trapecio.

La rapidez de desaparición del reactivo A está dada por la ecuación diferencial:

$$-\frac{dC_A}{dt} = k_1 {C_A}^2 - k_2 C_B$$

La constante de equilibrio puede expresarse como un cociente de las constantes cinéticas:

$$K = \frac{k_1}{k_2}$$

Despejamos k_2 y sustituimos en la ecuación de rapidez.

$$-\frac{dC_A}{dt} = k_1 ({C_A}^2 - \frac{1}{K} C_B)$$

Si la conversión es 50%, la concentración de B está dada por:

$$C_B = C_{B0} + 0.5(C_{A0} - C_A)$$
$$C_B = 0.05 + 0.5(0.6 - C_A)$$
$$C_B = 0.35 - 0.5C_A$$

Sustituyendo CB y K en la ecuación de rapidez:

$$-\frac{dC_A}{dt} = k_1 \left[{C_A}^2 - \frac{1}{5}(0.35 - 0.5C_A) \right]$$
$$-\frac{dC_A}{dt} = k_1 \left[{C_A}^2 + 0.1C_A - 0.07 \right]$$

Separando variables e integrando:

$$k = \frac{1}{t} \int_{CA}^{CA0} \frac{dC_A}{{C_A}^2 + 0.1C_A - 0.07}$$

Para calcular la constante cinética, es necesario evaluar la integral del lado derecho y dividir el resultado entre 30 minutos. El programa 6.2 realiza este cálculo mediante la regla del trapecio.

```
CA0 = 0.6; (* C en mol/L *)
X = 0.5;
CAF = X * CA0;
t = 30; (* min *)
h = 0.05;
n = (CA0 - CAF)/h;
F[C_] := 1/(C^2 + 0.1 * C - 0.07); (* Función a integrar *)

A = h/2 (F[CA0] + 2 * (Sum_{i=1}^{n} F[CAF + i * h]) + F[CAF]);
Print["El área bajo la curva es " A];
k = 1/t * A;
Print["La constante cinética es ", k, " L/(mol * min)"];
```

Programa 6.3. Cálculo de la constante cinética mediante la regla del trapecio

Nótese que, a diferencia del problema anterior, esta vez la función no está dada en forma tabular.

El resultado que da este programa es:

```
El área bajo la curva es 2.34332

La constante cinética es 0.0781108 L/(mol * min)
```

6.4.2.2. Solución por la regla de Simpson.

El programa 6.4 se escribió para resolver este mismo problema por medio de la regla de Simpson.

```
CA0 = 0.6; (* C en mol/L *)
X = 0.5;
CAF = X * CA0;
t = 30; (* min *)
h = 0.05;
n = (CA0 - CAF)/h;
F[C_] := 1/(C^2 + 0.1 * C - 0.07); (* Función a integrar *)

Suma1[] := (
   S1 = 0;
   For[i = 1, i < (n), i += 2,
    S1 = S1 + F[CAF + i * h];
   ]
  );
Suma2[] := (
   S2 = 0;
   For[i = 2, i ≤ (n - 1), i += 2,
    S2 = S2 + F[CAF + i * h];
   ]
  );
RunSimpson[] := (
   Suma1[]; Suma2[];
   A = h/3 * (F[CAF] + 4 * S1 + 2 * S2 + F[CA0]);
   k = 1/t * A;
   Print["El área bajo la curva es ", A];
   Print["La constante cinética es ", k, " L/(mol * min)"];
  );
RunSimpson[];
```

Programa 6.4. Cálculo de la constante cinética mediante la regla de Simpson.

Y el resultado que muestra este programa es:

El área bajo la curva es 2.15167

La constante cinética es 0.0717222 $\frac{L}{mol * min}$

6.4.2.3. Solución por cuadratura de Gauss.

Este problema también se resolvió por el método de cuadratura de Gauss. El programa 6.5 se escribió para solucionarlo. Es posible escoger entre dos, tres y cuatro puntos.

```
CA0 = 0.6; (* C en mol/L *)
X = 0.5;
CAF = X * CA0;
t = 30; (* min *)
a = CAF;
b = CA0;
puntos = 4;
F[C_] := 1/(C^2 + 0.1 * C - 0.07); (* Función a integrar *)
Gauss[p_] := (
  A = 0;
  (* Asignar valores a w y z *)
  If[p == 2,
   z[1] = -0.5773502692; w[1] = 1;
   z[2] = -z[1];  w[2] = 1,
   If[p == 3,
     z[1] = -0.7745966692; w[1] = 0.55555;
     z[2] = 0;  w[2] = 0.88888;
     z[3] = -z[1]; w[3] = w[1],
     If[p == 4,
       z[1] = -0.8611363116; w[1] = 0.3478548451;
       z[2] = -0.3399810436;  w[2] = 0.6521451549;
       z[3] = -z[2]; w[3] = w[2];
       z[4] = -z[1]; w[4] = w[1],
       Break[];
```

Programa 6.5. Cálculo de la constante cinética mediante cuadratura de Gauss. (Parte 1 de 2).

```
      ];
    ];
  ];
  For[i = 1, i ≤ p, i++,
   A = A + w[i] * F[(b - a)/2 * z[i] + (a + b)/2];
  ];
  A = A * (b - a)/2;
); (* Termina la función Gauss[] *)

Gauss[puntos];
k = 1/t * A;
Print["El área bajo la curva es ", A];
Print["La constante cinética es ", k, " L/(mol * min)"];
```

Programa 6.5. Cálculo de la constante cinética mediante cuadratura de Gauss. (Parte 2 de 2).

Y el resultado que da este programa es:

```
El área bajo la curva es 2.14492
La constante cinética es 0.0714973 L/(mol * min)
```

2.3. Comparación de los métodos utilizados.

En el primer problema se proporcionó una función en forma tabular, por lo que sólo fue posible aplicar las fórmulas de Newton-Cotes, que son la regla del trapecio y la regla de Simpson. La tabla 6.3 de la siguiente página muestra una comparación de los resultados.

Método	Valor obtenido	%Error
Regla del trapecio	2.223	2.68%
Regla de Simpson	2.16467	0.015%

Tabla 6.3. Comparación entre la regla del trapecio y la regla de Simpson

El porcentaje de error se calculó considerando que Fogler [14] reporta un valor de 2.165 dm^3 para el volumen del PFR.

La regla de Simpson puede calcular el área exacta para un polinomio de grado 3 o menor [12]. Por esta razón es mucho más exacto que la regla del trapecio.

El segundo problema la función ya no se presentó en forma tabular, por lo que también fue posible aplicar distintas fórmulas de Gauss-Legendre. La tabla 6.4 muestra una comparación.

Método	Valor obtenido	%Error
Regla del trapecio (n=6)	0.0781108	9.18%
Regla de Simpson	0.0717222	0.24977%
Cuadratura de Gauss (n=2)	0.0692236	3.225%
Cuadratura de Gauss (n=3)	0.0712118	0.46363%
Cuadratura de Gauss (n=4)	0.0714973	0.0646%

Tabla 6.4. Comparación entre los métodos de integración numérica.

El porcentaje de error se calculó considerando el valor teórico de 0.0715435. La regla del trapecio con seis subintervalos fue el método más inexacto, con 9.18% de error, mientras que la regla de Simpson fue el segundo método más exacto con 0.25% de error. El método más exacto fue el de cuadratura de Gauss con 4 puntos (0.065% de error), el cual tiene una exactitud de quinto orden. En la tabla 6.4 se observa cómo el porcentaje de error se reduce al aumentar el número de puntos.

Capítulo 7. Ecuaciones diferenciales.

Hay una gran cantidad de fenómenos físicos que se pueden modelar por medio de ecuaciones diferenciales. Recuérdese que una ecuación diferencial en x y y es una ecuación que incluye x y derivadas de y [16].

Una función $y = f(x)$ se denomina solución de una ecuación diferencial si la ecuación se satisface al sustituir y y sus derivadas por $f(x)$ y sus derivadas. Por ejemplo, considérese la ecuación diferencial:

$$y' + 2y = 0 \tag{7-1}$$

Cada solución de la ecuación (7-1) es de la forma:

$$y = Ce^{-2x} \tag{7-2}$$

Donde C es cualquier número real.

A la ecuación (7-2) se le denomina solución general. Geométricamente, la solución general de una ecuación diferencial representa una familia de curvas solución; una para cada valor de C. Si queremos obtener la solución particular, que es lo que nos interesa en este capítulo, es necesario tener las condiciones iniciales, que dan el valor de y o una de sus derivadas para un valor de x.

Las ecuaciones diferenciales pueden ser ordinarias o parciales. Las ordinarias contienen derivadas de una o más variables dependientes con respecto a una sola variable independiente, mientras que las parciales contienen derivadas respecto a dos o más variables independientes [16].

Aunque hay distintas formas de determinar la solución particular analíticamente, en ingeniería es frecuente encontrar ecuaciones diferenciales complicadas cuya solución analítica sería imposible o conllevaría mucho esfuerzo. Por esta razón se analizarán las principales técnicas que nos permiten obtener la solución numéricamente.

7.1. Ecuaciones diferenciales ordinarias.

7.1.1. Método de Euler.

El método de Euler es un método numérico que aproxima la solución particular de una ecuación diferencial de la forma:

$$y' = f(x, y) \tag{7-3}$$

Si se conoce un punto (x_0, y_0) por el que pasa la ecuación (7-3), se sabe que la curva solución pasa a través de él con una pendiente igual a $f(x, y)$. Esto da un primer punto para para aproximar la solución.

A partir del primer punto, continuamos en la dirección indicada por la pendiente. Usando un tamaño de paso h, nos desplazamos a lo largo de la recta tangente hasta llegar a un nuevo punto (x_1, y_1). El proceso se repite, ahora tomando el nuevo punto calculado para obtener una nueva aproximación (x_2, y_2). En general, la fórmula del método de Euler es:

$$y_{i+1} = y_i + h\, f(x_i, y_i) \tag{7-3}$$

Donde h es el tamaño de paso elegido y x está dada por:

$$x_{i+1} = x_i + h \tag{7-4}$$

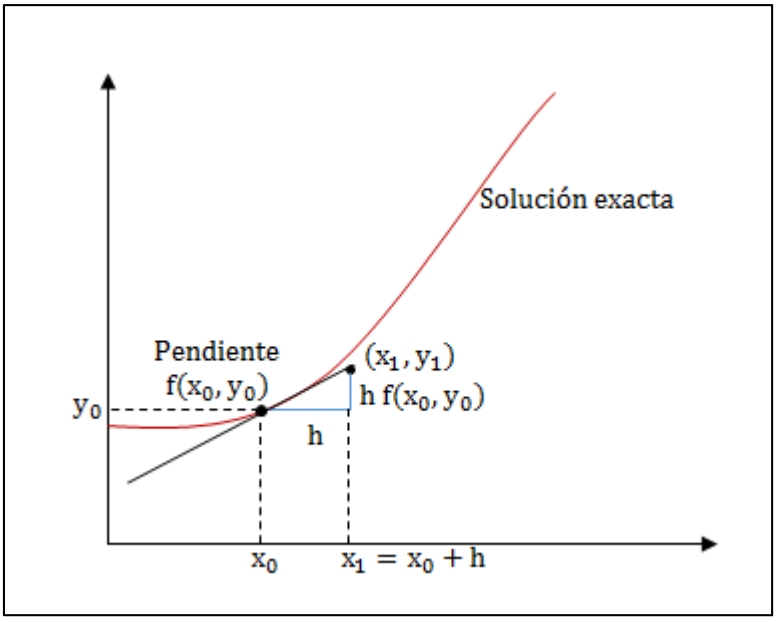


Figura 7.1. Interpretación gráfica del método de Euler.

Nótese que es posible mejorar la exactitud de la solución escogiendo un tamaño de paso más pequeño.

El pseudocódigo para este método es el siguiente:

ENTRADA	Condición inicial (x_0, y_0) función $y' = f(x)$, número máximo de iteraciones i_{max}, tamaño de paso h
SALIDA	Puntos solución (x, y)
Paso 1	Desde $i = 0, 1, ..., (i_{max})$
	$x_{i+1} = x_i + h$
	$y_{i+1} = y_i + h * f(x_i, y_i)$
Paso 2	$\text{Imprimir}(x_i, y_i)$
	Fin Desde.

7.1.2. Método de Runge-Kutta de cuarto orden.

Los métodos de Runge-Kutta, comúnmente abreviados como RK, logran la exactitud de la serie de Taylor sin necesidad de calcular derivadas de orden superior. En general, estos métodos tienen la forma:

$$y_{i+1} = y_i + f(x_i, y_i, h)h \tag{7-5}$$

Al término $f(xi, yi, h)$ se le conoce como función incremento, y se interpreta como una pendiente representativa en el intervalo. Está función se escribe en forma general como:

$$f(x_i, y_i, h) = a_1 k_1 + a_2 k_2 + \cdots + a_n k_n \tag{7-6}$$

Donde las a son constantes y las k están dadas por:

$$k_1 = f(x_i, y_i) \tag{7-7}$$

$$k_2 = f(x_i + p_1 h, y_i + q_{11} k_1 h) \tag{7-8}$$

$$k_3 = f(x_i + p_2 h, y_i + q_{21} k_1 h + q_{22} k_2 h) \tag{7-9}$$

$$\cdots$$

$$k_n = f(x_i + p_{n-1} h, y_i + q_{n-1,1} k_1 h + q_{n-1,2} k_2 h + \cdots + q_{n-1,n-1} k_{n-1} h) \tag{7-10}$$

Donde p y q son constantes, mientras que n es el orden del método. Como puede verse, el método de Runge-Kutta de orden 1 corresponde al método de Euler.

El más popular entre los métodos de Runge-Kutta es el de cuarto orden, también conocido como RK4. La fórmula para este método es:

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h \quad (7\text{-}11)$$

Y las pendientes k se calculan como:

$$k_1 = f(x_i, y_i) \quad (7\text{-}12)$$

$$k_2 = f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1 h) \quad (7\text{-}13)$$

$$k_3 = f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_2 h) \quad (7\text{-}14)$$

$$k_4 = f(x_i + h, y_i + k_3 h) \quad (7\text{-}15)$$

El pseudocódigo para este método se muestra a continuación:

ENTRADA	Condición inicial (x_0, y_0) función $y' = f(x)$, número máximo de iteraciones i_{max}, tamaño de paso h
SALIDA	Puntos solución (x, y)
Paso 1	Desde $i = 0, 1, ..., (i_{max})$
	$x_{i+1} = x_i + h$
Paso 2	$k_1 = f(x_i, y_i)$
Paso 3	$k_2 = f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1 h)$
Paso 4	$k_3 = f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_2 h)$
Paso 5	$k_4 = f(x_i + h, y_i + k_3 h)$
Paso 6	$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h$
Paso 7	$\text{Imprimir}(x_i, y_i)$
	Fin Desde.

7.1.3. Sistemas de ecuaciones diferenciales ordinarias.

Los métodos vistos hasta ahora se pueden aplicar también a sistemas de ecuaciones diferenciales ordinarias. El procedimiento consiste simplemente en aplicar el método a cada ecuación en cada iteración. También es necesario conocer las condiciones iniciales. Para el método de Euler el procedimiento se detalla en el siguiente pseudocódigo.

ENTRADA	Condiciones iniciales $(x_0, y1_0), (x_0, y2_0)$
	funciones $y_1' = f_1(x, y1, y2)$, $y_2' = f_2(x, y1, y2)$ número
	máximo de iteraciones i_{max}, tamaño de paso h
SALIDA	Puntos solución (x, y_1, y_2)
Paso 1	Desde $i = 0, 1, ..., (i_{max})$
	$x_{i+1} = x_i + h$
	$y1_{i+1} = y1_i + h * f_1(x_i, y1_i, y2_i)$
	$y2_{i+1} = y2_i + h * f_2(x_i, y1_i, y2_i)$
Paso 2	$\text{Imprimir}(x_i, y1_i, y2_i)$
	Fin Desde.

En el caso del método RK4 es necesario definir, además de las pendientes k, otro grupo de pendientes m para la segunda ecuación diferencial.

ENTRADA	Condiciones iniciales $(x_0, y1_0), (x_0, y2_0)$
	funciones $y_1' = f(x, y1, y2)$, $y_2' = f(x, y1, y2)$ número
	máximo de iteraciones i_{max}, tamaño de paso h
SALIDA	Puntos solución (x, y_1, y_2)
Paso 1	Desde $i = 0, 1, ..., (i_{max})$
	$x_{i+1} = x_i + h$
Paso 2	$k_1 = f_1(x_i, y1_i, y2_i)$
	$m_1 = f_2(x_i, y1_i, y2_i)$
Paso 3	$k_2 = f_1(x_i + \frac{1}{2}h, y1_i + \frac{1}{2}k_1 h, y2_i + \frac{1}{2}m_1 h)$

	$m_2 = f_2(x_i + \frac{1}{2}h,\ y1_i + \frac{1}{2}k_1 h,\ y2_i + \frac{1}{2}m_1 h)$
Paso 4	$k_3 = f_1(x_i + \frac{1}{2}h,\ y1_i + \frac{1}{2}k_2 h,\ y2_i + \frac{1}{2}m_2 h)$
	$m_3 = f_2(x_i + \frac{1}{2}h,\ y1_i + \frac{1}{2}k_2 h,\ y2_i + \frac{1}{2}m_2 h)$
Paso 5	$k_4 = f_1(x_i + h,\ y1_i + k_3 h,\ y2_i + m_3 h)$
	$m_4 = f_2(x_i + h,\ y1_i + k_3 h,\ y2_i + m_3 h)$
Paso 6	$y1_{i+1} = y1_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h$
	$y2_{i+1} = y2_i + \frac{1}{6}(m_1 + 2m_2 + 2m_3 + m_4)$
Paso 7	$\text{Imprimir}(\ x_i, y1_i, y2_i\ )$
	Fin Desde.

7.2. Problemas de ecuaciones diferenciales ordinarias.

7.2.1. Integración de una ecuación cinética. [15]

La rapidez de desaparición de un reactivo A está dada por la ecuación:

$$\frac{dC_A}{dt} = -k\ {C_A}^n$$

Con $n = 1$, $k = 0.15\ s^{-1}$ y $C_{A0} = 1.00\ mol/L$. Calcule la concentración desde $t=0$ hasta $t = 15\ s$:

a) Analíticamente.
b) Con el método de Euler (h = 0.5 s).
c) Con el método de Runge-Kutta de cuarto orden (h = 0.5 s).

7.2.1.1. Solución analítica.

Separando variables se obtiene:

$$\frac{dC_A}{C_A} = -k\ dt$$

Integrando:

$$\int_{C_{A0}}^{[C_A]} \frac{dC_A}{C_A} = -k \int_0^t dt$$

$$\ln\left(\frac{C_A}{C_{A0}}\right) = -kt$$

$$\frac{C_A}{C_{A0}} = e^{-kt}$$

$$C_A = C_{A0}\, e^{-kt}$$

El programa 7.1 calcula la concentración a varios tiempos usando la fórmula analítica.

```
A[0] = 1; (* mol/L *)
t[0] = 0; (* s *)
n = 1;
k = 0.15 ; (* s^-1 *)
h = 0.5; (* s *)
imax = 41;
F[A_, t_] = A[0] * e^-k*t;
Print["t (s)\tC (mol/L)"];

(*Vectores de respuestas*)
x = {t[0]};
y = {A[0]};
For[i = 0, i < imax, i++,
 t[i + 1] = t[i] + h;
 A[i] = F[A[i], t[i]];
 Print[t[i], "\t\t", A[i]];

 (*Guardar soluciones en vector*)
 x = Append[x, t[i + 1]];
 y = Append[y, A[i + 1]];]

(* Crear matriz que contenga los datos
 para graficarlos *)
datosExacta = Transpose@{x, y};
```

Programa 7.1. Solución analítica de la ecuación cinética. (Parte 1 de 2).

```
ListPlot[datosExacta, AxesLabel → {"t", "C"},
 PlotStyle → {Blue}, PlotRange → {{0, 15}, {0, 1}},
 PlotMarkers → {●, 10}, Background → LightBlue,
 PlotLabel → Style["C vs t", FontSize → 18,
    FontColor → Black], GridLines → Automatic]
```

Programa 7.1. Solución analítica de la ecuación cinética. (Parte 2 de 2).

La tabla de resultados (resumida) y la gráfica que se obtienen con este programa se muestran en la figura 7.2.

t (s)	C (mol/L)
0	1.
3.	0.637628
6.	0.40657
9.	0.25924
12.	0.165299
15.	0.105399
18.	0.0672055

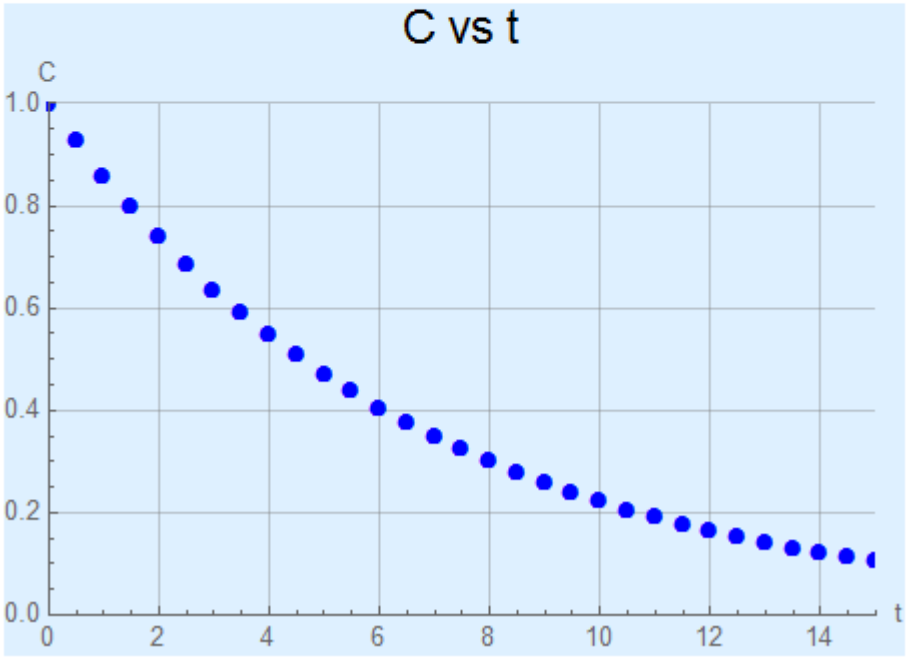


Figura 7.2. Resultados de la ecuación cinética obtenidos analíticamente.

Los resultados coinciden con los obtenidos por Levine, el autor del problema [15].

7.2.1.2. Solución por el método de Euler.

El programa 7.2 obtiene los resultados usando el método de Euler.

```
A[0] = 1; (* mol/L *)
t[0] = 0; (* s *)
n = 1;
k = 0.15 ; (* s^-1 *)
h = 0.5; (* s *)
imax = 41;
F[A_] = -k * A^n;(* dA/dt *)

(*Vectores de respuestas*)
xEuler = {t[0]};
yEuler = {A[0]};

Print["t (s)\tC (mol/L)"];

For[i = 0, i < imax, i++,
 A[i + 1] = A[i] + h * F[A[i]];
 t[i + 1] = t[i] + h;
 Print[t[i], "\t\t ", A[i]];

 (*Guardar soluciones en vector*)
 xEuler = Append[xEuler, t[i + 1]];
 yEuler = Append[yEuler, A[i + 1]];
]
(* Crear matriz que contenga los datos
 para graficarlos *)
datosEuler = Transpose@{xEuler, yEuler};

ListPlot[datosEuler, AxesLabel → {"t", "C"},
  PlotStyle → {Blue}, PlotRange → {{0, 15}, {0, 1}},
  PlotMarkers → {●, 10}, Background → LightBlue,
  PlotLabel → Style["C vs t", FontSize → 18,
    FontColor → Black], GridLines → Automatic];
```

Programa 7.2. Solución de la ecuación cinética por medio del método de Euler.

En la figura 7.3 se muestra la gráfica y un resumen de la tabla de resultados.

t (s)	C (mol/L)
0	1
3.	0.626398
6.	0.392375
9.	0.245783
12.	0.153958
15.	0.0964388
18.	0.0604091

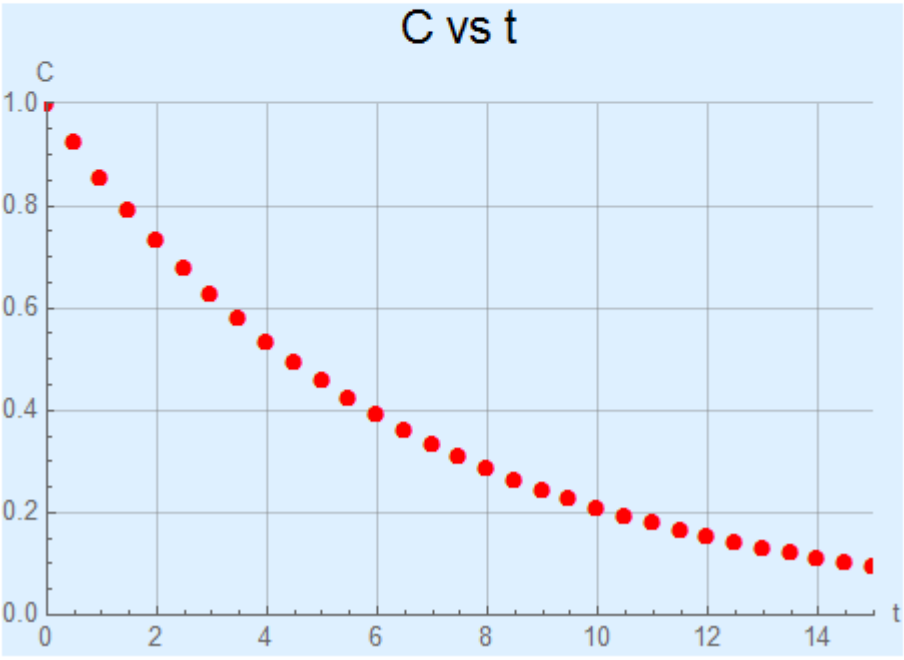


Figura 7.3. Resultados de la ecuación analítica obtenidos con el método de Euler.

7.2.1.3. Solución por el método de Runge-Kutta de 4to. orden.

El programa 7.3 resuelve este problema usando el método RK4.

```
A[0] = 1; (* mol/L *)
t[0] = 0; (* s *)
n = 1;
k = 0.15 ; (* s^-1 *)
h = 0.5; (* s *)
```

Programa 7.3. Solución de la ecuación cinética por medio del método de Runge-Kutta de cuarto orden. (Parte 1 de 2).

```
imax = 41;
F[A_] = -k * A^n; (* dA/dt *)

(*Vectores de respuestas*)
xRK = {t[0]};
yRK = {A[0]};

Print["t (s)\tC (mol/L)"];

For[i = 0, i < imax, i++,
 k1 = F[A[i]];
 k2 = F[A[i] + h/2 * k1];
 k3 = F[A[i] + h/2 * k2];
 k4 = F[A[i] + k3 * h];
 A[i + 1] = A[i] + h/6 (k1 + 2 * k2 + 2 * k3 + k4);
 t[i + 1] = t[i] + h;
 Print[t[i], "\t\t ", A[i]];

 (*Guardar soluciones en vector*)
 xRK = Append[xRK, t[i + 1]];
 yRK = Append[yRK, A[i + 1]];]
(* Crear matriz que contenga los datos
 para graficarlos *)
datosRK = Transpose@{xRK, yRK};

ListPlot[datosRK, AxesLabel → {"t", "C"},
 PlotStyle → {Red}, PlotRange → {{0, 15}, {0, 1}},
 PlotMarkers → {●, 10}, Background → LightBlue,
 PlotLabel → Style["C vs t", FontSize → 18,
   FontColor → Black], GridLines → Automatic]
```

Programa 7.3. Solución de la ecuación cinética por medio del método de Runge-Kutta de cuarto orden. (Parte 1 de 2).

En la figura 7.4 se muestra la gráfica y un resumen de la tabla de resultados para este método

t (s)	C (mol/L)
0	1
3.	0.637628
6.	0.40657
9.	0.25924
12.	0.165299
15.	0.105399
18.	0.0672056

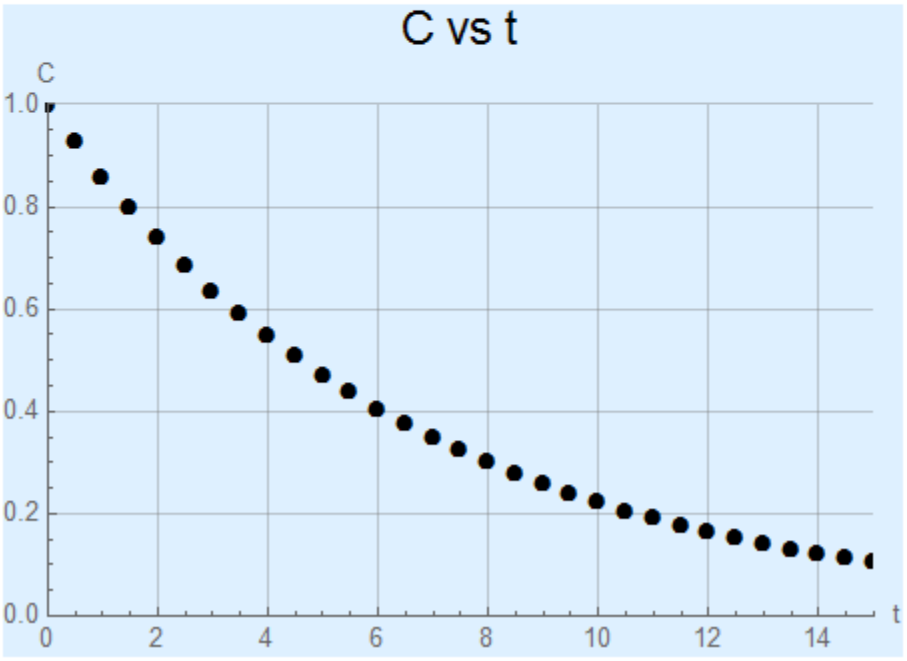


Figura 7.4. Resultados de la ecuación cinética obtenidos con el método de Runge-Kutta de cuarto orden.

7.2.2. Perfiles de conversión y temperatura en un PFR. [14]

La desintegración catalítica en fase vapor de la acetona se lleva a cabo en un PFR, según la siguiente reacción:

$$CH_3COCH_3 \rightarrow CH_2CO + CH_4$$

La reacción es de primer orden respecto a la acetona y la constante de rapidez está dada por:

$$k = 8x10^{14} \exp\left(\frac{-34222}{T}\right) s^{-1}$$

Se desea alimentar acetona a razón de 38.3 mol/s. La temperatura y presión de entrada son 1035 K y 162 kPa, respectivamente. Grafique los perfiles de conversión y la temperatura si el reactor opera adiabáticamente.

7.2.2.1. Solución por el método de Euler.

En este problema se deben resolver dos ecuaciones diferenciales ordinarias. De acuerdo con el autor Fogler [14], las ecuaciones para operación adiabática son:

$$-r_A = \frac{k\, C_{A0}(1-X)}{(1+X)}\frac{T_0}{T} \tag{7-11}$$

$$\frac{dX}{dV} = \frac{-r_A}{F_{A0}} \tag{7-12}$$

$$\frac{dT}{dV} = \frac{(-r_A)(\Delta H_R)}{F_{A0}(Cp_A + X\Delta Cp)} \tag{7-13}$$

El programa 7.4 resuelve (7-12) y (7-13) simultáneamente usando el método de Euler con h = 0.1. También calcula los valores requeridos como el Cp y ΔH.

```
T[0] = 1035; (* K *)
V[0] = 0; (* L *)
X[0] = 0;
Fa0 = 38.3;
Cpa = 26.63 + 0.183 T - 0.00004586 T^2; (* J/mol*K *)
ΔCp = 6.8 - 0.0115 T - 3.81 * 10^-6 T^2;
Ca0 = 18.8; (* mol/m^3 *)
TR = 298;(* T de referencia *)
ΔH = 80 770 + 6.8 (T - TR) - 0.00575 (T^2 - TR^2) - 1.27 * 10^-6 (T^3 - TR^3);
k = 8.2 * 10^14 * Exp[-34 222 * (1/T)]; (* s^-1 *)
```

Programa 7.4. Solución del problema utilizando el método de Euler. (Parte 1 de 3)

```
(* Vectores *)
xEuler = {V[0]};
y1Euler = {X[0]};
y2Euler = {T[0]};

h = 0.1; (* L *)
imax = 201;

F1[T_, X_] = -ra / Fa0 ;(* dX/dV *)
F2[T_, X_] = -ra (-ΔH) / Fa0 (Cpa + X * ΔCp) ; (* dT/dV *)


Print["V (L)\t X \t\tT (K)"];

For[i = 0, i < imax, i++,
 X[i + 1] = X[i] + h * F1[T[i], X[i]];
 T[i + 1] = T[i] + h * F2[T[i], X[i]];

 V[i + 1] = V[i] + h;
 Print[V[i], "\t\t", X[i], "\t\t", T[i]];
 xEuler = Append[xEuler, V[i + 1]];
 y1Euler = Append[y1Euler, X[i + 1]];
 y2Euler = Append[y2Euler, T[i + 1]];
]
datosEuler1 = Transpose@{xEuler, y1Euler};
datosEuler2 = Transpose@{xEuler, y2Euler};

(* Perfil de conversión *)
ListPlot[datosEuler1, AxesLabel → {"V (L)", "X"},
 PlotStyle → {Red},
 PlotMarkers → {●, 5}, Background → LightBlue,
 PlotLabel → Style["Perfil de conversión (Euler)",
   FontSize → 18, FontColor → Black],
 GridLines → Automatic]
```

Programa 7.4. Solución del problema utilizando el método de Euler. (Parte 2 de 3).

```
(* Perfil de temperatura *)
ListPlot[datosEuler2, AxesLabel → {"V (L))", "T (K)"},
 PlotStyle → {Red},
 PlotMarkers → {●, 5}, Background → LightBlue,
 PlotLabel → Style["Perfil de temperatura (Euler)",
   FontSize → 18, FontColor → Black],
 GridLines → Automatic]
```

Programa 7.4. Solución del problema utilizando el método de Euler. (Parte 3 de 3).

El programa 7.4 genera una tabla de resultados de doscientas filas y tres columnas. A continuación se muestra una pequeña parte de ella.

V (L)	X	T (K)
0	0	1035
2.	0.24281	917.916
4.	0.26915	904.238
6.	0.285265	895.796
8.	0.296835	889.699
10.	0.305834	884.936
12.	0.313183	881.034
14.	0.319382	877.732
16.	0.324738	874.872
18.	0.329446	872.353
20.	0.333645	870.102

Las figuras 7.5 y 7.6 muestran los perfiles de conversión y de temperatura, respectivamente, obtenidos mediante el método de Euler.

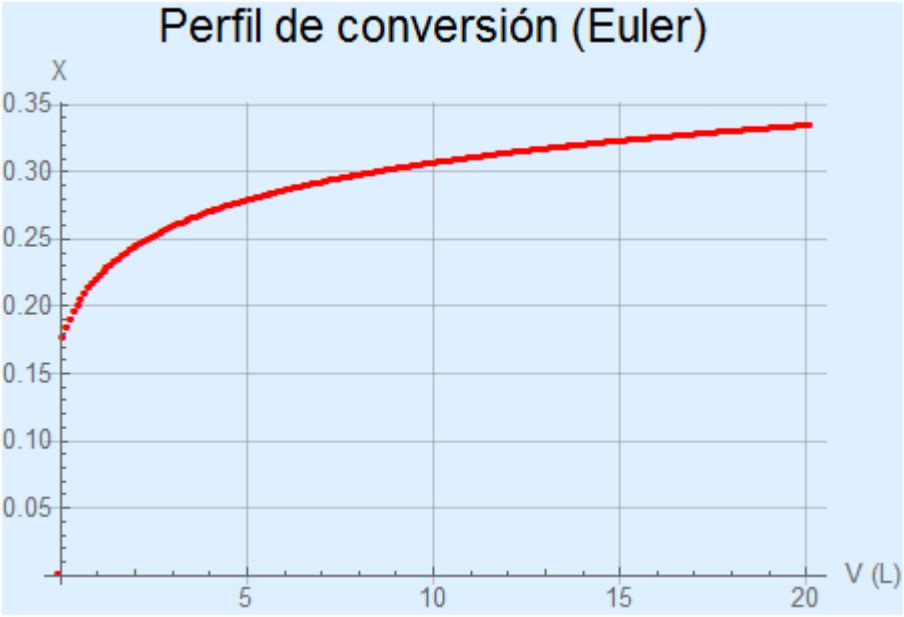


Figura 7.5. Perfil de conversión obtenido con el método de Euler.

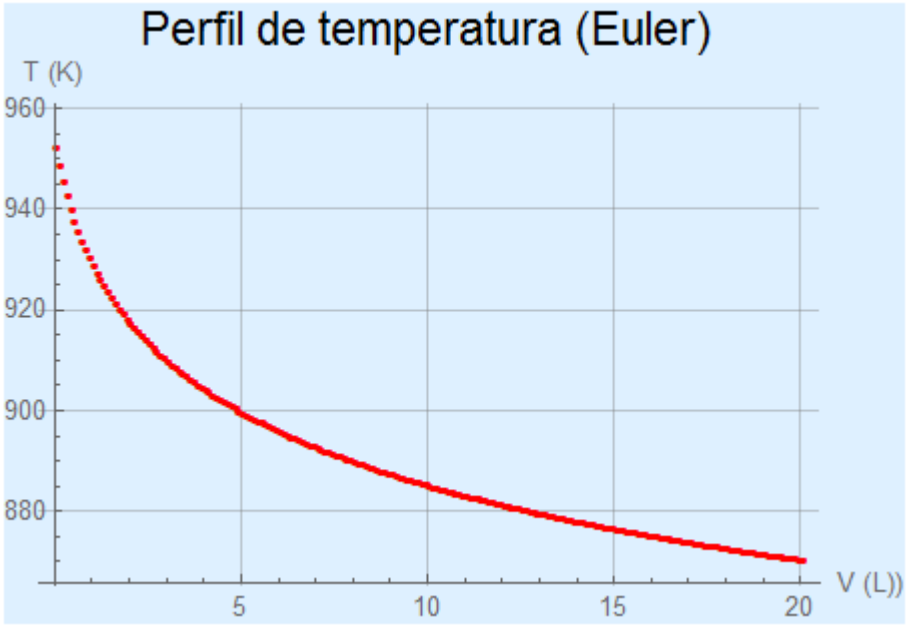


Figura 7.6. Perfil de temperatura obtenido con el método de Euler.

7.2.2.2. Solución por el método de Runge-Kutta de cuarto orden.

El programa 7.5, que comienza en la siguiente página, se utilizó para resolver el mismo problema por el método RK4.

```
T[0] = 1035; (* K *)
V[0] = 0; (* L *)
X[0] = 0;
Fa0 = 38.3;
Cpa = 26.63 + 0.183 T - 0.00004586 T^2; (* J/mol*K *)
ΔCp = 6.8 - 0.0115 T - 3.81 * 10^-6 T^2;
Ca0 = 18.8; (* mol/m^3 *)
TR = 298;(* T de referencia *)
ΔH = 80 770 + 6.8 (T - TR) - 0.00575 (T^2 - TR^2) - 1.27 * 10^-6 (T^3 - TR^3);
k = 8.2 * 10^14 * Exp[-34 222 (1/T)]; (* s^-1 *)
ra = (-k * Ca0 (1 - X))/(1 + X) * T[0]/T;

(* Vectores *)
xRK = {V[0]};
y1RK = {X[0]};
y2RK = {T[0]};

h = 0.1; (* L *)
imax = 201;

F1[T_, X_] = -ra/Fa0;(* dX/dV *)
F2[T_, X_] = -ra (-ΔH)/(Fa0 (Cpa + X * ΔCp)); (* dT/dV *)


Print["V (L)\t X \t\tT (K)"];

For[i = 0, i < imax, i++,
 k1 = F1[T[i], X[i]];
 m1 = F2[T[i], X[i]];
 k2 = F1[T[i] + h/2 * m1, X[i] + h/2 * k1];
 m2 = F2[T[i] + h/2 * m1, X[i] + h/2 * k1];
```

Programa 7.5. Solución del problema utilizando el método de Runge-Kutta de cuarto orden. (Parte 1 de 2).

```
  k3 = F1[T[i] + h/2 * m2, X[i] + h/2 * k2];
  m3 = F2[T[i] + h/2 * m2, X[i] + h/2 * k2];
  k4 = F1[T[i] + m3 * h, X[i] + k3 * h];
  m4 = F2[T[i] + m3 * h, X[i] + k3 * h];
  X[i + 1] = X[i] + h/6 (k1 + 2 * k2 + 2 * k3 + k4);
  T[i + 1] = T[i] + h/6 (m1 + 2 * m2 + 2 * m3 + m4);
  V[i + 1] = V[i] + h;
  Print[V[i], "\t\t", X[i], "\t\t", T[i]];
  xRK = Append[xRK, V[i + 1]];
  y1RK = Append[y1RK, X[i + 1]];
  y2RK = Append[y2RK, T[i + 1]];
]
datosRK1 = Transpose@{xRK, y1RK};
datosRK2 = Transpose@{xRK, y2RK};

(* Perfil de conversión *)
ListPlot[datosRK1, AxesLabel → {"X", "V (L)"},
 PlotStyle → {Black},
 PlotMarkers → {●, 5}, Background → LightBlue,
 PlotLabel → Style["Perfil de conversión (RK4)",
   FontSize → 18, FontColor → Black],
 GridLines → Automatic]

(* Perfil de temperatura *)
ListPlot[datosRK2, AxesLabel → {"T (K)", "V (L)"},
 PlotStyle → {Black},
 PlotMarkers → {●, 5}, Background → LightBlue,
 PlotLabel → Style["Perfil de temperatura (RK4)",
   FontSize → 18, FontColor → Black],
 GridLines → Automatic]
```

Programa 7.5. Solución del problema utilizando el método de Runge-Kutta de cuarto orden. (Parte 2 de 2).

A continuación se muestra la tabla de resultados resumida que se obtiene al correr el programa 7.5.

V (L)	X	T (K)
0	0	1035
2.	0.225739	924.145
4.	0.257854	907.53
6.	0.276089	897.998
8.	0.288763	891.331
10.	0.29844	886.215
12.	0.306248	882.074
14.	0.312778	878.598
16.	0.318384	875.608
18.	0.323289	872.985
20.	0.327645	870.651

Y los perfiles de conversión y temperatura obtenidos con Runge-Kutta de cuarto orden son los siguientes:

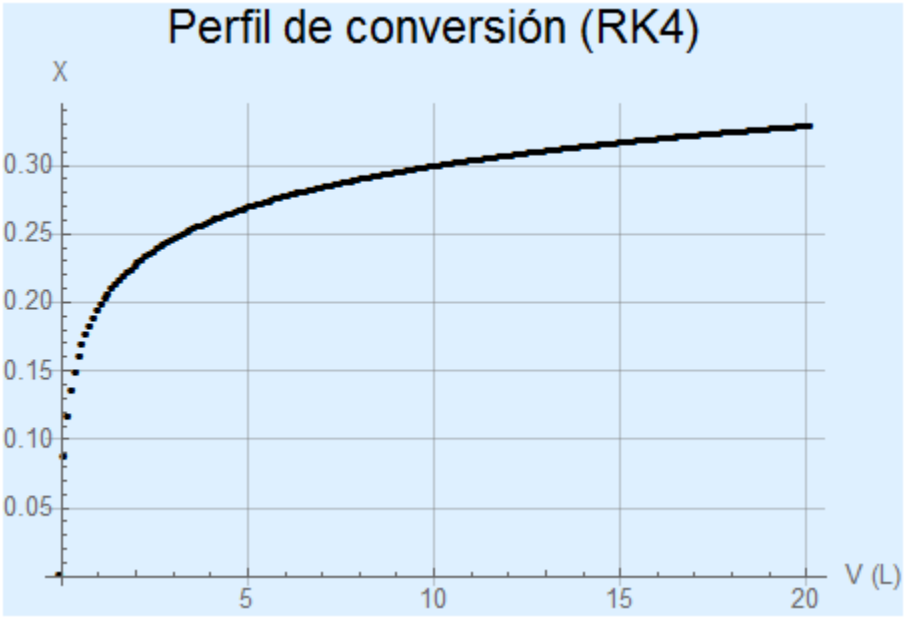


Figura 7.7. Perfil de conversión obtenido con el método de Runge-Kutta de 4to orden.

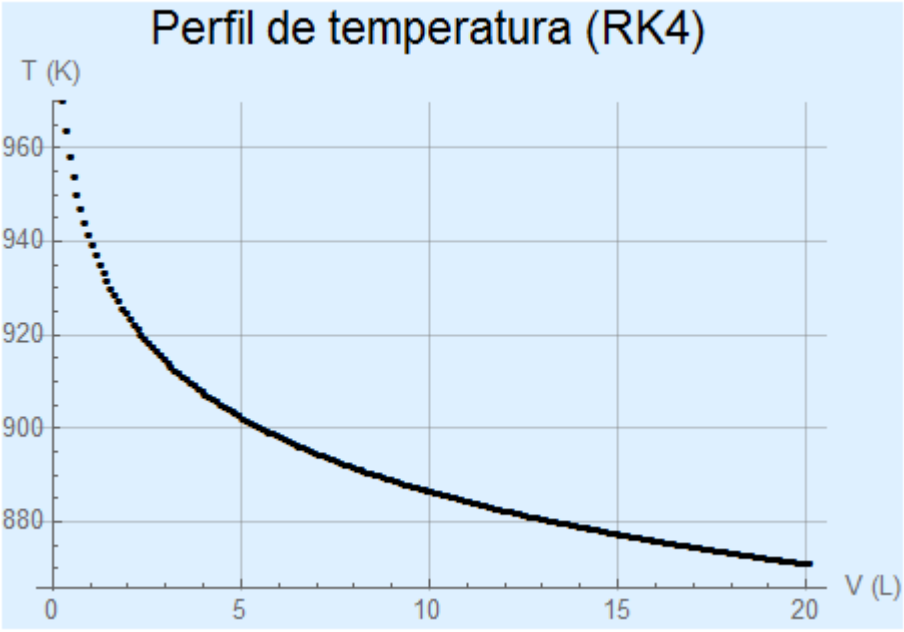


Figura 7.6. Perfil de temperatura obtenido con el método de Runge-Kutta de 4to orden.

7.3. Comparación de los métodos para resolver ecuaciones diferenciales ordinarias.

En el primer problema de este capítulo, se encontró la solución particular de una ecuación diferencial analítica y numéricamente. En la tabla 7.1 se muestra la comparación de los métodos numéricos utilizados contra la solución analítica para los tiempos 0.5, 2, 3 y 10 segundos.

t (s)	C calculada (Euler)	C calculada (RK4)	C exacta	%Error Euler	%Error RK4
0.5	0.925	0.927744	0.927743	0.29%	0.0001%
2	0.732094	0.740818	0.740818	1.17%	0%
3	0.626398	0.637628	0.637628	1.76%	0%
10	0.210298	0.22313	0.22313	5.75%	0%

Tabla 7.1. Comparación entre los métodos de Euler y Runge-Kutta.

Para esta ecuación, el método de Runge-Kutta de cuarto orden obtiene el resultado exacto. Asimismo, el método de Euler comienza con un error pequeño, el cual se

va haciendo mayor conforme se avanza en t. La comparación se puede visualizar mejor en la siguiente gráfica.

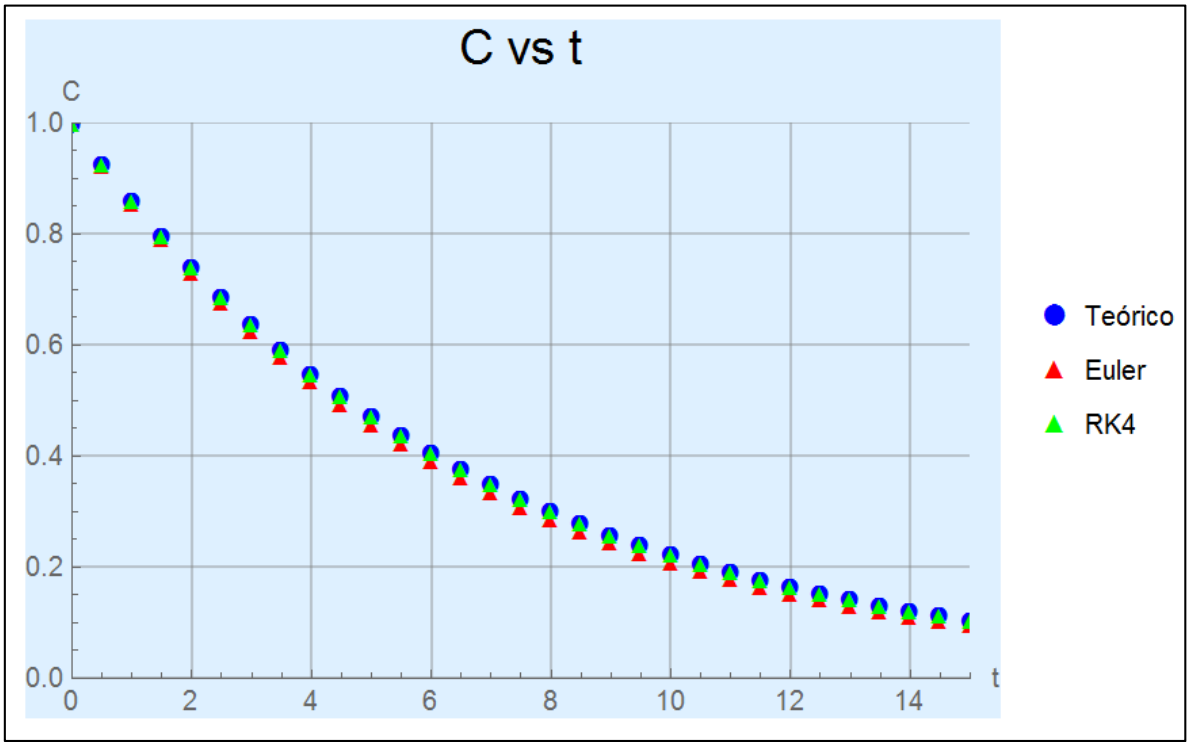


Figura 7.7. Comparación del método de Euler y Runge-Kutta contra la solución exacta de la ecuación diferencial.

El método de Runge-Kutta y la curva solución van prácticamente superpuestos. La gráfica de Euler sigue la misma tendencia pero se desvía ligeramente hacia abajo a medida que el tiempo pasa. La exactitud se puede mejorar reduciendo el tamaño de paso h, aunque esto también hará que el programa tarde un poco más en correr.

El segundo problema era más complicado; se trataba de un sistema de ecuaciones diferenciales y no había forma de resolverlas analíticamente de forma simultánea. En la siguiente tabla se comparan ambos métodos contra la solución teórica para dos valores de V.

V (L)	T calculada (Euler)	T calculada (RK4)	T teórica	%Error Euler	%Error RK4
0.5	939.648	957.891	958.584	1.98%	0.07%
1	929.968	941.031	941.367	1.21%	0.036%

Tabla 7.2. Comparación del método de Euler vs RK4 para el sistema de ecuaciones diferenciales.

En las siguientes gráficas se visualiza mejor la comparación entre ambos métodos.

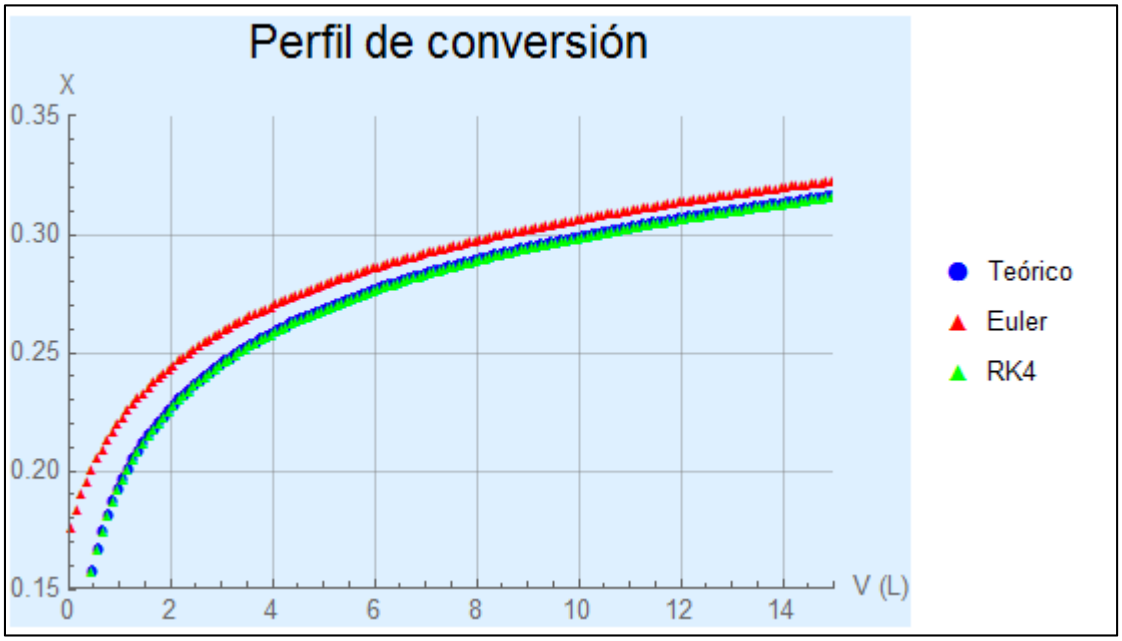


Figura 7.8. Comparación del método de Euler y Runge-Kutta contra la solución teórica para el perfil de conversión.

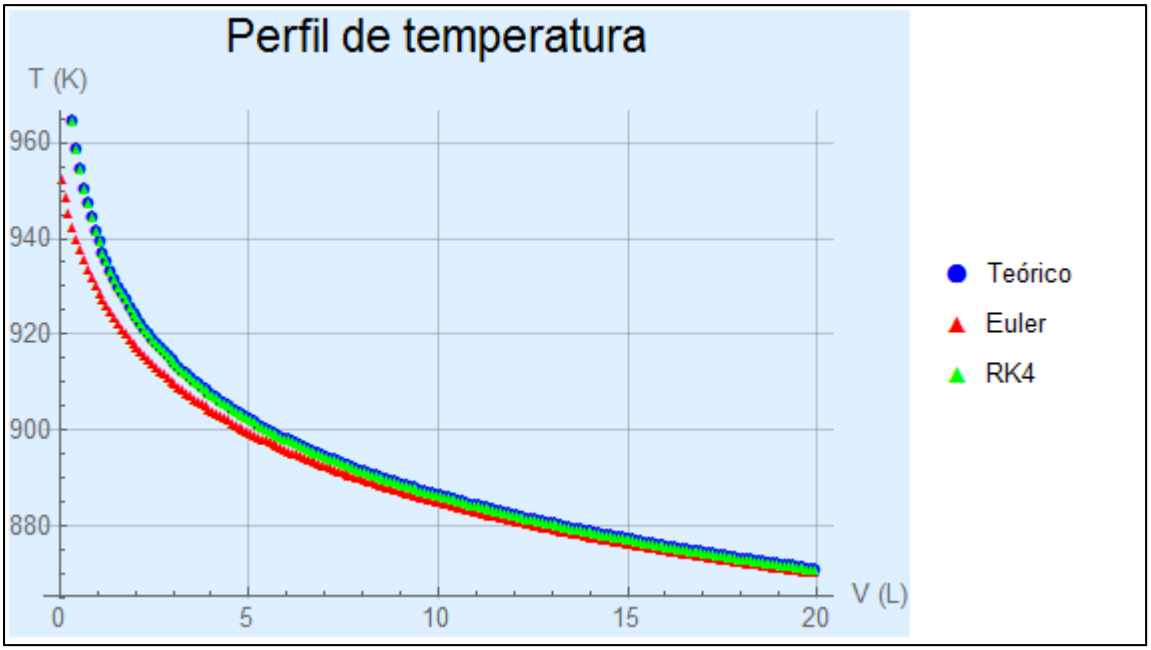


Figura 7.9. Comparación del método de Euler y Runge-Kutta contra la solución teórica para el perfil de temperatura.

El método de Runge-Kutta nuevamente sigue muy de cerca a la solución teórica. El de Euler inicia muy desviado pero va mejorando su exactitud conforme V avanza.

Los resultados son coherentes, pues el método RK4 considera un promedio de pendientes y por lo tanto su aproximación se acerca más al valor teórico.

El método de Euler es sencillo y fácil de programar, por lo que puede ser una opción si sólo se requiere conocer la forma de la curva solución sin importar la exactitud. Si por el contrario, se requiere conocer los resultados de forma más precisa, es recomendable utilizar el método de Runge-Kutta de cuarto orden, aunque programarlo tome más tiempo.

Cualquiera de estos métodos mejora si se reduce el tamaño de paso, pero es importante considerar que reducirlo demasiado puede hacer que el programa tarde más en ejecutarse. Si se dispone de una computadora con procesador avanzado, esto no debería ser un inconveniente.

7.4. Ecuaciones diferenciales parciales.

Hasta ahora se han resuelto ecuaciones diferenciales ordinarias, es decir, que sólo contienen derivadas respecto a una sola variable independiente. Ahora se estudiará un caso distinto.

$$k\frac{\partial^2 T}{\partial x^2} = \frac{\partial T}{\partial t} \tag{7-14}$$

La ecuación (7-14) es conocida como la ecuación de conducción de calor. Como puede verse, es una ecuación diferencial parcial y por consiguiente, no es posible aplicar los métodos ya vistos. Para resolverla, se representa la segunda derivada como una diferencia finita centrada:

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}}{(\Delta \mathrm{x})^2} \tag{7-15}$$

Ahora, la primera derivada respecto al tiempo se aproxima por una diferencia finita hacia adelante:

$$\frac{\partial T}{\partial t} = \frac{T_{i,j+1} - T_{i,j}}{\Delta t} \tag{7-16}$$

Y sustituyendo (7-15) y (7-16) en (7-14) se llega a:

$$k\frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}}{(\Delta x)^2} = \frac{T_{i,j+1} - T_{i,j}}{\Delta t} \tag{7-17}$$

7.4.1. Método explícito.

Para resolver la ecuación de conducción de calor por el método explícito se despeja $T_{i,\,j+1}$ de la ecuación (7-17):

$$T_{i,j+1} = T_{i,j} + k\,\Delta t\left(\frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}}{(\Delta x)^2}\right) \tag{7-18}$$

Sea $\boldsymbol{\lambda} = k\Delta t / (\Delta x)^2$:

$$T_{i,j+1} = T_{i,j} + \lambda(T_{i-1,j} - 2T_{i,j} + T_{i+1,j}) \tag{7-19}$$

El método explícito utiliza la información del sistema en un tiempo y espacio inicial para calcular los valores de los nodos adyacentes mediante la ecuación (7-19).

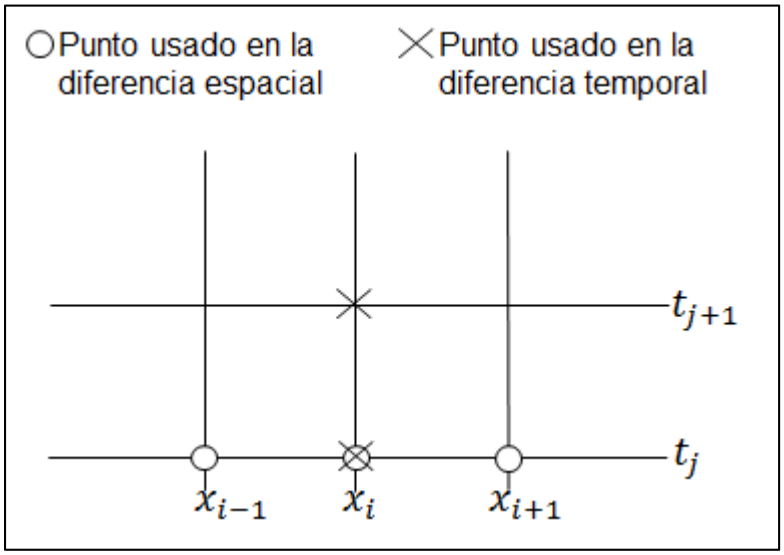


Figura 7.10. Método explícito.

7.4.2. Método de Crank-Nicolson.

Otro método para resolver la ecuación de conducción de calor es el de Crank-Nicolson [18], el cual tiene una exactitud de segundo orden. Para lograr tal exactitud, se desarrollan aproximaciones por diferencias en el punto medio del incremento del tiempo.

$$\frac{\partial T}{\partial t} \cong \frac{T_{i,j+1} - T_{i,j}}{\Delta t} \tag{7-20}$$

$$\frac{\partial^2 T}{\partial x^2} \cong \frac{1}{2}\left[\frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}}{(\Delta x)^2} + \frac{T_{i-1,j+1} - 2T_{i,j+1} + T_{i+1,j+1}}{(\Delta x)^2}\right] \tag{7-21}$$

Sustituyendo (7-20) y (7-21) en (7-14) se llega a:

$$\frac{T_{i,j+1} - T_{i,j}}{\Delta t} = \frac{k}{2}\left[\frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}}{(\Delta x)^2} + \frac{T_{i-1,j+1} - 2T_{i,j+1} + T_{i+1,j+1}}{(\Delta x)^2}\right] \tag{7-22}$$

Que se puede rearreglar como:

$$T_{i,j+1} - T_{i,j} = \frac{\lambda}{2}\left[T_{i-1,j} - 2T_{i,j} + T_{i+1,j} + T_{i-1,j+1} - 2T_{i,j+1} + T_{i+1,j+1}\right] \tag{7-23}$$

La ecuación (7-3) es el algoritmo de Crank-Nicolson. Si lo aplicamos a los nodos (1,0) y (1,1) tenemos:

$$T_{1,1} - T_{1,0} = \frac{\lambda}{2}\left[T_{0,0} - 2T_{1,0} + T_{2,0} + T_{0,1} - 2T_{1,1} + T_{2,1}\right] \tag{7-24}$$

Rearreglando:

$$(1+\lambda)T_{1,1} - \frac{\lambda}{2}T_{2,1} = (1-\lambda)T_{1,0} + \frac{\lambda}{2}\left[T_{0,0} + T_{0,1} + T_{2,0}\right] \tag{7-24}$$

Aplicando de nuevo la ecuación (7-23) a los nodos (2,0) y (2,1) se tiene:

$$T_{2,1} - T_{2,0} = \frac{\lambda}{2}\left[T_{1,0} - 2T_{2,0} + T_{3,0} + T_{1,1} - 2T_{2,1} + T_{3,1}\right] \tag{7-25}$$

Los términos $T_{1,0}$, $T_{2,0}$ y $T_{3,0}$ ya están dados por la condición inicial. Las incógnitas a determinar son $T_{1,1}$, $T_{2,1}$ y $T_{3,1}$.

Rearreglando (7-25):

$$-\frac{\lambda}{2}T_{1,1} + (1+\lambda)T_{2,1} - \frac{\lambda}{2}T_{3,1} = \frac{\lambda}{2}(T_{1,0} + T_{3,0}) + (1-\lambda)T_{2,0} \qquad (7\text{-}26)$$

Aplicando la ecuación (7-23) ahora a los nodos (2,1) y (3,1):

$$T_{3,1} - T_{3,0} = \frac{\lambda}{2}[T_{2,0} - 2T_{3,0} + T_{4,0} + T_{2,1} - 2T_{3,1} + T_{4,1}] \qquad (7\text{-}27)$$

Rearreglando (7-27):

$$-\frac{\lambda}{2}T_{2,1} + (1+\lambda)T_{3,1} = \frac{\lambda}{2}(T_{2,0} + T_{4,0} + T_{4,1}) + (1-\lambda)T_{3,0} \qquad (7\text{-}28)$$

El método de Crank-Nicolson consiste en resolver el sistema formado por las ecuaciones (7-24), (7-26) y (7-28):

$$\begin{aligned}
(1+\lambda)T_{1,1} - \frac{\lambda}{2}T_{2,1} &= (1-\lambda)T_{1,0} + \tfrac{\lambda}{2}\left[T_{0,0} + T_{0,1} + T_{2,0}\right] \\
-\frac{\lambda}{2}T_{1,1} + (1+\lambda)T_{2,1} - \frac{\lambda}{2}T_{3,1} &= \tfrac{\lambda}{2}(T_{1,0} + T_{3,0}) + (1-\lambda)T_{2,0} \\
-\frac{\lambda}{2}T_{2,1} + (1+\lambda)T_{3,1} &= \tfrac{\lambda}{2}(T_{2,0} + T_{4,0} + T_{4,1}) + (1-\lambda)T_{3,0}
\end{aligned}$$

Una vez que se resuelve el sistema, este procedimiento se sigue aplicando para los siguientes nodos hasta calcular las temperaturas de toda la malla.

7.5. Problema de ecuaciones diferenciales parciales.

Una barra metálica con k= 1 ft^2/h y longitud de 1 ft es calentada en sus extremos. Calcule las temperaturas que alcanza cada 0.25 ft después de 1 hora bajo las siguientes condiciones:

$$T(x,0) = 20°C$$
$$T(0,t) = 50°C$$
$$T(1,t) = 100°C$$

7.5.1. Solución por el método explícito.

El programa 7.6 resuelve este problema usando el método explícito.

```
F[x_] := 20; (* Cond. inicial *)
CF1 = 50;
CF2 = 100;
k = 1; L = 1;
tmax = 1; subt = 100;
xmax = 1; subx = 4;
x = 0; t = 0; (* Valores iniciales *)
Δx = xmax / subx;
Δt = tmax / subt;
λ = k * Δt / Δx^2;

(* Temperaturas al tiempo t = 0 *)
T[0, 0] = (F[0] + CF1) / 2;
T[subx, 0] = (F[subx] + CF2) / 2;

For[i = 1, i < subx, i++,
 T[i, 0] = F[x];
 x = x + Δx;
]
(* Fin temperaturas a t = 0 *)

(* Temperaturas en x=0 y x=xmax *)
For[j = 1, j ≤ subt, j++,
 T[0, j] = CF1;
 T[subx, j] = CF2;
]
(* Fin. Temperaturas en x=0 y x=xmax *)

For[j = 0, j < subt, j++,
 For[i = 1, i < subx, i++,
  T[i, j + 1] = λ * (T[i - 1, j] - 2 * T[i, j] + T[i + 1, j]) + T[i, j];
 ]
]
```

Programa 7.6. Solución del problema por el método explícito.

Luego de correr este programa, se obtiene una matriz de 7 columnas y 101 filas con los resultados. A continuación se muestra una parte de ella:

		x			
	0.25	0.5	0.75	1.0	
i	t		T		
0	0.	35.	20.	20.	20.	60.
1	0.01	50.	22.4	20.	26.4	100.
2	0.02	50.	26.432	21.408	37.152	100.
3	0.03	50.	29.399	24.7309	44.6886	100.
4	0.04	50.	31.9483	28.671	50.3452	100.
5	0.05	50.	34.3122	32.6633	54.8221	100.
6	0.06	50.	36.5584	36.4725	58.5052	100.
7	0.07	50.	38.6953	40.0115	61.6191	100.
8	0.08	50.	40.7147	43.2581	64.3028	100.
9	0.09	50.	42.6073	46.2183	66.6472	100.
10	0.1	50.	44.3679	48.9092	68.715	100.
11	0.11	50.	45.9956	51.3515	70.5517	100.
12	0.12	50.	47.4933	53.5666	72.1914	100.
13	0.13	50.	48.8661	55.5748	73.6608	100.
14	0.14	50.	50.1209	57.3952	74.9813	100.
15	0.15	50.	51.2654	59.0451	76.1705	100.
16	0.16	50.	52.3077	60.5404	77.2432	100.
17	0.17	50.	53.2557	61.8956	78.2118	100.
18	0.18	50.	54.1172	63.1238	79.0873	100.
19	0.19	50.	54.8995	64.2369	79.8792	100.
20	0.2	50.	55.6096	65.2457	80.5958	100.
21	0.21	50.	56.2538	66.1599	81.2444	100.
22	0.22	50.	56.8382	66.9885	81.8318	100.

90	0.9	50.	62.493	74.9901	87.493	100.
91	0.91	50.	62.4936	74.991	87.4936	100.
92	0.92	50.	62.4942	74.9918	87.4942	100.
93	0.93	50.	62.4948	74.9926	87.4948	100.
94	0.94	50.	62.4953	74.9933	87.4953	100.
95	0.95	50.	62.4957	74.9939	87.4957	100.
96	0.96	50.	62.4961	74.9945	87.4961	100.
97	0.97	50.	62.4965	74.995	87.4965	100.
98	0.98	50.	62.4968	74.9955	87.4968	100.
99	0.99	50.	62.4971	74.9959	87.4971	100.
100	1.	50.	62.4974	74.9963	87.4974	100.

7.5.2. Solución por el método de Crank-Nicolson.

El programa 7.7 llega a la solución del problema por el método de Crank-Nicolson.

```
F[x_] := 20; (* Cond. inicial *)
CF1 = 50;
CF2 = 100;
k = 1; L = 1;
tmax = 1; subt = 100;
xmax = 1; subx = 4;
x = 0; t = 0; (* Valores iniciales *)
Δx = xmax / subx;
Δt = tmax / subt;
λ = k * Δt / Δx^2;

For[i = 1, i < subx, i++,
 T[i, 0] = F[x];
 x = x + Δx;
]
(* Fin temperaturas a t = 0 *)

(* Temperaturas en x=0 y x=xmax *)
For[j = 1, j ≤ subt, j++,
 T[0, j] = CF1;
 T[subx, j] = CF2;
]
(* Fin. Temperaturas en x=0 y x=xmax *)


For[j = 0, j < subt, j++,


 A = {{1 + λ, -λ/2, 0}, {-λ/2, 1 + λ, -λ/2}, {0, -λ/2, 1 + λ}};
```

Programa 7.7. Solución del problema por el método de Crank-Nicolson. (Parte 1 de 2).

```
B = {{1 - λ, λ/2, 0}, {λ/2, 1 - λ, λ/2}, {0, λ/2, 1 - λ}};

tt = {{T[1, j]}, {T[2, j]}, {T[3, j]}};

c = {{λ/2 * (T[0, j] + T[0, j + 1])}, {0}, {λ/2 * (T[subx, j] + T[subx, j + 1])}};

Resultado1 = B.tt + c;
(*Print[MatrixForm[N[Resultado1]]];*)
Resultado2 = LinearSolve[A, Resultado1];
(*Print[MatrixForm[N[Resultado2]]];*)

T[1, j + 1] = Resultado2[[1]][[1]];
T[2, j + 1] = Resultado2[[2]][[1]];
T[3, j + 1] = Resultado2[[3]][[1]];
]

Resultados = Table[{i, N[i * Δt], N[T[0, i]], N[T[1, i]],
    N[T[2, i]], N[T[3, i]], N[T[4, i]]}, {i, 0, 100}];
Print["\n\t\t  \t  x\n  \t0.25\t0.5\t0.75\t1.0\n  i\tt
_____________________________________"];
Print[MatrixForm[Resultados]];
```

Programa 7.7. Solución del problema por el método de Crank-Nicolson. (Parte 2 de 2).

En la siguiente página se muestra una parte de la matriz de resultados obtenida por medio de este método.

		x			
	0.25	0.5	0.75	1.0	
i	t				
0	0.	35.	20.	20.	20.	60.
1	0.01	50.	23.1581	20.7923	28.3305	100.
2	0.02	50.	26.6879	23.0225	37.33	100.
3	0.03	50.	29.6154	26.1785	44.2183	100.
4	0.04	50.	32.1955	29.6945	49.6666	100.
5	0.05	50.	34.5524	33.2625	54.1004	100.
6	0.06	50.	36.7437	36.7206	57.7957	100.
7	0.07	50.	38.7943	39.9887	60.9355	100.
8	0.08	50.	40.7146	43.0324	63.6444	100.
9	0.09	50.	42.5088	45.8426	66.0097	100.
10	0.1	50.	44.1799	48.4235	68.0943	100.
11	0.11	50.	45.7309	50.786	69.9448	100.
12	0.12	50.	47.1659	52.9442	71.5966	100.
13	0.13	50.	48.4896	54.9133	73.0774	100.
14	0.14	50.	49.7078	56.7086	74.4093	100.
15	0.15	50.	50.8265	58.3444	75.6104	100.
16	0.16	50.	51.8522	59.8346	76.6957	100.
17	0.17	50.	52.7914	61.1917	77.678	100.
18	0.18	50.	53.6503	62.4277	78.5682	100.
19	0.19	50.	54.4351	63.5531	79.3756	100.
20	0.2	50.	55.1517	64.5778	80.1086	100.
21	0.21	50.	55.8056	65.5108	80.7744	100.
22	0.22	50.	56.4021	66.3604	81.3795	100.
23	0.23	50.	56.9459	67.1339	81.9296	100.
24	0.24	50.	57.4417	67.8381	82.4298	100.
90	0.9	50.	62.4896	74.9853	87.4896	100.
91	0.91	50.	62.4906	74.9866	87.4906	100.
92	0.92	50.	62.4914	74.9878	87.4914	100.
93	0.93	50.	62.4922	74.9889	87.4922	100.
94	0.94	50.	62.4929	74.9899	87.4929	100.
95	0.95	50.	62.4935	74.9908	87.4935	100.
96	0.96	50.	62.4941	74.9916	87.4941	100.
97	0.97	50.	62.4946	74.9924	87.4946	100.
98	0.98	50.	62.4951	74.9931	87.4951	100.
99	0.99	50.	62.4955	74.9937	87.4955	100.
100	1.	50.	62.4959	74.9943	87.4959	100.

7.6. Comparación de los métodos para resolver ecuaciones diferenciales parciales.

Para comprobar si el método obtuvo resultados coherentes, podemos calcular la temperatura que debió alcanzarse en el estado estacionario. Esto lo hacemos utilizando la ecuación de la línea recta.

$$T = 50 + 50x$$

$$T = 50 + 50 * 0.25 = 62.5°C$$

La temperatura que debió obtenerse en el estado estacionario es de 62.5°C. Viendo las matrices de resultados, se puede ver que con ambos métodos se consiguió ese resultado. El valor obtenido con el método implícito fue de 62.4974°C, mientras que con el método de Crank-Nicolson fue 62.4959°C.

Para examinar qué tan exactos fueron, se pueden analizar los cálculos de ambos métodos en $T_{2,1}$, que es justo en el centro de la barra en la primera iteración. Vemos que los resultados son:

Método	Temperatura $T_{2,1}$
Explícito	20°C
Crank-Nicolson	20.7923°C

Con el método explícito, la temperatura en el nodo (2,1) sigue siendo la misma que en la condición inicial. Por otro lado, el resultado de Crank-Nicolson es más coherente con lo que en realidad sucede, pues se sabe que al calentar la barra, la temperatura en el centro también debe aumentar.

Ambos métodos obtienen resultados satisfactorios. El implícito es fácil de programar aunque menos preciso. El de Crank-Nicolson es conveniente si se requiere una exactitud mayor, aunque escribir el código sea más difícil.

Si se dispone de un paquete matemático potente como Wolfram Mathematica, la programación de Crank-Nicolson es menos laboriosa que en otros lenguajes.

Conclusiones.

El objetivo del presente proyecto fue proporcionar información que facilitara la resolución de problemas de ingeniería química utilizando métodos numéricos. Para esto, nos apoyamos en una invaluable herramienta que la Universidad pone a nuestra disposición: Mathematica 10.0. Este costoso software hoy en día es gratuito para cualquier alumno de la UNAM y a la fecha no está siendo aprovechado como debería.

A lo largo de siete capítulos se proporcionó información útil para resolver problemas típicos que solemos encontrar quienes aspiramos a ser ingenieros químicos. Algunas de las materias que se abordaron utilizando técnicas de análisis numérico y programación fueron:

- Química General.
- Termodinámica.
- Balances de materia.
- Fenómenos de transporte.
- Cinética Química.
- Ingeniería de reactores.

En total se presentaron 21 métodos numéricos, lo cual abarca casi la totalidad del curso que se imparte en la Facultad de Química. Utilizando los 21 métodos mencionados, se resolvió un total de 12 problemas de las disciplinas mencionadas con ayuda de Mathematica.

En cada capítulo se analizan los resultados obtenidos para determinar cuál de los métodos estudiados es más eficiente. Además de eso, se plantea bajo qué circunstancias conviene utilizar uno u otro.

Con base en esto, se puede concluir que el objetivo planteado al inicio de este trabajo se cumple satisfactoriamente. Los programas aquí escritos, además de proporcionar soluciones con exactitud y eficiencia, también pueden considerarse de utilidad para la comunidad de la Facultad de Química, ya que cualquier persona los puede reusar y modificar de acuerdo con sus necesidades. Por otro lado, quien esté llevando la materia de métodos numéricos encontrará aquí información para complementar y aprovechar el curso.

El futuro de la ciencia está cada vez más ligado a la computación. Por este motivo, la falta de conocimiento en esta disciplina no debe prevalecer.

Apéndice.

Método numérico	Fórmula (s)
Interpolación de Newton	$p_n(x) = \sum_{k=0}^{n} a_k \prod_{i=0}^{k-1} (x - x_i)$
Interpolación de Lagrange	$L_k(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{(x - x_i)}{x_k - x_i}$ $p_n(x) = \sum_{i=0}^{n} L_i(x)\, f(x_i)$
Interpolación de Lagrange (bidimensional)	$L_i(x) = \prod_{j=0}^{n-1} \frac{(x - x_j)}{x_i - x_j}$ $L_i(y) = \prod_{j=0}^{n-1} \frac{(y - y_j)}{y_i - y_j}$ $p_n(x,y) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} [L_i(x) * L_j(y) * f(x_i, y_j)]$
Punto fijo	$x_{i+1} = g(x_i)$
Aitken	$x_{i+1} = g(x_i)$ $x_4 = x_3 - \frac{(x_3 - x_2)^2}{(x_3 - 2x_2 + x_1)}$
Newton-Raphson	$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$
Secante	$x_{i+1} = x_i - \frac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)}$
Wegstein	$q = 1 + \frac{1}{W - 1}$ $W = \frac{g(x_i) - g(x_{i-1})}{x_i - x_{i-1}}$ $x_{i+1} = x_i * q + (1 - q) * g(x_i)$

Método numérico	Fórmula (s)
Bisección	$x_M = \frac{x_I + x_D}{2}$
Posición falsa	$x_M = x_D - \frac{f(x_D)(x_I - x_D)}{f(x_I) - f(x_D)}$
Newton-Raphson multivariable	$x_{n+1} = x_n - J_f^{-1}(x_n) f(x_n)$
Broyden	$(A^k)^{-1} = (A^{k-1})^{-1} + \frac{[\Delta x^k - (A^{k-1})^{-1}\Delta f^k](\Delta x^k)^T (A^{k-1})^{-1}}{(\Delta x^k)^T (A^{k-1})^{-1}\Delta f^k}$
Regla del trapecio	$I \cong \frac{h}{2}\left[f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)\right]$
Regla de Simpson	$I \cong \frac{h}{3}\left[f(x_0) + 4\sum_{\substack{i=1 \\ \Delta i=2}}^{n-1} f(x_i) + 2\sum_{\substack{i=2 \\ \Delta i=2}}^{n-2} f(x_i) + f(x_n)\right]$
Cuadratura de Gauss	$x = \frac{(b+a)+(b-a)z}{2}$ $dx = \frac{b-a}{2}dz$ $I \cong w_1 F(z_1) + w_2 F(z_2) + \cdots + w_n F(z_n)$
Euler	$y_{i+1} = y_i + h\, f(x_i, y_i)$
Runge-Kutta de cuarto orden	$k_1 = f(x_i, y_i)$ $k_2 = f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1 h)$ $k_3 = f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_2 h)$ $k_4 = f(x_i + h, y_i + k_3 h)$ $y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h$
Explícito	$T_{i,j+1} = T_{i,j} + \lambda(T_{i-1,j} - 2T_{i,j} + T_{i+1,j})$
Crank-Nicolson	$T_{i,j+1} - T_{i,j} = \frac{\lambda}{2}[T_{i-1,j} - 2T_{i,j} + T_{i+1,j} + T_{i-1,j+1}$ $- 2T_{i,j+1} + T_{i+1,j+1}]$

Bibliografía.

[1] Chapra, Steven; Canale, Raymond (2010), Métodos numéricos para ingenieros, 6ta edición, México, McGraw Hill.

[2] Nieves Hurtado, Antonio; Domínguez Sánchez (2002), Federico, Métodos numéricos aplicados a la ingeniería, 2da edición, México, Grupo Editorial Patria.

[3] Burden, Richard L.; Faires, J. Douglas (1985), Numerical Analysis, 3rd ed, PWS Publishers.

[4] B. Carnahan, H. A. Lither and J. O. Wilkes (1969), Applied numerical methods, 1st ed, Wiley, New York.

[5] Rincón Orta, César A.; Granados Aguilar, A. Salvador; Fautsch Tapia, Eugenio L.; Rubín Rivero, Susana; Vázquez Islas, Manuel; Díaz García, Antonio F. (2014), Álgebra Superior, 1ª edición, México, McGraw-Hill.

[6] Eberhart, J.G. (1986), "Solving equations by successive substitution: The problems of divergence and slow convergence", Journal of Chemical Education, 63 (7), p 576, DOI: 10.1021/ed063p576.

[7] Perry, R.H.; Green, D (1986), Manual del ingeniero químico, 6ta edición, México, McGraw-Hill.

[8] Iriarte V. Valderrama, Rafael (1990), Métodos numéricos, 1ª edición, México, Trillas.

[9] Smith, J.M.; Van Ness, H.M.; Abbot, M.M (2007), Introducción a la termodinámica en Ingeniería Química, 7ª edición, México, McGraw-Hill.

[10] Broyden, C. G. (1965). "A Class of Methods for Solving Nonlinear Simultaneous Equations". Mathematics of Computation (American Mathematical Society) 19 (92): 577–593, doi:10.2307/2003941.

[11] Cutlip, Michael B.; Shacham, Mordechai (2008), Resolución de problemas en Ingeniería Química y Bioquímica con Polymath, Excel y Matlab, 2da edición, Madrid (España), Pearson Educación.

[12] Ryan, Mark (2014), Calculus for Dummies, 2nd edition, Hoboken (New Jersey), John Wiley & Sons, Inc.

[13] Walas, Stanley M. (1995), Chemical Reaction Engineering Handbook of Solved Problems, Amsterdam, Gordon and Breach Publishers.

[14] Fogler, H. Scott, Elementos de ingeniería de las reacciones químicas, 4ta edición, México, Pearson Prentice Hall.

[15] Levine, Ira N. (2005), Problemas de Fisicoquímica, 5ta edición, Aravaca (Madrid), McGraw-Hill.

[16] Larson, Ron; Edwards, Bruce H. (2010, Cálculo, 9na edición, México, McGraw-Hill.

[17] Carmona Jover, Isabel (2011), Ecuaciones diferenciales, 5ta edición, México, Pearson Educación.

[18] Crank, John; Nicolson, Phyllis (1947). "A practical method for numerical evaluation of solutions of partial differential equations of the heat conduction type". Proc. Camb. Phil. Soc. 43 (1): 50–67. doi:10.1007/BF02127704

[19] Ramírez Delgado, J. Emmanuel (2014), Unidad didáctica para el curso de métodos numéricos, usando la herramienta Visual Basic de Excel. Tesis de licenciatura, Universidad Nacional Autónoma de México, Facultad de Química.

[20] Rodríguez Sotelo, Carlos D. (2015), Programación de métodos numéricos en Python aplicados a problemas de ingeniería química. Tesis de licenciatura, Universidad Nacional Autónoma de México, Facultad de Química.