UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Material de apoyo a la docencia para la materia de Programación Avanzada

MATERIAL DIDÁCTICO

Que para obtener el título de

Ingeniero Petrolero

P R E S E N T A

Yaret Marina Hernández Montoya

ASESORA DE MATERIAL DIDÁCTICO

Ing. Adriana Alejandra Enríquez Solís

Ciudad Universitaria, Cd. Mx., 2019

Índice

Índice de figuras

Índice de tablas

Resumen

Al buscar dar solución a problemas matemáticos, en ocasiones resulta complicado de llegar al resultado deseado, ya que en ciertos casos no hay una manera directa para resolverlos, es decir que al realizar las operaciones fundamentales sobre una función para dar solución resulta de cierta forma difícil o complicado, por lo que es necesario recurrir a métodos numéricos, llamados así porque, consisten principalmente en realizar una sucesión de operaciones numéricas para encontrar un valor numérico, que si bien no es la solución exacta al problema planteado, es la mejor aproximación razonablemente buena.

Es por eso que este trabajo tiene su fundamento en los Métodos Numéricos, utilizados para dar solución a diferentes problemáticas las cuales pueden ser representadas en funciones, ecuaciones, sistemas matriciales, etc., que mediante la implementación de uno o más métodos buscan dar la mejor aproximación al valor real buscado. También se tiene contemplado que para llegar a las soluciones y para ejecutar cualquier método numérico, resulta más fácil si se recurre a otro tipo de herramientas, como puede ser la elaboración de un algoritmo, ya que facilita su entendimiento y resulta útil cuando se busca llevarlo a algún lenguaje de programación para la elaboración del pseudocódigo. Estos lenguajes son herramientas que sirven para llegar más fácil a la solución, ya que, al realizar cierto número de iteraciones, reduce el tiempo de trabajo y mejora la aproximación del resultado.

El principal objetivo de este Cuaderno de ejercicios es que los alumnos que inician un curso de Programación en la Carrera de Ingeniería Petrolera tengan un apoyo para un fácil y mejor entendimiento de algunos de los métodos numéricos, los cuales les servirá para implementar en varios ejercicios de diferentes materias durante su desarrollo académico. Este trabajo pretende apoyar al estudiante y/o profesor de la asignatura de Programación Avanzada en los fundamentos del tema, mediante el desarrollo de la teoría y algoritmos de solución para los métodos abordados. Los ejercicios que se presentan están dirigidos a la ingeniería, y que, implementando el algoritmo propuesto, resulte más fácil la programación de este para poder darle solución.

Capítulo 1
Fundamentos básicos para el uso de FORTRAN

1.1 Introducción a FORTRAN

FORTRAN es un lenguaje de programación comúnmente empleado en aplicaciones de ingeniería y matemáticas, por lo que resulta importante tener las bases para crear, leer y modificar un código creado en dicho lenguaje.

Para realizar exitosamente un programa en FORTRAN es necesario saber cómo estructurarlo, ya que resulta más simple la programación si se elabora previamente un algoritmo acerca del proceso que se desea desarrollar. Por ello, se partirá de la definición de algoritmo hacia el desarrollo de los códigos subsecuentes.

1.1.1 Algoritmo

Un algoritmo es una secuencia finita de instrucciones lógicas, realizables, no ambiguas, cuya ejecución da una solución a un problema específico en un tiempo finito. Es **finito, claro, eficiente y eficaz**.

Los algoritmos pueden ser de dos tipos:

i. Representación gráfica de las operaciones que deben ser realizadas.

ii. Representación descriptiva de las operaciones que se deben realizar.

Las etapas por las que pasa un algoritmo pueden ser tres:

1. Análisis del problema: datos iniciales, datos de entrada, restricciones y salida esperada.
2. Construcción del algoritmo: pasos a seguir para dar solución al problema.
3. Verificación del algoritmo: prueba de escritorio.

Existen diferentes formas para diseñar un algoritmo, **Fig. 1. 1**, siendo las más empleadas el pseudocódigo, mezcla de palabras coloquiales con las utilizadas en un lenguaje de programación, y el diagrama de flujo, que se ayuda de un conjunto de símbolos

gráficos para representar cada instrucción. De esta forma, una vez que se ha generado un algoritmo, puede ser codificado utilizando un lenguaje de programación respetando la sintaxis y reglas establecidas para éste, dando como resultado un programa.

Un lenguaje de programación es un software integrado por un conjunto de reglas, símbolos y palabras especiales que permiten editar, compilar y ejecutar programas. Antes, de ser ejecutado un programa se lleva a cabo una revisión de sintaxis; los lenguajes que revisan y ejecutan instrucción por instrucción se denominan intérpretes, en cambio, los que ejecutan todas las instrucciones después de revisadas son los compiladores.

Cuando se habla de sintaxis se refiere al conjunto de reglas que gobiernan una construcción o formación de sentencias (instrucciones) válidas en un lenguaje; la semántica es el conjunto de reglas que les dan el significado; el vocabulario se refiere al conjunto de símbolos (letras, dígitos, caracteres y palabras clave); y las reglas sintácticas permiten reconocer si una cadena o serie de símbolos es correcta gramaticalmente.

a)

```
INICIO
a: Entero
b: Entero
ESCRIBIR "Valor del primer número"
LEER a
ESCRIBIR "Valor del segundo número"
LEER b
SI a > b entonces
ESCRIBIR "El número a es mayor a b"
FIN DEL SI
EN CASO CONTRARIO
   ESCRIBIR "El número b es mayor o
igual a a"
FIN DE EN CASO CONTRARIO
FIN
```

b)

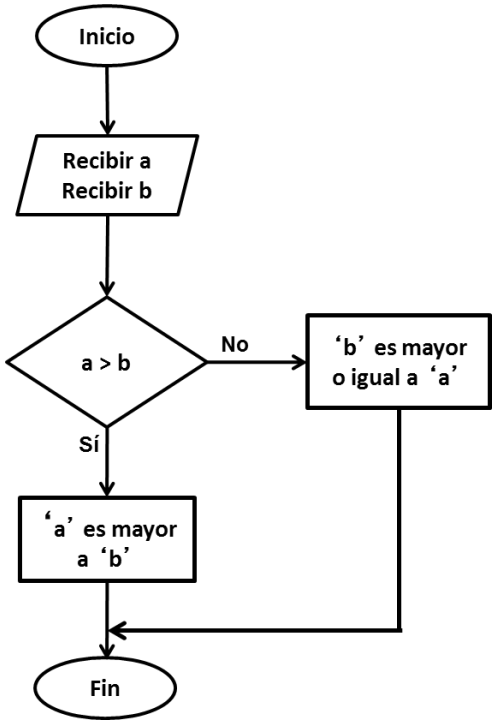


Fig. 1. 10

Representación de un algoritmo: a) seudocódigo y b) diagrama de flujo.

1.1.2 Tipos de lenguajes de programación

De acuerdo con el tipo de comunicación que se establece con el hardware, los lenguajes de programación pueden clasificarse en dos grupos: de bajo nivel y de alto nivel. Los primeros tienen la característica de que el procesador de la computadora solamente puede ejecutar instrucciones simples o de máquina, que se indican en lenguaje binario.

Por otro lado, los lenguajes de alto nivel son más generales, con lo cual pueden escribirse algoritmos que utilizan palabras y reglas entendibles para el ser humano, y ejecutables en un mayor número de computadoras; tienen la enorme ventaja de permitir la programación sin la necesidad de conocer el funcionamiento interno de la máquina. Estos lenguajes poseen una potencia considerable en cuanto a sus instrucciones, una sentencia de alto nivel equivale a varias en el lenguaje máquina.

1.1.3 Compilar y ejecutar programas

Para realizar un programa en FORTRAN puede escribirse en un editor de texto cualquiera (Vi, emacs, notepad ++, Gedit, u otro similar), y este debe de llevar el sufijo .f o bien .fxx, donde –xx depende de la versión con la que se trabaje. Este archivo recibe el nombre de código fuente.

Una vez que se encuentra hecho el programa este debe ser compilado. En este proceso el código debe ser leído por un programa llamado compilador, que cumple la función de traducir las instrucciones estructuradas para resolver un problema al leguaje máquina y crear un nuevo archivo con el programa ya ejecutable.

Existen diferentes compiladores para FORTRAN (GFortran, Force, PLATO, G95, PathScale, etc.), cabe mencionar que para este trabajo se utilizó PLATO IDE con la versión de FORTRAN 95.

1.2 Elementos básicos de un programa en FORTRAN

Un programa en FORTRAN se constituye por los siguientes elementos básicos:

1. Nombre: aunque opcional, ayuda a tener una mejor organización.

2. Declaración: donde se definen el número y tipo de variables a utilizar.
3. Cuerpo: instrucciones a ejecutar (código), éstas son estructuradas en el orden en el que suelen aparecer. El programa termina con el comando END.
4. Subprogramas: cuando dentro del cuerpo del programa se llama a subprogramas que realizan tareas específicas.

La estructura de un programa en FORTAN **Fig 1.2,** tiene las siguientes características:

1. Las instrucciones se escriben en líneas de máximo 132 caracteres, incluidos los espacios.
2. Los espacios en blanco al principio de una línea son ignorados, pero ayudan a mejorar visualmente la estructuración del programa.
3. Cuando se exceden los 132 caracteres, las instrucciones pueden continuarse en la línea siguiente, utilizando el símbolo **&** al final de la línea a continuar y al inicio de la siguiente.
4. El símbolo de admiración (**!**) sirve para incluir comentarios dentro del programa, mismo que son ignorados al momento de la compilación.
5. El símbolo punto y coma (**;**) sirve para poner varias instrucciones en una misma línea.
6. El programa no distingue entre mayúsculas y minúsculas, también ignora las líneas en blanco.
7. Los objetos del programa, como unidades programáticas (conjunto de instrucciones que se encargan de realizar tareas específicas), deben ser nombradas utilizando una composición de 1 a 31 caracteres alfanuméricos, siendo el primero una letra, los nombres dados a los objetos deben ser distintos a los nombres reservados para las instrucciones.

```
<Identificación unidad programática > !Tipo y Nombre de la Unidad
        :
        : Declaraciones de los objetos utilizados en el programa ! Instrucciones no ejecutables
        :
        : Cuerpo del programa. Operaciones matemáticas, lectura y escritura de datos en la
                pantalla y, archivos, llamadas a otras unidades, etc. ! Instrucciones ejecutables
        :
end <Unidad Programática> !Final y Nombre de la Unidad
```

Fig. 1. 11
Estructura de una unidad programática.

1.2.1 Tipos de datos

En FORTRAN se trabaja con constantes y variables, que representan un tipo específico de dato que indica al programa la forma en que se almacenará y manejará, estos se muestran en la **Tabla 1. 1**.

Tabla 1. 13
Tipos de datos manejados en FORTRAN

Tipo	Descripción	Ejemplo
Character	Cadenas de uno o más caracteres.	'fortran'
Integer	Números enteros, que pueden tomar todos los valores positivos o negativos, se indican como números sin punto decimal.	-251
Logical	Valores lógicos o booleanos, que toman solamente uno de los dos valores; falso (.false.) o verdadero (.true.).	.true.
Real	Números reales positivos o negativos; se indican con punto decimal, y de ser necesario un exponente de potencia 10.	123. 1.23e06
Complex	Números complejos, con su parte real e imaginaria reales.	(-1.,2.e-3)

1.2.2 Declaración de variables

FORTRAN maneja variables del tipo implícitas (propias de FORTRAN), si en un código aparecen nombres que no se han definido en instrucciones específicas, el tipo de la variable depende de su primera letra:

1. I, j, k, l, m, n representan variables enteras.
2. Las demás letras representan variables reales.

Este carácter implícito de las variables puede ser modificado, a través de la instrucción *implicit*, cuya sintaxis es la siguiente:

Implicit<tipo>(<lista_1>,<lista_2>, …, <lista_k>)

La cual sirve para modificar una o más variables. Para evitar este tipo de declaraciones implícitas es más recomendable que el usuario declare el tipo de variables enteras o reales que quiera utilizar, esto se hace escribiendo al inicio del programa la instrucción “implicit none”.

Así entonces, la declaración de una o más variables del mismo tipo se hace siguiendo la instrucción de especificación, bajo la siguiente sintaxis.

<tipo> [,<atributo(s)>][::]<variable(s)>[=<valor>]>, el uso de ***::*** *es obligatorio.*

Los atributos posibles son: parameter, save, intent, pointer, target, allocatable, dimension, public, private, external, intrinsic, optional.

En la **Tabla 1. 2** presenta las distintas declaraciones que pueden ser empleadas, de la misma manera se brinda un ejemplo que facilite su entendimiento.

Tabla 1. 14
Tipos de declaraciones

Declaración	Función	Ejemplo
Constantes	Cuando una o más variables toman únicamente un solo valor en el programa.	Integer,parameter:: amax=60, bmax=2*amax
Cadenas de caracteres	Cuando se declaran variables de tipo character de un tamaño especifico o bien si no se conoce el tamaño este puede dejarse de manera que sea ajustado automáticamente.	Character (len=*), parameter:: ciudad='Mexico'
Real	FORTRAN trabaja con dos tipos de real: de simple precisión y doble precisión, aunque también hay compiladores que trabajan con una cuádruple precisión, la sintaxis a utilizar es: *Real(kind=<np>)* Donde *np* toma los valores: 4 (simple), 8 (doble) y 16 (cuádruple). O con letras: simple- e, doble- d y cuádruple- q. Si esta no es especificada, se asume como simple.	Integer, parameter::np=8 real(kind=np)::r=1.e-10_np
Complejo	Como se especificó anteriormente, los números complejos son derivados de los reales se manejan la misma característica en cuanto a la precisión: simple, doble y cuádruple	Integer, parameter:: np=8 Real(kind=4)::c=(1.e0,0.)

1.3 Operaciones elementales

Una diferencia existente entre el lenguaje de FORTRAN y el matemático es el uso del símbolo =, ya que mientras en el lenguaje matemático corresponde a una igualdad, en FORTRAN se emplea para asignar un valor. De esta manera la forma en que se asigna un valor a una variable es bajo la siguiente sintaxis:

<Variable> = <expresión>

Ejemplo:

real :: x= 12

*y=4*x**2*

Donde primero se declara la variable *x* como un número real con un valor fijo de 12 y después a la variable *y* se le asigna un valor dependiente de *x*.

1.3.1 Operaciones aritméticas

Estas operaciones, **Tabla 1. 3**, se realizan sobre valores del tipo entero (integer), real (real) y complejo (complex).

Tabla 1. 15
Operadores aritméticos

Operador	Significado	Ejemplo
**	Exponenciación	5**2 (5^2)
*	Multiplicación	5*A
+	Suma	A+ 6
-	Resta	x-A

Si en una expresión se presentan varias operaciones aritméticas, la computadora las ejecutará en el siguiente orden:

1. Términos entre paréntesis, comenzando por los de más adentro.
2. Potencias.
3. Multiplicaciones y divisiones (de izquierda a derecha).
4. Cambio de signo.
5. Sumas y restas (de izquierda a derecha).

1.3.2 Operaciones de comparación

Corresponden a símbolos que son empleados cuando se desea comparar dos valores, **Tabla 1. 4**; su sintaxis es la siguiente:

<Expresión 1> <operador de comparación> <expresión 2>

Es importante mencionar que los operadores comparativos se aplican una vez que se realizan las expresiones aritméticas o numéricas.

Tabla 1. 16
Operadores comparativos

Operador	Operador equivalente	Significado	Ejemplo
.LT.	<	Menor que	a.LT.6
.LE.	<=	Menor o igual que	4.LE.x
.EQ.	==	Igual a	b.EQ.c
.NE.	/=	Diferente de	d.NE.y
.GT.	>	Mayor que	y.GT.5
.GE.	>=	Mayor o igual que	a.GE.0

1.3.3 Operaciones lógicas

Las instrucciones lógicas, **Tabla 1. 5**, presentan la siguiente sintaxis:

<expresión 1> <operador lógico> <expresión 2>

Donde *<expresión 1>* y *<expresión 2>* son de tipo logical y por consiguiente el resultado es igual.

Los operadores lógicos son evaluados después de las operaciones de comparación, de izquierda a derecha.

Tabla 1. 17
Operadores lógicos

Operador	Significado
.NOT.	Negación
.AND.	Intersección
.OR.	Unión
.EQV.	equivalencia
.NEQVT.	No equivalencia

1.3.4 Funciones intrínsecas

Éstas corresponden a las funciones definidas por el compilador que se está empleando, la sintaxis es la siguiente:

<función intrínsecas>(<argumento>)

La **Tabla 1. 6,** presenta las funciones intrínsecas más comunes y empleadas en FORTRAN.

Tabla 1. 18
Principales funciones intrínsecas

Función	Argumento	Resultado	Descripción
ABS	Real, complex, integer	Real, integer	Valor absoluto
SQRT	Real, complex	Real, complex	Raíz
INT	Real	Integer	Parte entera de un número real
FRACCION	Real	Real	Parte fraccionaria de un número real
COS	Real, complex	Real, complex	Coseno
SEN	Real, complex	Real, complex	Seno
TAN	Real, complex	Real, complex	Tangente
ACOS	Real, complex	Real, complex	Arco coseno
ASIN	Real, complex	Real, complex	Arco seno
ATAN	Real, complex	Real, complex	Arco tangente
EXP	Real, complex	Real, complex	Exponencial
LOG	Real, complex	Real, complex	Logaritmo
LOG10	Real	Real	Logaritmo base 10

1.4 Instrucciones básicas de lectura y escritura de datos

La forma en que se presentan datos en un programa es a travez de la pantalla, mediante un texto que es la forma en que se introducen los datos; es por eso que en este capítulo se presenta la forma en que interactúa el usuario y la máquina para crear el programa en sí.

1.4.1 Escritura sobre pantalla

La sentencia *read* se emplea para la entrada de datos, mientras que *write* se emplea para la salida. El formato o sintaxis a utilizar es la siguiente:

read (núm_unidad, núm_formato) lista_de_variables

write(núm_unidad, núm_formato) lista_de_variables

El número de unidad se puede referir a la salida o entrada estándar o a un archivo.

Una manera más sencilla de escritura para cuando se quiere utilizar el formato estándar en el cual no es necesario emplear la sintaxis anterior, es la siguiente:

read (,*) lista_de_variables ! leerá valores de la entrada estándar y asignará los valores a las variables que aparecen en la lista*

write(,*) lista_de_variables ! escribe la lista en la salida estándar.*

Es importante mencionar que la sentencia *print* es equivalente a un *write* y también se emplea para la salida, su sintaxis es la siguiente:

print,<expresión 1>,...,<expresión n>*

1.4.2 Formatos de representación de datos

Se tiene dos formas de representar los datos, la más simple de ellas corresponde al formato automático el cual está presente en la sintaxis anterior con la aparición del símbolo asterisco (*****). Es importante que el usuario tenga presente que los datos requeridos en un *read* deben estar separados por espacios o comas, mientras que los caracteres estarán delimitados por comillas simples o dobles.

La otra forma de representar los datos es mediante formatos preestablecidos tanto en la lectura como en la escritura, **Tabla 1. 7**, ayudando a una mejor presentación de estos; cabe mencionar que se puede presentar algunas dificultades como lo son:

1. Confeccionar cuadros o tablas presentables.
2. Utilizar datos obtenidos, como datos de entrada.
3. Confusión en las cadenas de caracteres.

La sintaxis para esta presentación es la siguiente:

write(,<etiqueta>)lista de variables<etiqueta>format codigos_de_formato*

Ejemplo:

Write(,900) x,y*
900 format (I4,F8.3)
! x es de tipo entero con cuatro caracteres
! el punto flotante a usar es de ocho caracteres, donde tres son para
! la parte fraccionaria, uno para el punto o sigo, y cuatro para la parte entera
Read (,1)*
1 format (I1)

Tabla 1. 19
Principales especificaciones de formato y control

Especificación	Nombre	Acción
In	Entero	Asignación de un campo de *n* caracteres para representar un entero en notación decimal, justificado a la derecha.
Fn.d	Punto fijo	Asignación de un campo de n caracteres para representar un real en formato punto fijo, con d dígitos para la parte fraccionaria, justificado a la derecha.
En.d	Punto flotante	Asigna un campo de n caracteres para presentar un real en formato de punto flotante, con d dígitos para la parte fraccionaria, justificado a la derecha.
Dn.d	Punto flotante doble precisión	Tiene la misma función que E, pero con doble precisión.
Gn.g	Real	Asigna un campo de n caracteres, con d dígitos; dependiendo del valor: en punto fijo, justificación a la derecha o punto flotante, justificación a la derecha.
An	Texto	Asigna un campo de longitud n, para expresiones de tipo carácter.
A	Texto	Asigna un campo de longitud, longitud de la expresión de tipo carácter.

Tabla 1. 19
Principales especificaciones de formato y control

Especificación	Nombre	Acción
‘ …’	Texto fijo	Asigna el campo de longitud de la cadena de caracteres y escribe en el campo la cadena.
Ln	Lógico	Asigna un campo de n caracteres para valores lógicos, justificación a la derecha.
L	Lógico	Asigna un campo de dos caracteres para valores lógicos, justificación por derecha
X	Espacio	Desplaza el siguiente campo de un espacio
Tn	Tabulador	El siguiente campo comienza en la columna n.
$	Mantención de línea	No cambia la línea al final de una instrucción de lectura o escritura.
/	Cambio de línea	Cambio de línea

1.5 Estructuras de control

Algunas veces es necesario que ciertas instrucciones sean ejecutadas si se requiere cumplir una condición y algunas otras cuando se requiere repetir un grupo de instrucciones sin la necesidad de estar repitiendo una y otra vez la o las instrucciones dentro del código fuente del programa. Para ello FORTRAN maneja estructuras de control que facilitan estos, **Tabla 1. 8**.

Tabla 1. 20
Estructuras de control

Sentencia	Significado	Ejemplo

if (expresión lógica) **then**
instrucción 1
Instrucción 2
else
instrucción 3
Instrucción 4
End if
	Permite someter la ejecución de una o varias instrucciones a una condición.
	Es necesario dar por terminada la sentencia.
		integer n
		n = 1
		10 if (n .lt. 100) then
		n = 2*n
		write (*,*) n
		goto 10
		end if

Tabla 1. 20
Estructuras de control

Sentencia	Significado	Ejemplo

do i= inicio, fin, incremento
Instrución 1
Instrucción 2
…
end do
	Permite repetir la ejecución de un bloque de instrucciones.
	Es necesario dar por terminada la sentencia.
		integer i, cuadrado(10)
		do i=1, 10, 1
		cuadrado(i) = i**2;
		write(*,*) cuadrado(i)
		end do

Do while (expresión lógica) Instrucción 1 Instrucción 2 … **End do**	Limita el número de iteraciones de un bloque de instrucciones Es necesario dar por terminada la sentencia.	I=1 Do while (i<=10) i=i+1 end do
[name:] **select case** (expresión) **Case (selección 1)[name]** Block 1 **Case (selección 2)[name]** Block 2 **Case default [name]** Block 3 **End select [name]**	Permite elegir un bloque de instrucciones, entre varios bloques, en función del valor de la expresión elegida.	character ::ch Read*, ch vocales: select case (ch) Case (‘a’, ‘e’, ‘i’, ‘o’, ‘u’) Print*, ‘vocal’ Case default Print*, ‘consonante u otro caracter’ End select vocales End
go to <etiqueta> … **<Etiqueta>** ... ! se llega a este lugar … donde <etiqueta> es un valor numérico integer de 1 a 99999	Realiza un salto incondicional el cual produce un cambio en el flujo de un programa a otra línea de este. Esta línea de destino debe estar identificada con un número de línea, la cual deberá situarse al inicio de la línea a la que se desea regresar.	A = 2.0 B = sqrt(A) ... **go to 1** ... **1** C = B**2

1.6 Instrucciones básicas de lectura y escritura de datos en archivos

Una manera diferente para presentar datos aparte en un programa sin la necesidad de ingresarlos es mediante el uso de archivos, los cuales se crean en un editor de texto con la terminación .txt, .dat, .out, etc.

Para acceder a estos archivos existen dos formas: la primera, de manera secuencial, en la cual se deben leer todos los registros anteriores al que se desea acceder, la segunda forma es de acceso directo, en donde la instrucción de lectura/escritura indica el número de registro que será tratado.

1.6.1 Manejador lógico

El manejador lógico es especificado en el parámetro conocido como unidad lógica “UNIT” , **Tabla 1. 9**. Para el manejo de archivos existen seis comandos básicos a saber.

Tabla 1. 21
Unidades lógicas

Comando	Descripción
OPEN	Apertura de un archivo de entrada y/o salida con número de unidad especial.
CLOSE	Cierre de un archivo de entrada y/o salida con número de unidad especial.
READ	Lectura de datos de un archivo de entrada y/o salida con número de unidad especial.
WRITE	Escritura de un archivo de entrada y/o salida con número de unidad especial.
REWIND	Posiciona el control de acceso al inicio de un archivo.
BACKSPACE	Retrocede el control de acceso al registro anterior.

UNIT: es la forma de especificar el número de unidad lógica asignada a un archivo y es representada con un número entero, de tal forma que, aunque puede ser designado, existen cuatro unidades lógicas ya predeterminadas, **Tabla 1. 10**.

Tabla 1. 22
Unidades lógicas predeterminadas

Unidad lógica	Descripción
0	Se predetermina como pantalla o como teclado.
5	Se predetermina como teclado.
6	Se predetermina como pantalla.
*	Siempre representa la pantalla y el teclado.

1.6.2 Apertura de archivos

Existe una manera fácil y sencilla para abrir archivos cuya sintaxis corresponde a:

Open(<n>,file='<archivo>')

o bien

open(<n>,file='<nombre_archivo>') ,

donde *<n>* es de tipo entero, cuyo valor debe distinto a 0,5 y 6. *<archivo>* corresponde al archivo con la ubicación que se desea abrir y *<nombre_archivo>* es de tipo carácter, cuyo valor da el archivo (con su ubicación) que se desea abrir.

Ejemplo:

Open(10,file='informe.out')

⋮

Open(101,file=archivo)

La otra forma para abrir un archivo, consiste en utilizar la instrucción *open* con sus respectivas opciones, la sintaxis corresponde a:

Open([unit=]<n>,file=<archivo>[,<opción>=<valor>])

Siendo las posibles opciones útiles: status, action, position, iostat, err. form, fccess y recl.

La **Tabla 1. 11** presenta una recopilación de las posibles opciones que pueden ser empleadas

Tabla 1. 23
Apertura de archivos

Instrucción	Tipo	Función	Sintaxis
Unit	entero	Identifica la unidad lógica que identificara el archivo a abrir. Se coloca implícitamente en el open.	[unit=]<unidad>
File	carácter	Opción obligatoria después de la opción unit.	File=<nombre_archivo>
Status	carácter	<valor> : ‘unknow’ - si existe lo mantiene y si no, lo crea. ‘old´-cuando el archivo ya existe y en caso de que no existiera, la computadora marca un error. ‘new’ – el archivo no existe y es creado, cuando existe este es borrado para crearlo. ‘scratch´- archivo temporal creado para el uso del programa, y una vez creado se destruye ‘replace’ – el archivo ya existe, pero se reemplazará por uno nuevo.	Status=<valor>
Action	carácter	Donde <valor> puede ser: ‘readwrite’ – opción por defecto, el archivo puede leerse y escribirse ‘read’ – el archivo es solo de lectura. ‘write’- el archivo es solo de escritura.	Action=<valor>
Position	carácter	Donde <valor> puede ser: ‘asis’ – una vez abierto, si la primera instrucción con el archivo es de lectura, la	Position=<valor>

Tabla 1. 23
Apertura de archivos

Instrucción	Tipo	Función	Sintaxis
		posición del archivo es la primera línea del archivo y si es de escritura la posición es una nueva línea posterior a la última. Opción por defecto. 'rewind' – la posición es la primera línea. 'append'- la posición es la línea posterior a la última línea.	
Iostat	entero	Permite que el programa continúe en caso de presentarse un error en la apertura de un archivo. Si <ierr>=0 no se ha detectado ningún problema Si <ierr> diferente a 0 hay un problema en la apertura	Iostat=<ierr>
Err	entero	Permite que el programa continúe en caso de un error durante la apertura del archivo. Donde <etiqueta> puede tomar valores de 1 a 99999. Puedes poner una nota especifica al momento de mostrar el error y lo referencia a la etiqueta.	Err=<etiqueta>
Form	Caractér	'formatted'- Es la opción por defecto cuando el archivo es de tipo texto. 'unformatted' – cuando la representación de los datos del archivo se realiza en lenguaje máquina o binario.	Form=<valor>
Access	Carácter	'sequential' – opción por defecto, la computadora lee y escribe los datos línea a línea.	Access=<valor>

Tabla 1. 23
Apertura de archivos

Instrucción	Tipo	Función	Sintaxis
		'direct' – la instrucción de lectura/escritura indica el número de registro a tratar	
Recl	entero	Obligatorio, cuando el acceso es directo Donde <n> es el número de bytes que ocupa en el registro	Recl=<n>

En el ejemplo de la **Fig. 1. 3** que implementa el manejo de archivos, se escribe en la unidad preconectada 6, luego se conecta con un archivo externo SENOS y se escribe en él, finalmente se conecta a la pantalla.

```
program archivos
real::twta,angulo
write(6,*)"esta es la unidad 6" !unidad pre conectada 6
open(unit=10, file="cosenos.txt", status="new") !uso de la instruccion open
!para conectar a la unidad 10 a un archivo externo llamado Cosenos
do teta=.1,6.3,.1
  angulo=cos(teta)
write(10,100)teta,angulo !Escribe en el archivo cosenos
write(6,100)teta, angulo ! Escribe en pantalla los resultados
  100 format(f3.1,f5.2)
  end do
  close(10) !cerrando
write(6,*)"tabla de cosenos "
end program archivos
```

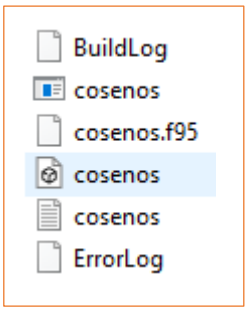


Fig. 1. 12
Código de programa manejo de archivos y carpetas creadas tras ejecutarlo.

1.7 Arreglos en FORTRAN

En FORTRAN puede ser utilizado un tipo de variable que sirve para almacenar vectores y matrices; utilizándose un solo nombre para referirse al conjunto de direcciones de memoria en la que es almacenado la matriz o vector que puede contener hasta n elementos; a este conjunto de datos se les conoce como tablas, arrays o tableros.

La manera de declarar una tabla es: nombre (nombre válido de variable), su tipo (entero, real, …) y sus dimensiones (número de subíndices y extensión de estos).

Estos arreglos pueden ser declarados de dos tipos: estáticos (dimensión fija) y dinámicos (dimensión abierta).

Los arreglos pueden ser tratados por Fortran, como listas de valores para ciertas funciones y como matrices, en el sentido matemático, por otras.

Un tablero unidimensional (vector) corresponde a una lista ordenada de valores a un índice

$$x_{ni}, x_{ni+1}, \dots, x_i, \dots, x_{ns-1}, x_{ns}$$

donde ni corresponde al límite inferior y suele tomar el valor de uno y ns corresponde al límite superior y puede tomar cualquier valor mientras que $ns \geq ni$.

Un tablero bidimensional (matriz) corresponde a una lista indexada de valores a dos índices:

$$\begin{matrix} x_{ni,mi} & x_{ni,mi+1} & \cdots & x_{ni,j} & \cdots & x_{ni,ms} \\ x_{ni+1,mi} & x_{ni+1,mi+1} & & & \cdots & x_{ni+1,ms} \\ \vdots & & & & & \\ x_{i,mi} & & & & & \\ \vdots & & & x_{i,j} & & \\ x_{ns,mi} & & & x_{ns,j} & & x_{ns,ms} \end{matrix} ,$$

donde i corresponde a las filas y j a las columnas

Arreglos dinámicos: permite indicar que sera utilizada una tabla sin especificar el número de subíndices y extención hasta el momento en que se necesite durante la ejecucion del programa.

1.7.1 Arreglos de tamaño fijo

Arreglos estáticos: se declaran las dimenciones de la tabla, la memoria necesaria para almacenar los elementos de la tabla se reserva cuando se carga el programa en memoria,el tamaño de la tabla, y la cantidad de memoria que usa se mantiene constante durante su ejecucion.

Hay diferentes formas para hacer una declaración estática de una tabla. Pero para una mejor facilidad se recomienda declararla bajo las siguientes sintaxis:

<tipo> nombre (<dimensiones>)

<tipo> <dimensión>(dimensiones)::nombre

Por ejemplo, para declarar una matriz real de nombre matriz con 20 filas y 30 columnas se haría de la siguiente manera:

Real::matriz (20,30) *Real, dimension (20,30)::matriz*

Para declarar un vector entero, cuyo nombre sea vector, con capacidad para cien elementos, se tiene:

Integer::vector (100) *Integer, dimension (100)::vector*

De la misma manera, se pueden hacer declaraciones del tipo carácter de la siguiente manera:

Carácter(len=15)::nombre *!variable nombre de longitud de 15 caracteres incluyendo espacios*

1.7.2 Asignación dinámica de memoria

La declaración de este tipo de tablas se realiza en dos etapas:

1. Declaración de la tabla mediante el atributo ALLOCATABLE, no se da valor a la extensión:
   <Tipo> allocatable :: <nombre_tabla> (ind-dim)
   <Tipo> allocatable,dimensión (ind-dim) :: <nombre_tabla>
   donde ind-dim indica el número de dimensiones de la tabla
2. Cuando se quiera crear una tabla, se usará la instrucción:
   allocate(<nombre_tabla>(dim))
   y para destruir una tabla que ya no será utilizada la instrucción es:
   deallocate(<nombre de la tabla>)

A continuación, se presenta un pequeño ejemplo ilustrando una declaración dinámica de una matriz:

real, allocatable::tabla(:,:)

print,'dimension de la matriz:'*

read', n

allocate (tabla(n,n))

real :: a(10,10)

a=0

Cuando se trabaja con matrices fortran suele convertirlos a una forma unidimensional por columnas[1], **Fig. 1. 4**, es decir coloca los valores de la primera columna, enseguida la segunda y así sucesivamente.

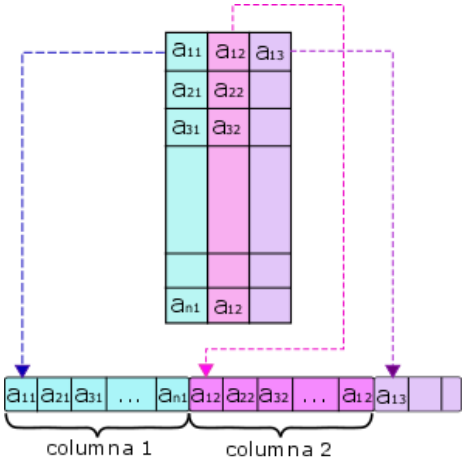


Fig. 1. 13
Conversión de tableros

En general, la asignación de un escalar a una tabla es entendida por el compilador como una asignación elemento a elemento. También se puede utilizar elementos de una

[1] Considerar la forma de conversión de tablas en otros lenguajes, por ejemplo, C las convierte por filas NO en columnas

tabla de manera individual. Para ello se referencian mediante su nombre y los valores de sus subíndices encerrados entre paréntesis: $v(i)$ para v_i y $a(i,j)$ para $a_{i,j}$.

Es posible asignar valores a vectores escribiendo a la derecha del signo igual los valores encerrados entre los signos (/y/), y separándolos por comas, **Tabla 1.12.**

Tabla 1. 24
Ejemplos de tablas

Caso 1	**Explicación**
Real :: v(5), w(5) *V=(/1.,2.,3.,4.,5./)* *X=1.1* *W=(/0.,x,x-1,0.,-1./)* *W=(/0.,1.1,0.1,0.,-1./)*	*! Se declaran dos vectores v y w de cinco elementos reales.* *! A los elementos del vector v se les asigna los valores de 1, 2, 3,4 y 5.* *! A la variable x se le asigna un valor fijo (1.1 en este caso).* *! Se asignan los valores al vector w, donde dos de sus elementos dependen de x.* *!se muestra el vector w con los valores que toma para el valor de x.*
Caso 2	**Explicación**
Integer::b(5) *b=(/(i,i=1,5)/)*	*! Vector b de cinco elementos enteros* *! Asigna al vector b los valores de 1 a 5 para cada elemento*
Caso 3 – utilizando un ciclo Do	**Explicación**
Integer ::b(10) *b=(/i,i=2,20,2)/)*	*! Vector b de 10 elementos enteros* *! El vector b se le asignaran los valores de* *2,4,6,8,10,12,14,16,18,20 debido a las especificaciones del ciclo*

Tabla 1. 24
Ejemplos de tablas

Caso 1	Explicación
program vector	
implicit none	
integer:: i, n	
integer :: a(20)	!se declara vector de 20 posiciones
read(*,1) n	!lee un número entero de solo un número
1 format (i1) do i=1, n	!lee um num entero de solo un número
read(*,2) a(i)	! Va guardando elemento por elemento con formato de etiqueta 2
write (*,2) a(i) 2 format (i5)	! Imprimimos los elementos ingresados, con formato de la etiqueta
end do	! Formato que permite números de hasta 5 números etiqueta 2
end program vector	

1.7.3 Función reshape

Esta función sirve para reestructurar una tabla interpretándola con otras dimensiones. Puesto que los elementos de una tabla se almacenan en direcciones consecutivas de memoria, ordenados de la forma natural si son vectores y por columnas en caso de matrices, es posible reinterpretarlos cambiando sus dimensiones.

Sea la matriz *M* de dimensiones 3x4:

$$M = \begin{bmatrix} 1 & 5 & 9 \\ 2 & 6 & 10 \\ 3 & 7 & 11 \\ 4 & 8 & 12 \end{bmatrix}$$

M 1

Sus elementos son almacenados en la memoria en el siguiente orden:

$M_{1,1}$	$M_{2,1}$	$M_{3,1}$	$M_{4,1}$	$M_{1,2}$	$M_{2,2}$	$M_{3,2}$	$M_{4,2}$	$M_{1,3}$	$M_{2,3}$	$M_{3,3}$	$M_{4,3}$
1	2	3	4	5	6	7	8	9	10	11	12

Si se reinterpreta este conjunto de direcciones de memoria como una matriz 2x6 se obtiene:

$$M = \begin{bmatrix} 1 & 3 & 5 & 7 & 9 & 11 \\ 2 & 4 & 6 & 8 & 10 & 12 \end{bmatrix}$$

M 2

Y si la interpretamos como una matriz de 3x4 se obtendrá:

$$M = \begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix}$$

M 3

Para esto es utilizada la función reshape, siendo su sintaxis:

Reshape (input,forma)

donde

Input: corresponde a la tabla que se quiere re-interpretar.

Forma: corresponde a un vector entero indicando las nuevinput.as dimensiones que se quieran dar a la tabla, por ejemplo, con reshape (M,(/2,6/)) se obtendrá la matriz **M 2**. Esta función se puede utilizar para asignar valores a tablas con más de una dimensión. Así, por ejemplo, para conseguir la matriz **M 3** , se emplearían las siguientes instrucciones:

```
Integer:: m(4,3)                    ! Matriz m de 4x3, cuyos elementos que la
M=reshape((/i,i=1,12)/), (/4,3))    formarán estarán dados desde i=1, 2, 3, …,12
```

De este modo si requiriera realizar la asignación de valores a una matriz de 20x20 (nxn), esto se podría hacer de la siguiente manera:

```
Real*4:: a(20,20)                   ! declaración de la matriz a de dimensión
n=20                                20x20
n1=19                               ! a n el valor de 20 y a n1 el valor de 19
A=reshape                           ((/4.,  ! se da la instrucción de llenar la
(((0.,i=1,n1),1.,4./),j=1,n1)/),(/n,n/))    diagonal n y n1 con el valor de 20 y 19
```

1.8 Subprogramas

Cuando se realiza un programa complejo se recomienda dividir el programa en partes cada una de las cuales una tarea definida. Estas partes, es lo que llamaremos subprogramas; hay veces que el programa se presenta un conjunto de instrucciones que son repetidas, si éstas son programadas en un subprograma, el programa principal solo tiene que llamar a dicho subprograma sin la necesidad de repetir instrucciones una y otra vez.

Es por eso que el programar con subprogramas, facilita el trabajo del programador ya que el programa estará más estructurado.

Los programas escritos en FORTRAN se encuentran estructurados en unidades programáticas, en base a niveles jerárquicos, **Tabla 1. 13**. El programa principal, las subrutinas y funciones, son llamados procedimientos.

Tabla 1. 25
Unidades Programáticas

Unidad programática (UP)	Función
Program	UP principal, es en si el programa inicial que se está estructurando, explicado a lo largo de todo el capítulo.
Subroutine	Contempla las instrucciones ejecutables
Function	Contempla las instrucciones ejecutables
Module	Contempla instrucciones de declaración de variables, inicialización de variables, interfaces entre funciones y subrutinas

1.8.1 Subrutinas

Una subrutina es un subprograma de orden jerárquico de nivel inferior al subprograma principal, tiene como objetivo realizar las instrucciones ejecutables, éstas serán ejecutadas por un subprograma de nivel jerárquico superior a través de una instrucción que la llame. La estructura de la sintaxis a emplear es:

Subrutine <nombre>[(<argumentos(ficticios)>)]

! instrucciones de declaración de los argumentos (ficticios)

! instrucciones de declaración de los objetos locales

! instrucciones ejecutables

end subroutine <nombre>

Donde <argumentos(ficticios)>, corresponde a los objetos sobre los cuales la subrutina trabajará preferentemente, se encuentran separados por comas y pueden ser variables, funciones, subrutinas, tableros. Los objetos locales son de uso exclusivo de la rutina.

La subrutina es llamada por un subprograma de nivel jerárquico superior, mediante la instrucción call:

<identificación UP>

⋮

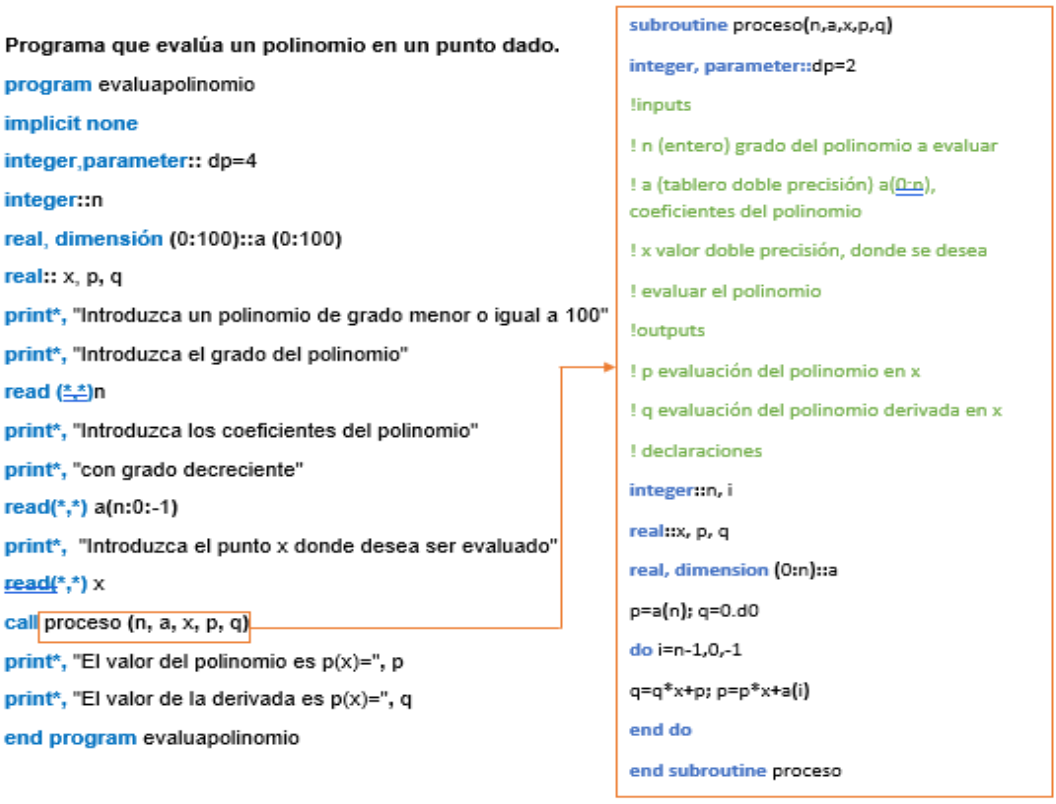
Call <nombre> [<argumentos (usados[2])>]

⋮

End <UP>

La **Fig. 1. 5,** presenta el programa principal que hace el llamado a otro programa mediante el uso de una subrutina.

[2] Donde lista de <argumentos (de uso)> debe coincidir con la de <argumentos (ficticios)>, en posición, tipo y clase.



```
Programa que evalúa un polinomio en un punto dado.
program evaluapolinomio
implicit none
integer,parameter:: dp=4
integer::n
real, dimensión (0:100)::a (0:100)
real:: x, p, q
print*, "Introduzca un polinomio de grado menor o igual a 100"
print*, "Introduzca el grado del polinomio"
read (*,*)n
print*, "Introduzca los coeficientes del polinomio"
print*, "con grado decreciente"
read(*,*) a(n:0:-1)
print*,  "Introduzca el punto x donde desea ser evaluado"
read(*,*) x
call proceso (n, a, x, p, q)
print*, "El valor del polinomio es p(x)=", p
print*, "El valor de la derivada es p(x)=", q
end program evaluapolinomio
```

```
subroutine proceso(n,a,x,p,q)
integer, parameter::dp=2
!inputs
! n (entero) grado del polinomio a evaluar
! a (tablero doble precisión) a(0:n),
coeficientes del polinomio
! x valor doble precisión, donde se desea
! evaluar el polinomio
!outputs
! p evaluación del polinomio en x
! q evaluación del polinomio derivada en x
! declaraciones
integer::n, i
real::x, p, q
real, dimension (0:n)::a
p=a(n); q=0.d0
do i=n-1,0,-1
q=q*x+p; p=p*x+a(i)
end do
end subroutine proceso
```

Fig. 1.14
Ejemplo del uso de una subrutina.

1.8.2 Funciones

Una función es un tipo particular de subrutina, a la cual se accede directamente sin utilizar la instrucción *call*, por ello, es aplicable para ella lo abordado en la subrutina.

La función como UP es llamada al interior de una expresión, cuyo resultado se utiliza en la misma. La sintaxis correspondiente para una función es la siguiente:

[Tipo]function <nombe>[(<argumentos ficticios>)][result (<esultado>)]

! declaración función (nombre o resultados)

! declaración de argumentos ficticios

! declaración de objetos locales

! declaración de instrucciones ejecutables

End function <nombre>

1.9 Graficador Gnuplot

Gnuplot es un programa que visualiza datos científicos, permitiendo graficar en 2D y 3D, tiene la característica que resulta ser fácil de usar y maneja un acabado de alta calidad ofreciendo a los usuarios:

1. Representación bidimensional y tridimensional.
2. Facilidad para manejar gráficas, ejes y puntos.
3. Manejo números enteros, decimales y complejos.
4. Tiene funciones definidas, pero también pueden ser definidas por el usuario
5. Se puede recibir ayuda en línea.

1.9.1 Descarga e instalación de Gnuplot

A continuación, se presenta un pequeño instructivo para la instalación del software en línea.

1. Ingresar al navegador de su preferencia y escribir la palabra “gnuplot”,
2. Una vez realizada la búsqueda en línea, ingresar al sitio oficial de gnuplot[3].
3. Ya ingresado a la página identificar en la parte inferior “versión 5.0 (previus stable)” e identificar “Download from SourceForge”, **Fig. 1. 6**, acceder a la liga desde sourceforge para descargar la versión más reciente de gnuplot.
4. Terminada la descarga, se deberá ingresar a la carpeta de descargas y desde ahí se deberá iniciar la instalación
5. El sistema solicitará la autorización para que el programa realice cambios en el equipo las cuales deberán ser aceptadas.

[3] http://www.gnuplot.info/

Fig. 1. 15
Ventana principal Gnuplot

6. Elegir el idioma a usar durante el proceso (no incluye el español, por lo que se recomienda el inglés).
7. Aceptar los términos de la licencia y dar siguiente
8. Cerrar las aplicaciones abiertas y dar clic en el botón next, se abrirá una nueva ventana, dar siguiente.
9. Se desplegará la ruta de alojamiento del programa una vez instalado (se recomienda dejar la mostrada por defecto) y dar next.
10. Aparecerá en una nueva ventana los elementos a ser instalados, se aceptan los que ya están seleccionados y dar siguiente.
11. En la nueva ventana se recomienda que el usuario creé un acceso directo al programa con una carpeta con el nombre del programa, activar en la parte inferior izquierda y dar next.
12. Se abrirá una ventana donde se muestran opciones adicionales, se recomienda dejar los valores seleccionados o dar next.
13. Por último, se mostrará la información referente a la instalación del programa, dar clic en “install” para dar por iniciado el proceso de instalación.

14. Una vez concluida la instalación, buscar la carpeta “gnuplot” e identificar el ícono con el nombre “gnuplot 5.0 patchlevel 3”, **Fig. 1. 7**, y ejecutarlo.

Fig. 1. 16
Ícono de acceso directo a Gnuplot

15. Para comprobar la instalación, ejecutar el programa, si todo fue correcto este deberá iniciar con una ventana como la de la **Fig. 1. 8** .

Fig. 1. 17
Ventana de inicio de Gnuplot

A partir de este punto puede iniciarse el proceso de graficar datos generados por FORTRAN.

1.9.2 Comandos básicos de GNUPLOT

Cuando se grafica en Gnuplot es importante señalar que este maneja al eje “x” como la variable independiente en gráficos 2D y a “x” y “y” para los de 3D.

Así bien, los comandos fundamentales para graficar funciones o datos de diversas maneras son:

1. Plot-usado para gráficas bidimensionales

2. Splot- usado para gráficas tridimensionales

La sintaxis por emplear es la siguiente:

Plot {<rango>}

{<iteración>}

{<función>}

{<función>|{<archivo>}

Donde función puede corresponder a una función en específico o bien al nombre entre comillas del archivo que se desea graficar. Algunas de las funciones a emplear son las mostradas en la **Tabla 1. 6**.

A continuación se presenta en la **Fig. 1. 9** un ejemplo de la función seno (2x) directamente desde gnuplop y se adjunta su gráfica.

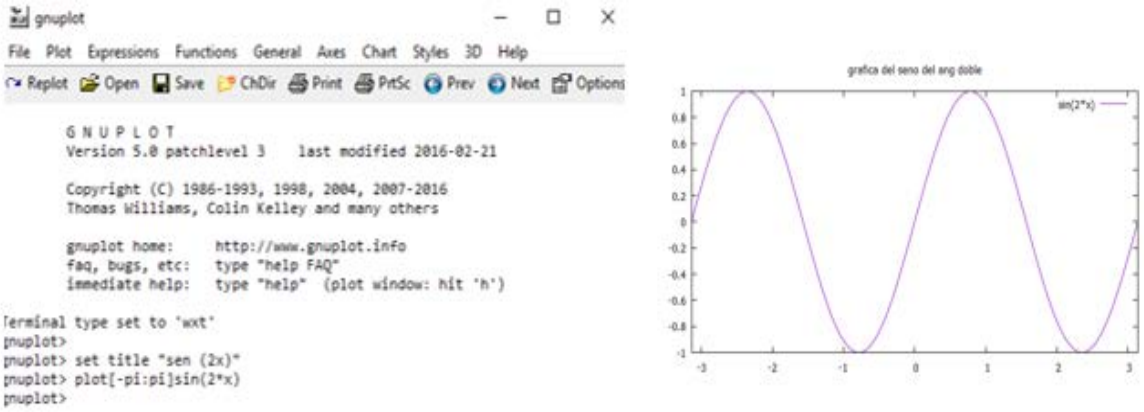


Fig. 1. 18
Función Sin (2x)

Una vez que ya se tiene graficada la función, se pueden realizar distintos cambios en cuanto a la forma en que puede ser presentada, pudiéndose modificar: el título, nombre de los ejes, escala, color, tipo de línea, etc.; para esto existen dos formas de especificar las opciones.

1. Utilizar comandos Set y Show para establecer y mostrar su valor.

2. Parámetros específicos de una orden de representación gráfica (Plot y Splot), donde solo se ve afectado el grafico.

Una vez que son implementados estos cambios, para que estos sean implementados, será necesario ejecutar el comando “replot” o bien, graficar nuevamente. En caso de que se quiera graficar la misma función dos veces o más juntas, esto puede hacerse simplemente separándolas con una coma en la misma línea:

“plot<funcion1>, <funcion2>”

Y si llegase a darse el caso que al graficar dos o más funciones uno o ambos rangos necesitan ser modificados, esto puede hacerse mediante el comando “set_range[xx,xx]”, donde se deberá especificar en el espacio el aje a modificar, si solo se requiriere cambiar uno y el otro no, el segundo deberá indicarse como “[ ]”.

En Gnuplot se pueden realizar gráficos múltiples en una misma ventana, esto se hace si se define el modo “multiplot” y definiendo el área a utilizar.

Capítulo 2

Solución a sistemas de ecuaciones lineales

Corresponde a un conjunto de ecuaciones de la forma:

$$\begin{matrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{matrix}, \quad \text{.....} \quad \textbf{(2.1)}$$

donde se pueden tener *n* ecuaciones y *n* incógnitas.

Dar solución a un sistema lineal de ecuaciones consiste en calcular las incógnitas presentes para que se cumplan todas las condiciones de manera simultánea; esto puede hacerse si se expresa el sistema en forma matricial, para tener un manejo óptimo de las variables.

Por lo anterior, en este capítulo se abordarán métodos directos e iterativos de solución, buscando así la mejor aproximación en cada caso.

2.1 Métodos directos de solución

Son aquellos que permiten resolver de forma eficiente el sistema de ecuaciones. Entre estos tipos se encuentran los métodos de Eliminación Gaussiana y de Descomposición LU, que se describen a continuación.

2.1.1 Eliminación Gaussiana (EG)

Este método permite reducir el número de incógnitas del sistema de ecuaciones inicial a través de una serie de combinaciones lineales. Es empleado preferiblemente cuando se trabaja con un sistema de más de tres ecuaciones.

Dado un sistema como el de la **Ec. (2.1)**, el proceso para su solución a través del método de **EG** consiste en los siguientes dos pasos:

(1) Eliminación de incógnitas hacia adelante, este proceso busca reducir el conjunto de ecuaciones a un sistema triangular superior, eliminando la primera incógnita, x_1 , desde la segunda hasta la n-ésima ecuación.

(2) Sustitución hacia atrás, teniendo el sistema triangular superior se despeja x_n para conocer su respectivo valor, después este valor es sustituido hacia atrás desde la (n-1) -enésima ecuación para despejar y obtener el valor de cada una de las incógnitas que conforman el sistema de ecuaciones.

El algoritmo de implementación se presenta enseguida, y el diagrama correspondiente se incluye en la **Fig. 2. 1**:

1. Tener un sistema de ecuaciones en la forma presentada en la **Ec. (2.1)**.
2. *Eliminación de incógnitas hacia adelante.*
   2.1 Eliminar la primera incógnita, x_1 , desde la segunda ecuación, para ello se multiplica a la ecuación por $\frac{a_{21}}{a_{11}}$ para obtener **Ec. (2.2):**

$$a_{21}x_1 + \frac{a_{21}}{a_{11}}a_{12}x_2 + \cdots + \frac{a_{21}}{a_{11}}a_{1n}x_n = \frac{a_{21}}{a_{11}}b_1 \ , \quad \text{.....} \quad \textbf{(2.2)}$$

2.2 Esta ecuación se resta a la segunda fila, obteniendo:

$$a'_{22}x_2 + \cdots + a'_{2n}x_n = b'_2 \ , \quad \text{.....} \quad \textbf{(2.3)}$$

donde el subíndice prima indica que los elementos ahora son distintos a sus valores originales.

2.3 Este procedimiento se repite con cada una de las ecuaciones restantes, hasta eliminar a x_1 de todas ellas.

2.4 Repetir el procedimiento antes descrito para eliminar cada una de las incógnitas $x_2, x_3, \dots, x_{n-1}$, del nuevo sistema de ecuaciones que se vaya obteniendo, hasta llegar al sistema triangular superior:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\ a'_{22}x_2 + a'_{23}x_3 + \cdots + a'_{2n}x_n &= b'_2 \\ a''_{33}x_3 + \cdots + a''_{3n}x_n &= b''_{33} \\ \vdots & \\ a_{nn}^{n-1}x_n &= b_n^{n-1} \end{aligned} \quad . \quad \text{.....} \quad \textbf{(2.4)}$$

3. Sustitución *hacia atrás*

3.1 De la Ec. **(2.4)**, despejar a x_n:

$$x_n = \frac{b_n^{(n-1)}}{a_{nn}^{(n-1)}} \ , \quad \text{.....} \quad \textbf{(2.5)}$$

3.2 Sustituir hacia atrás el valor de x_n, en la (n-1)- ésima ecuación y despejar a x_{n-1}

3.3 Repetir el procedimiento para evaluar las x restantes, mediante:

$$x_i = \frac{b_i^{(i-1)} - \sum_{j=i+1}^{n} a_{ij}^{(i-1)}\, x_j}{a_{ii}^{(i-1)}} \ , \quad \text{.....} \quad \textbf{(2.6)}$$

para $i = n-1, n-2, \dots, 1$.

4. Una vez obtenidas todas las incógnitas, se puede verificar la solución al sustituirlos en cada una de las ecuaciones que forman el sistema **(2.1)**, éstas deberán cumplirse para cada una de ellas.
5. El vector solución x es: x_i para i={1, 2,…,n}

Parte esencial para la elaboración del código

- Eliminación hacia adelante

```
do k=1, n-1
do i =k+1, n
factor= ai,k/ak,k
do j=k+1, n
ai,j= ai,j-(factor*ak,j)
end do
bi=bi-(facor*bk)
end do
end do
```

- Eliminación hacia atrás

```
xn= bn/an,n
do i=n-1,1,-1
sum=bi
do j=i+1, n
sum=sum-(ai,j*xj)
end do
xi=sum/ai,i
end do
```

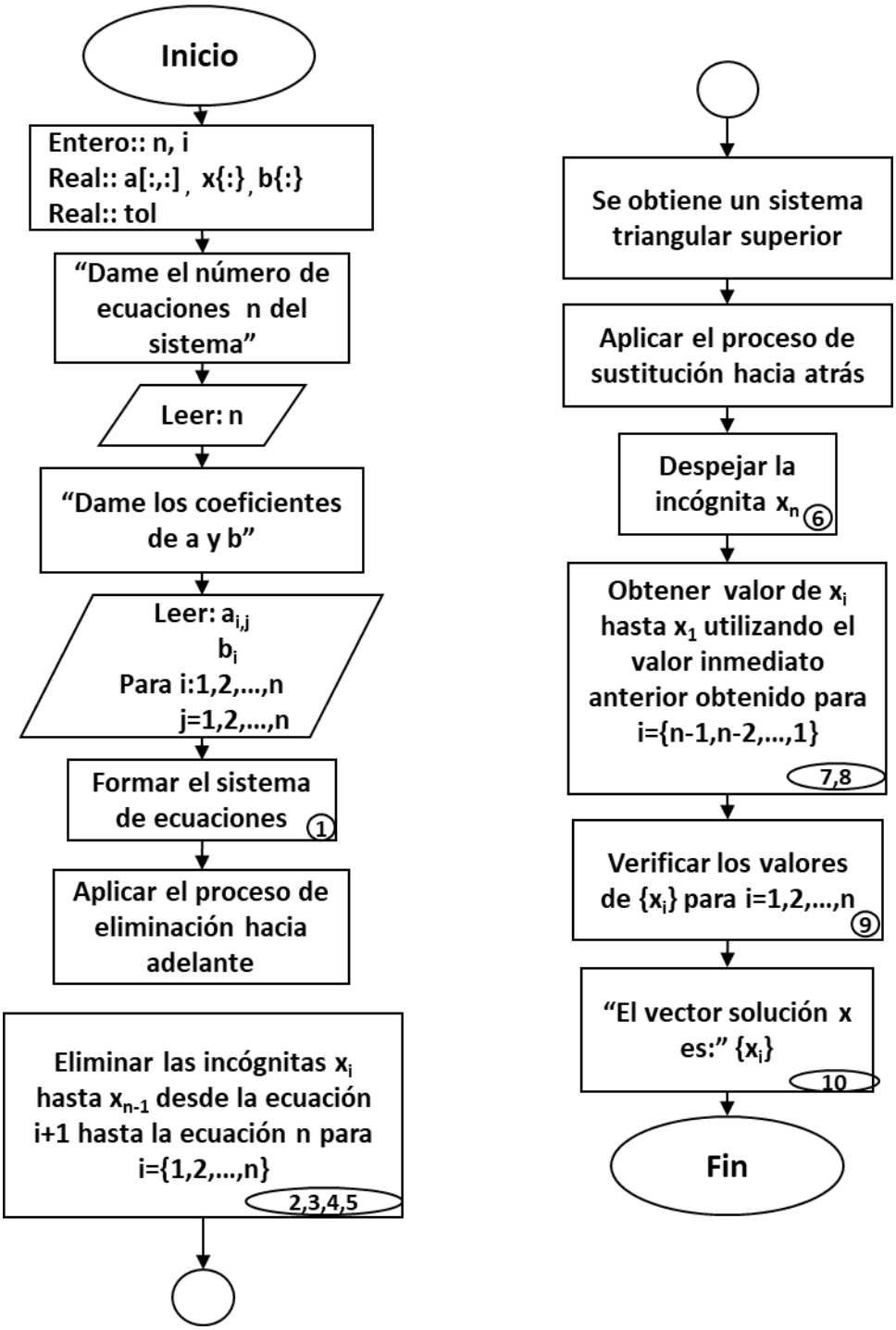


Fig. 2. 6
Eliminación Gaussiana (EG).

2.1.2 Descomposición LU

Al igual que la EG, la descomposición LU busca dar solución a un sistema de ecuaciones de forma simultánea[4],

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{21} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{Bmatrix} , \quad \text{..... (2.6)}$$

con la diferencia de que la descomposición LU consiste en encontrar dos matrices, L y U construidas de tal forma que cumpla que:

$$A = L \cdot U \ , \quad \text{..... (2.7)}$$

por lo que partiendo de la **Ec. (2.8):**

$$[A]\{X\} - \{B\} = 0 \ , \quad \text{..... (2.8)}$$

esta ecuación también puede ser expresa de forma matricial y tras un reordenamiento se obtiene:

$$[U]\{X\} - \{D\} = 0 \ , \quad \text{..... (2.9)}$$

ahora suponiendo que existe una matriz diagonal inferior con números 1 en su diagonal principal de forma:

$$L = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & 0 \\ l_{31} & l_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & 1 \end{bmatrix} , \quad \text{..... (2.10)}$$

de manera que

$$[L]\big\{[U]\{X\} - \{D\}\big\} = [A]\{X\} - \{B\} \ ,$$

que, si se satisface, se obtiene la **Ec. (2.7)** en su forma matricial:

[4] Los elementos de la matriz $[A]$ se obtienen del sistema de ecuaciones; $\{X\}$ es el vector solución del sistema y $\{B\}$ es el vector al cual esta igualado el sistema inicialmente

$$[L]\,[U] = [A]$$

y también se llega a:

$$[L]\{D\} = \{B\}\ . \qquad \text{.....} \qquad \textbf{(2.11)}$$

Los elementos de las matrices $[L]\ y\ [U]$ se obtienen a partir de aplicar el proceso de EG a la matriz $[A]$.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{21} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix}, \qquad \text{.....} \qquad \textbf{(2.12)}$$

$$U = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & a'_{22} & a'_{23} & \cdots & a'_{2n} \\ 0 & 0 & a''_{33} & \cdots & a''_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a''_{nn} \end{bmatrix}, \qquad \text{.....} \qquad \textbf{(2.13)}$$

y

$$L = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & 0 \\ l_{31} & l_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & 1 \end{bmatrix}. \qquad \text{.....} \qquad \textbf{(2.10)}$$

El algoritmo de implementación se presenta enseguida, así como el diagrama correspondiente se incluye en la **Fig. 2. 2**:

1. Representar el sistema como la **Ec. (2.6)**.
2. Obtener la matriz $[U]$ haciendo uso de EG al diagonalizar la matriz $[A]$ **Ec. (2.13)** (usando solo la operación adición).
3. Obtener la matriz $[L]$ que surgirá del proceso de EG para obtener la matriz anterior.
4. Utilizando la fase de sustitución hacia adelante con la **Ec. (2.11)** de donde se obtendrán los valores respectivos de $\{D\}$[5]:

[5] Se conoce $[L]$ y $\{B\}$ la única incógnita será $\{D\}$.

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ \frac{a_{21}}{a_{11}} & 1 & 0 & \cdots & 0 \\ \frac{a_{31}}{a_{11}} & \frac{a'_{32}}{a'_{22}} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{a_{n1}}{a_{11}} & \frac{a''_{n2}}{a''_{22}} & \frac{a'''_{n3}}{a''_{33}} & \cdots & 1 \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{Bmatrix} . \quad \text{.....f} \quad \textbf{(2.11)}$$

5. Sustituir a U y D en la **Ec. (2.9)**, para obtener los valores de $\{X\}$ a partir del proceso de sustitución hacia atrás.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & a_{21} & a_{21} & \cdots & a_{2n} \\ 0 & 0 & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{Bmatrix} = \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{Bmatrix} . \quad \text{.....} \quad \textbf{(2.9)}$$

6. Los elementos del vector $\{X\}$ dan solución al sistema de ecuaciones representado por la **Ec**. **(2.6)**.

Parte esencial para la elaboración del código

• Descomposición	• Sustitución hacia adelante	• Sustitución hacia atrás
do k=1, n-1	sub sustitute (a,n,b,x)	xn=bn/an,n
do i=k+1,n	do i=2,n	do i=n-1,1,-1
factor=ai,k/ak,k	sum=bi	sum=0
ai,k=factor	do j=1,i-1	do j=i+1,n
do j=k+1,n	sum=sum-ai,j*bj	sum=sum+ai,j*xj
ai,j=ai,j-factor*ak,j	end do	end do
end do	bi=sum	xi=(bi-sum)/ai,i
end do	end do	end do
end do		end substitute
end descompose		

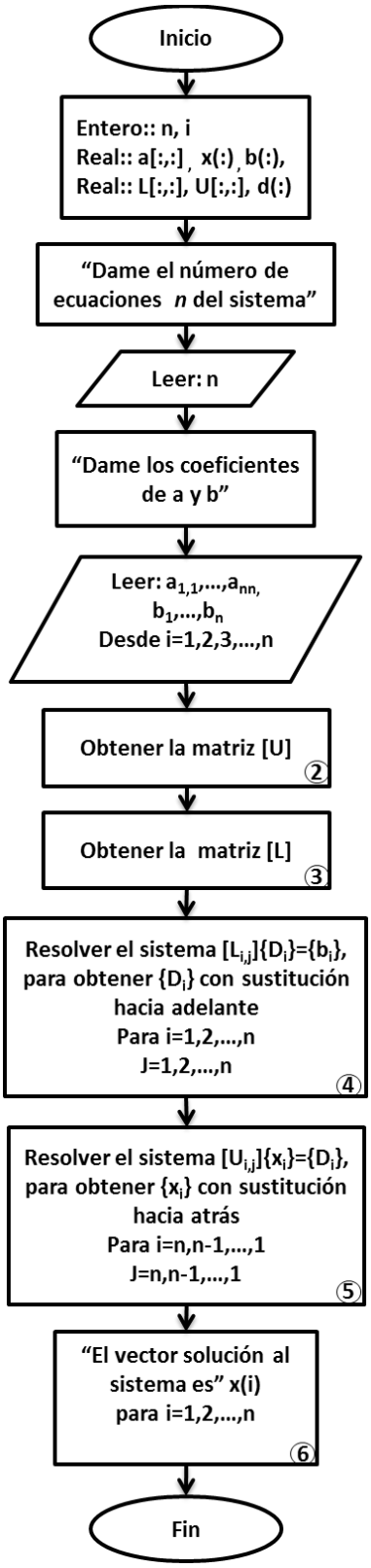


Fig. 2. 7
Descomposición LU

2.1.3 Método de Thomas

El algoritmo de Thomas o también conocido como algoritmo para matrices tridiagonales, sirve para resolver matrices tridiagonales de manera eficiente.

Donde las ecuaciones de las matrices tridiagonales tienen la forma de la **Ec. (2.10)**:

$$a_i\, x_{i-1} + b_i x_i + c_i\, x_{i+1} = r_i\,, \quad \text{.....} \quad \textbf{(2.10)}$$

donde $a_1 = 0$ y $c_n = 0$, pudiéndose representar matricialmente como:

$$\begin{bmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ & & & a_n & b_n \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{Bmatrix} = \begin{Bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_n \end{Bmatrix}. \quad \text{.....} \quad \textbf{(2.11)}$$

Este método parte de una matriz tridiagonal la cual busca transformarse a una triangular mediante el método de EG, por lo que se reemplaza cada fila por una combinación lineal de filas apropiada, de manera que sean anulados los elementos de la diagonal inferior y los de la diagonal superior tengan un valor de uno.

Para realizar este proceso es importante identificar los vectores por los cuales se encuentra transformada la matriz. El vector $\{a\}$ se encuentra formado por los elementos de la diagonal inferior de la matriz tridiagonal, con el primer elemento igual a cero, el vector $\{b\}$ estará formado por los elementos de la diagonal principal, el vector $\{c\}$ conformado por los elementos de la diagonal superior con el último elemento igual a cero y el vector $\{r\}$ se formará con los valores correspondientes de los resultados, todos los vectores serán de orden n.

Este método recurre a la descomposición LU y a su vez a la EG, por lo que se busca descomponer la matriz tridiagonal en L y U mediante las siguientes Ecuaciones

$$u_{1,1} = b_1\,, \quad \text{.....} \quad \textbf{(2.12)}$$

$$u_{i,i} = b_i - \left(l_{i,i-1}\right)\left(u_{i-1,i}\right)\,, \quad \text{.....} \quad \textbf{(2.11)}$$

$$u_{i-1,i} = c_{i-1} \ , \qquad \text{.....} \qquad \textbf{(2.12)}$$

$$l_{i,i-1} = \frac{a}{u_{i-1,i-1}} \ , \qquad \text{.....} \qquad \textbf{(2.13)}$$

las cuales se usarán n número de veces para obtener así las matrices de orden n, una vez que ya se tienen cada uno de los elementos, se llevan a la forma:

$$[L]\{D\} = \{r\} \ , \qquad \text{.....} \qquad \textbf{(2.16)}$$

y mediante sustitución progresiva se obtiene los elementos de r; finalmente se realiza:

$$[U]\{X\} = \{D\} \ , \qquad \text{.....} \qquad \textbf{(2.17)}$$

para obtener los valores de los elementos del vector $\{X\}$.

El algoritmo de implementación se presenta a continuación, así como el diagrama correspondiente se incluye en la **Fig. 2. 3**:

1. Representar el sistema como el expresado en la **Ec. (2.6)**.
2. Identificar los vectores $\{a\}$, $\{b\}$, $\{c\}$ y $\{r\}$, de la siguiente forma:

$$a = \begin{Bmatrix} 0 \\ a_2 \\ a_3 \\ \vdots \\ a_{n-1} \\ a_n \end{Bmatrix} \qquad b = \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{n-1} \\ b_n \end{Bmatrix} \qquad c = \begin{Bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{n-1} \\ o \end{Bmatrix} \qquad r = \begin{Bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_n \end{Bmatrix}$$

3. Verificar que el sistema de ecuaciones cumpla con la condición:

$$|b_i| > |a_i| + |c_i|$$

4. De no cumplirse la condición anterior el sistema tridiagonal no puede resolverse mediante este método.
5. Descomponer la matriz tridiagonal en LU, aplicando **Ec. (2.12)**, **Ec. (2.13)**. **Ec. (2.13)** y **Ec**. **(2.14)** desde $i = 1$ hasta n para obtener los elementos correspondientes a las matrices.
6. Mediante sustitución progresiva obtener uno a uno los elementos de $[D]$, mediante la **Ec. (2.16).** Donde $[L]$ y $\{r\}$ ya son conocidos.

7. Mediante el proceso de sustitución regresiva encontrar los valores correspondientes al vector incógnita $\{X\}$ empleando la **Ec. (2.17)**.
8. El vector solución es: $x_i = \{x_i\}$.

Parte esencial para la elaboración del código

```
program metodothomas (n,a,b,c,d)
do k=2,n,1
    m= a(k)/b(k-1)
    b(k)=b(k)-m*c(k-1)
    d(k)=d(k)-m*d(k-1)
    end do
  r(n)=d(n)/b(n)
do k=n-1,1,-1
  r (k)=(d(K)-c(k)*r(k+1))/b(k)
  end do
end program Thomas
```

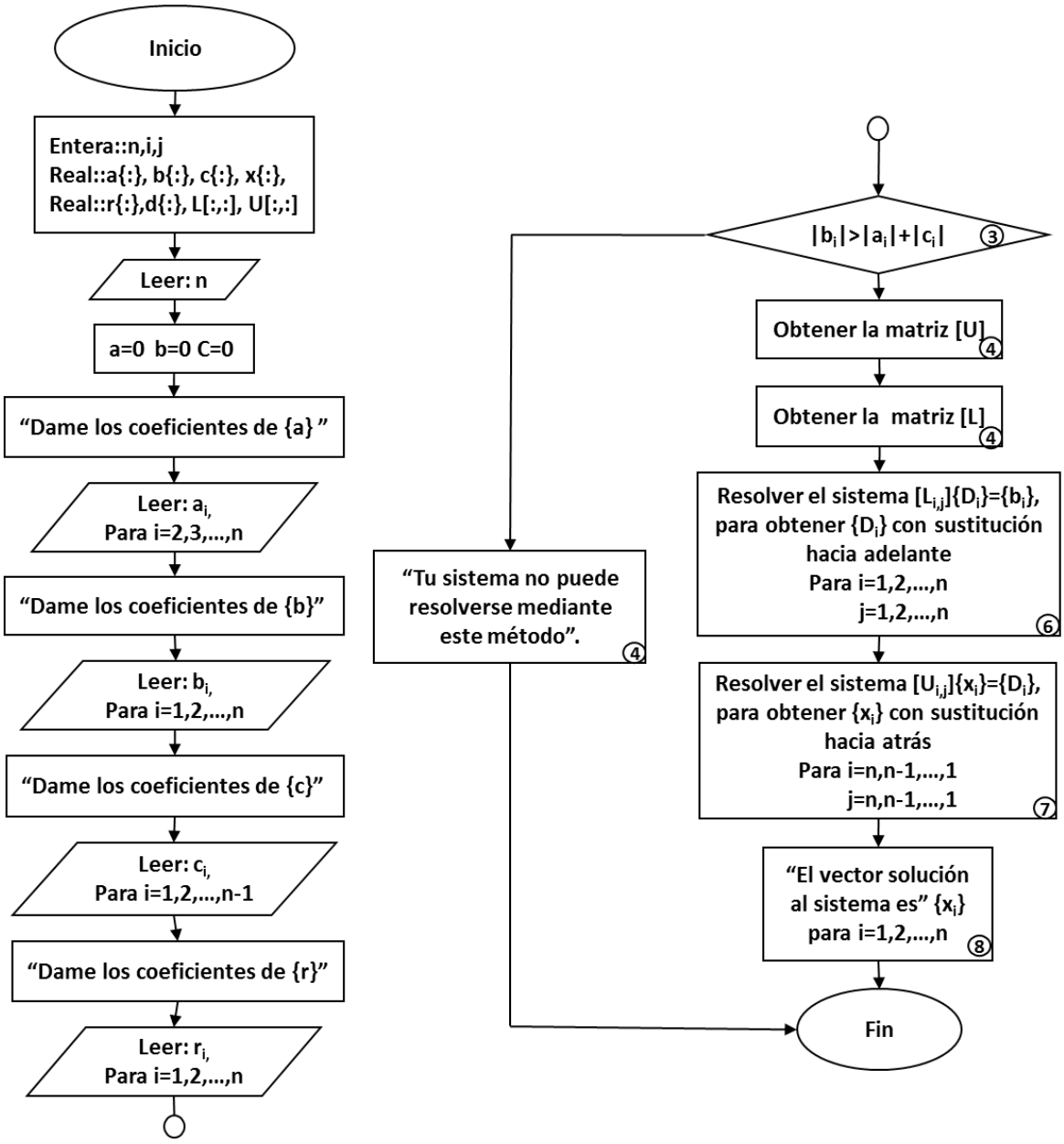


Fig. 2. 8
Método de Thomas

2.2 Métodos iterativos de solución

La aplicación de los métodos iterativos para dar solución a sistemas de ecuaciones lineales consiste primordialmente en transformarlo a otro, que sea a su vez equivalente para aproximar la solución paso a paso mediante un sistema repetitivo.

Es importante remarcar que la resolución iterativa no siempre puede ser aplicada, pero si resulta ser muy útil cuando se presentan sistemas que manejan un número grande de incógnitas.

Estos métodos requieren de una estimación inicial para poder dar comienzo a la iteración. La precisión de la solución obtenida por estos métodos iterativos depende principalmente del número de iteraciones que se realicen, así como de su convergencia[6].

2.2.1 Método de Jacobi

Es utilizado para resolver sistemas de ecuaciones lineales, el cual consiste primordialmente en obtener una ecuación o matriz de recurrencia y proponer un vector solución inicial, para posteriormente ir realizando las iteraciones necesarias hasta cumplir con la tolerancia establecida entre un valor y otro.

Partiendo del sistema de ecuaciones lineales $[A]\{X\} = \{B\}$ donde $[A]$ es la matriz de coeficientes, $\{X\}$ es el vector de incógnitas y corresponde al vector de términos independientes:

$$[A]\{X\} = \{B\} \ , \qquad \text{.....} \qquad \textbf{(2.18)}$$

$[A]$ puede ser sustituida por la suma entre dos matrices:

$$[A] = [D] + [R] \ , \qquad \text{.....} \qquad \textbf{(2.19)}$$

donde $[D]$ es una matriz cuya diagonal principal es igual a la de $[A]$ y sus demás elementos son ceros, y $[R]$ es una matriz con la diagonal principal igual a cero y sus demás elementos corresponden a los elementos de $[A]$.

[6] Convergencia: se refiere a cuando la sucesión de las aproximaciones es cada vez más próxima a la solución.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{21} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix}, \quad \text{..... (2.20)}$$

$$D = \begin{bmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ 0 & a_{22} & 0 & \cdots & 0 \\ 0 & 0 & a_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn} \end{bmatrix} \quad \text{..... (2.21)}$$

y

$$R = \begin{bmatrix} 0 & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & 0 & a_{21} & \cdots & a_{2n} \\ a_{31} & a_{32} & 0 & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & 0 \end{bmatrix}. \quad \text{..... (2.22)}$$

Al realizar la sustitución de la **Ec. (2.19)** en **Ec.(2.18)** y despejando $\bar{X}$, se obtiene la expresión representativa a este método, la cual será aplicada de manera recursiva:

$$X^{(k+1)} = D^{-1} \cdot \left(b - RX^{(k)}\right), \quad \text{..... (2.23)}$$

donde $k = 0,1,2,3,\ldots,n$ y $\bar{X}^{(k)}$ es el vector solución inicial y $\bar{X}^{(k+1)}$corresponde a la aproximación posterior dependiente de $\bar{X}^{(k)}$, además D^{-1} corresponde a la matriz inversa de D donde los elementos de la diagonal estarán dados por $\frac{1}{a_{ii}}$.

Es importante mencionar que para que el método converja a la solución verdadera debe de cumplirse la siguiente condición:

$$|a_{ii}| > \sum |a_{ij}|, \quad \text{..... (2.24)}$$

en donde la norma queda definida mediante la siguiente expresión:

$$N = \sqrt{(x_1^{(k+1)} - x_1^{(k)})^2 + (x_2^{(k+1)} - x_2^{(k)})^2 +, \ldots, +(x_n^{(k+1)} - x_n^{(k)})^2}. \quad \text{..... (2.25)}$$

Dependiendo del número de incógnitas que se tenga en el sistema de ecuaciones éstas deberán despejarse una a una, de tal manera que:

$$
\begin{aligned}
X_1^{(k+1)} &= \frac{b_1 - (a_{12}X^{(k_2)} + a_{13}X^{(k_3)} + \cdots + a_{1n}X^{(k_n)})}{a_{11}} \\
X_2^{(k+1)} &= \frac{b_2 - (a_{21}X^{(k_2)} + a_{23}X^{(k_3)} + \cdots + a_{2n}X^{(k_n)})}{a_{22}} \\
X_1^{(k+1)} &= \frac{b_1 - (a_{31}X^{(k_2)} + a_{32}X^{(k_3)} + \cdots + a_{3n}X^{(k_n)})}{a_{11}} \quad , \qquad \text{.....} \qquad \textbf{(2.26)} \\
&\vdots \\
X_n^{(k+1)} &= \frac{b_n - (a_{n1}X^{(k_2)} + a_{n2}X^{(k_3)} + \cdots + a_{nn-1}X_{n-1}^{(k)}}{a_{nn}}
\end{aligned}
$$

donde $X^{(k_2)}$, $X^{(k_3)}$,…, $X^{(k_n)}$ correspondes a los elementos que conforman el vector $X^{(k)}$.

Es recomendable para este método que el vector inicial $\bar{X}^0$ sea igual a cero, para obtener así la primera aproximación de $\bar{X}^1$.

$$
\bar{X}^1 = \begin{Bmatrix} \frac{b_1}{a_{11}} \\ \frac{b_2}{a_{22}} \\ \frac{b_3}{a_{33}} \\ \vdots \\ \frac{b_n}{a_{nn}} \end{Bmatrix}
$$

Ahora este vector será sustituido en las expresiones anteriores para obtener la siguiente aproximación $\bar{X}^2$, y así sucesivamente hasta que la norma entre la diferencia de los vectores $\bar{X}^{(n+1)}$ y $\bar{X}^n$ sea menor a la tolerancia establecida.

El algoritmo de implementación se presenta a continuación, así como el diagrama correspondiente se incluye en la **Fig. 2. 4**:

1. Una vez dado el sistema de ecuaciones, representarlo mediante la **Ec. (2.19)**.
2. Verificar que los elementos de la diagonal principal de cada ecuación sean mayores al valor absoluto que el resto de sus elementos de la misma ecuación, es decir que sea una diagonal pesada. $|a_{ii}| > \sum|a_{ij}|$.

3. Obtener $\bar{X}^1$ mediante:

$$\bar{X}^1 = \begin{Bmatrix} \frac{b_1}{a_{11}} \\ \frac{b_2}{a_{22}} \\ \frac{b_3}{a_{33}} \\ \vdots \\ \frac{b_n}{a_{nn}} \end{Bmatrix}$$

4. Realizar la segunda iteración $(k = 1)$ utilizando la **Ec. (2.26)** y los elementos del vector $\bar{X}^1$ para obtener $\bar{X}^2$
5. Obtener la norma entre la diferencia de dos vectores consecutivos y verificar que sea menor a la tolerancia establecida:

$$N = \sqrt{(x_1^{(k+1)} - x_1^{(k)})^2 + (x_2^{(k+1)} - x_2^{(k)})^2 +, \dots, +(x_n^{(k+1)} - x_n^{(k)})^2} \quad , \quad \text{.....} \quad \textbf{(2.27)}$$

$$N > TOL \ .$$

6. De no cumplirse el **paso (5)** realizar las iteraciones necesarias hasta que se cumpla la condición.
7. La mejor aproximación para el sistema es $\{X_i^{k+1}\}$

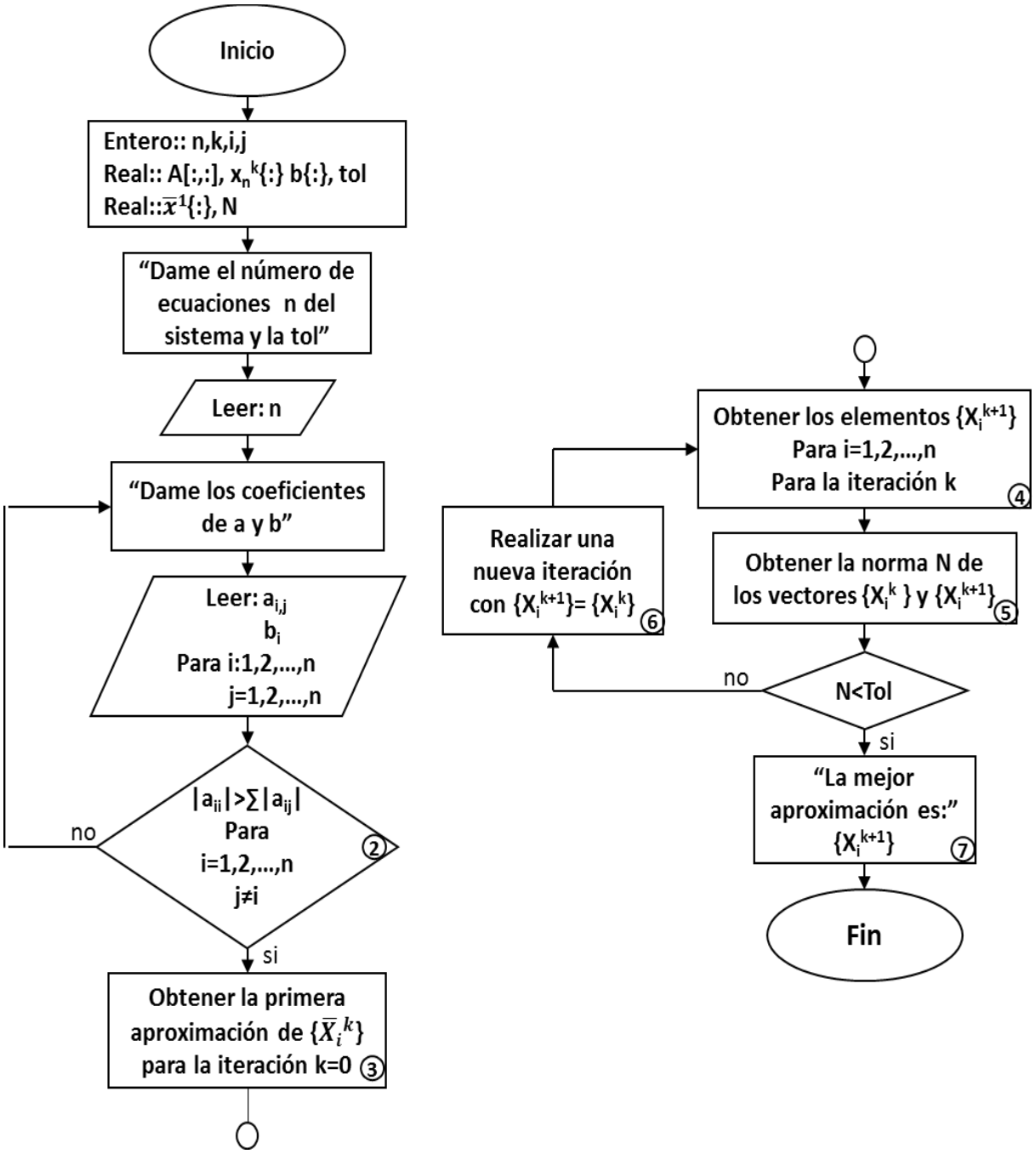


Fig. 2. 9
Método de Jacobi

2.2.2 Método de Gauss-Seidel

Es una técnica utilizada para dar solución a sistemas de ecuaciones lineales, es altamente implementado cuando se tiene gran número de ecuaciones, parte de proponer valores iniciales y después itera para obtener mejores aproximaciones de la solución de la raíz, teniendo presente un error por redondeo.

Supóngase un sistema de n ecuaciones de forma:

$$[A]\{X\} = \{B\} \ , \quad \text{.....} \qquad \textbf{(2.18)}$$

donde $[A]$ estará conformada por los elementos del sistema de ecuaciones, $\{B\}$ está formado por el término independiente de las ecuaciones y $\{X\}$ es el vector solución que deseamos encontrar.

Se parte de la suposición de un vector $\{X\}$ inicial para poder llevar a cabo las iteraciones necesarias hasta alcanzar el rango de tolerancia establecido, que se obtiene de:

$$\{X^0\} = \begin{Bmatrix} \frac{b_1}{a_{11}} \\ \frac{b_2}{a_{22}} \\ \vdots \\ \frac{b_n}{a_{nn}} \end{Bmatrix} ,$$

después se calculan los nuevos elementos de la matriz de forma

$$\begin{aligned} x_1^{k+1} &= \frac{1}{a_{11}}[b_1 - a_{12}x_2^k - a_{13}x_3^k] \\ x_2^k &= \frac{1}{a_{22}}[b_2 - a_{21}x_1^k - a_{23}x_3^k] \\ &\vdots \\ x_n^k &= \frac{1}{a_{nn}}[b_n - a_{n1}x_1^k - \cdots - a_{nn-1}x_{n-1}^k] \end{aligned} \ , \quad \text{.....} \qquad \textbf{(2.28)}$$

Si el problema se limita a un sistema de ecuaciones de $3x3$, y los elementos de la diagonal no son todos cero. Este método suele converger al resultado correcto si la diagonal de la

matriz $[A]$ es dominante, esto quiere decir qué el elemento $|a_{11}| > |a_{12}| + |a_{13}|$, $|a_{22}| > |a_{21}| + |a_{23}|$, y $|a_{33}| > |a_{31}| + |a_{32}|$.

El algoritmo de implementación se presenta a continuación, así como el diagrama correspondiente se incluye en la **Fig. 2. 5**:

1. Representar el sistema como el de la **Ec. (2.1)**.
2. Conformar la matriz $[A]$ con los elementos del sistema de ecuaciones
3. Formar el vector $\{B\}$
4. Obtener el vector $\{x^0\}$ para iniciar con las iteraciones más mediante la siguiente expresión:

$$\{X^0\} = \begin{Bmatrix} \frac{b_1}{a_{11}} \\ \frac{b_2}{a_{22}} \\ \vdots \\ \frac{b_n}{a_{nn}} \end{Bmatrix},$$

5. Obtener los elementos del vector x^1 que sera el resultado de la primera iteración, empleando la **Ec. (2.28)**.
6. Sustituir los nuevos valores del vector $\{X^1\}$ en la **Ec. (2.28)** para obtener un nuevo conjunto de valores para el vector $\{X^2\}$, siendo esta la segunda iteración.
7. Establecer la tolerancia deseada y aplicar la condición de la **Ec. (2.30)** mediante:

$$|[|x^{k-1}] - [x^k]| < tolerancia\,^{7}, \quad \text{.....} \quad \textbf{(2.30)}$$

8. Si la tolerancia es mayor a la establecida realizar las iteraciones necesarias, hasta cumplir con la condición.
9. La solución al sistema de ecuaciones es $\{X_i^k\}$.

[7] Donde k corresponde al número de iteraciones

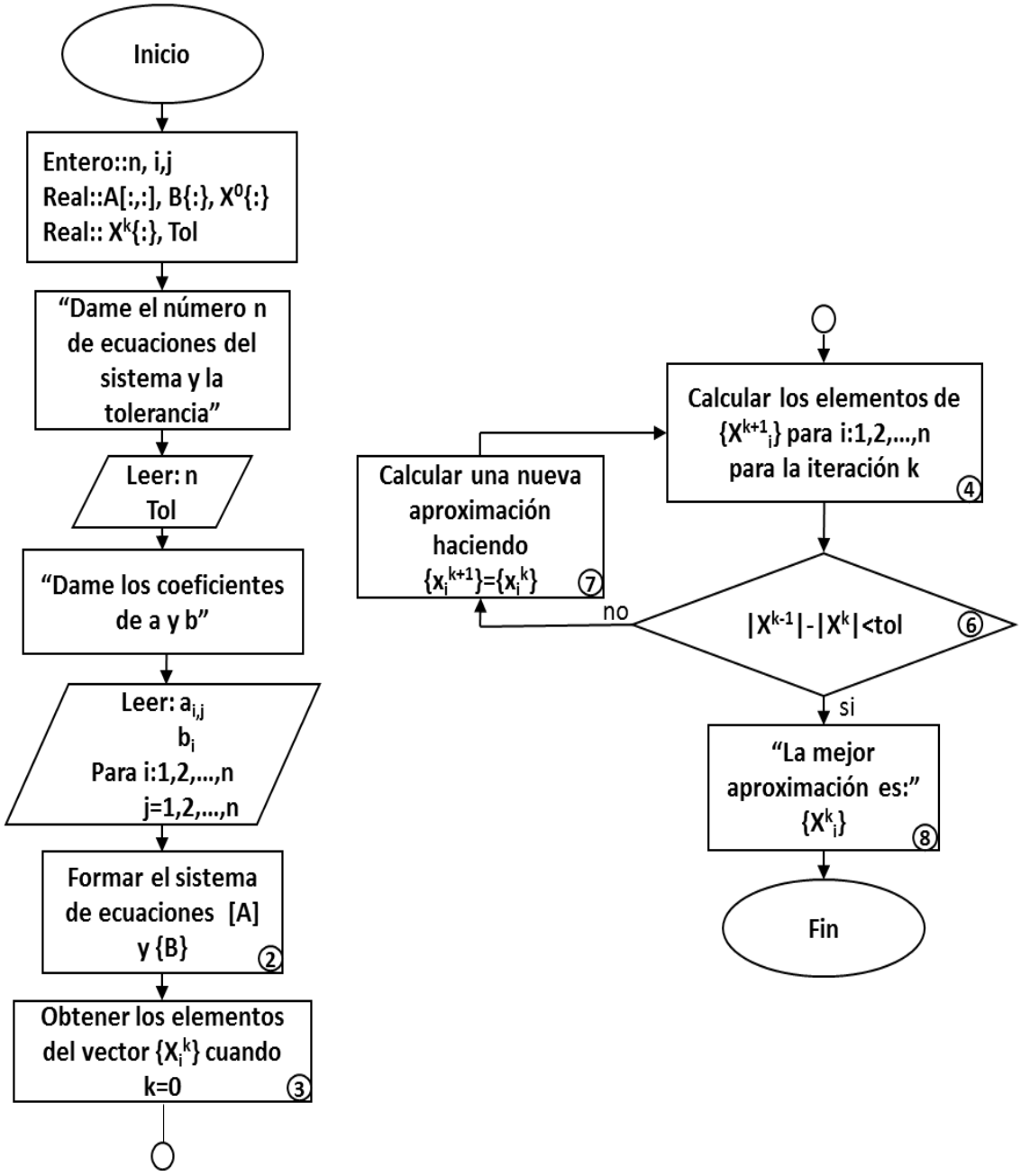


Fig. 2. 10
Método de Gauss-Seidel (G-S)

2.3.3 Método de Relajación Sucesiva (SOR)

La relajación sucesiva o SOR se emplea cuando se busca acelerar la convergencia de un sistema convergente, pues hace que el método requiera de menor tiempo de computo, este método se emplea mayormente en combinación con el de G-S utilizado para dar solución a sistemas de ecuaciones lineales, pues una vez obtenido un nuevo valor de x con la **Ec. (2.28),** ese valor es modificado mediante un ponderado de los resultados de las iteraciones anterior y actual:

$$x_n^{nuevo} = \lambda x_n^{nuevo} + (1-\lambda) x_n^{anterior} \ , \quad \text{.....} \quad \textbf{(2.31)}$$

donde λ es un factor ponderado que tiene un valor de 0 a 2.

Cuando se utiliza valores de $0 < \lambda < 1$ se trata de una *subrelajación*, empleada comúnmente para que los sistemas no convergentes converjan y cuando varía entre $1 < \lambda < 2$ se trata de una *sobrerelajación (SOR).*

La SOR no solo puede emplearse en el Método de Gauss Seidel, sino también puede introducirse en el método de Jácobi, pero dado que ofrece un menor rango de error y convergencia es preferente emplearlo y común que sea más utilizado en el Método de Gauss Seidel.

Parte esencial para la elaboración del código

subroutine Gseid (a,b,n,x,imax,es,lambda)	sum=bi
do i=1,n	do j=1,n
dummy=ai,i	if (i.ne.j) then
do j=1,n	sum=sum-ai,j*xj
ai,j=ai,j/dummy	end do
end do	xi=lambda*sum+(1.-lambda)*old
do i=1,n	if (centinela.eq.1.and. xi.ne.0) then
sum=bi	ea=abs((xi-old)/xi)*100.

```
do j=1,n
if (i.ne.j) then
sum=sum-ai,j*xj
end do
xi=sum
end do
iter=1
do
centinela=1
do i=1,n
old=xi
```

```
if (ea.gt.es) then
centinela=0
end if
end do
iter=iter+1
if (centinela.eq.1.or.iter.ge.imax)then
exit
end do
end Gseid
```

Capítulo 3

Solución a sistemas de ecuaciones no lineales

En este capítulo se buscará dar solución a sistemas de ecuaciones no lineales, empleando el método de Newton-Raphson, el cual hace uso de la función y sus derivadas para probar el criterio de convergencia, con el fin de obtener la mejor aproximación de la ecuación o del sistema de ecuaciones.

3.1 Sistema de ecuaciones no lineales

Al igual que en los sistemas lineales los no lineales buscan obtener las raíces de un conjunto de ecuaciones simultáneamente, donde la solución consta de un conjunto de valores x que hacen a todas las ecuaciones igual a cero:

$$\begin{matrix} f_1(x_1, x_2, \dots, x_n) = 0 \\ f_2(x_1, x_2, \dots, x_n) = 0 \\ \vdots \\ f_n(x_1, x_2, \dots, x_n) = 0 \end{matrix} . \quad \text{.....} \quad \textbf{(3.1)}$$

La mayoría de los métodos empleados para para dar solución a este tipo de sistemas son extensiones de los métodos que resuelven ecuaciones simples.

3.2 Métodos iterativos de Newton-Raphson

Con este método iterativo se puede encontrar la aproximación a la solución de ecuaciones no lineales. El método parte de una sola ecuación no lineal con una incognita, a la cual se le asigna un valor inicial que se introduce en una expresión relacionada con la ecuación para obtener un resultado, y a su vez este valor se introduce a la misma expresión, para obtener un nuevo resultado, y así sucesivamente hasta obtener la aproximación deseada.

Se recurre a la **Ec. (3.2)** para dar solución al problema antes planteado.

$$X_{i+1} = X_i - \frac{f(x_i)}{f'(x_i)} . \quad \text{.....} \quad \textbf{(3.2)}$$

Es importante que antes de intentar resolver el sistema mediante la ecuación general **Ec. (3.3)**, se aplique el criterio de convergencia para verificar que este tenga solución, empleando la ecuación:

$$\boldsymbol{G'(x_0) = \left| \frac{f(x_0) * f''(x_0)}{f'(x_0)^2} \right| < 1} \ , \quad \text{.....} \quad \textbf{(3.3)}$$

Una vez que ya se tiene la certeza de que dicho sistema tiene solución, se procede a aplicar la expresión general n veces hasta que el error (ε) entre el valor X_i y X_{i+1}sea mínimo:

$$\varepsilon = \left| \frac{x_{i+1} - x_i}{x_{i+1}} \right| = \left| \frac{Valor\ actual - Valor\ anterior}{Valor\ anterior} \right| \ , \quad \text{.....} \quad \textbf{(3.4)}$$

El algoritmo de implementación será presentado a continuación, así como el diagrama correspondiente se incluye en la **Fig. 3. 1:**

1. Suponer un valor inicial que sea cercano a la raíz que soluciona el sistema x_0.
2. Llevar al sistema a la forma de la **Ec. (3.1)**
3. Verificar que el grado del polinomio ≥ 2
4. Obtener la primera y segunda derivada de la función, f'(x_0) =0 y f''(x_0) =0
5. Aplicar el criterio de convergencia para ver si el sistema tiene solución, empleando la **Ec. (3.3)**.
6. Si no se cumple con el criterio de convergencia el sistema no puede ser resuelto, en caso contrario seguir **paso 7.**
7. Considerar a $x_0 = x_i$
8. Calcular y hacer: $f\ (x_i) = 0\ y\ f'(x_i) = 0$
9. Obtener x_{i+1} mediante la **Ec. (3.2).**
10. Calcular el error haciendo uso de **Ec. (3.4).**
11. Verificar que se cumpla $\varepsilon < Tol$
12. En caso de no cumplirse la condición, hacer una nueva iteración haciendo $x_{i+1} = x_i$ y regresar al **paso 7**.
13. Si la condición anterior se cumple la mejor aproximación es: x_{i+1}**.**

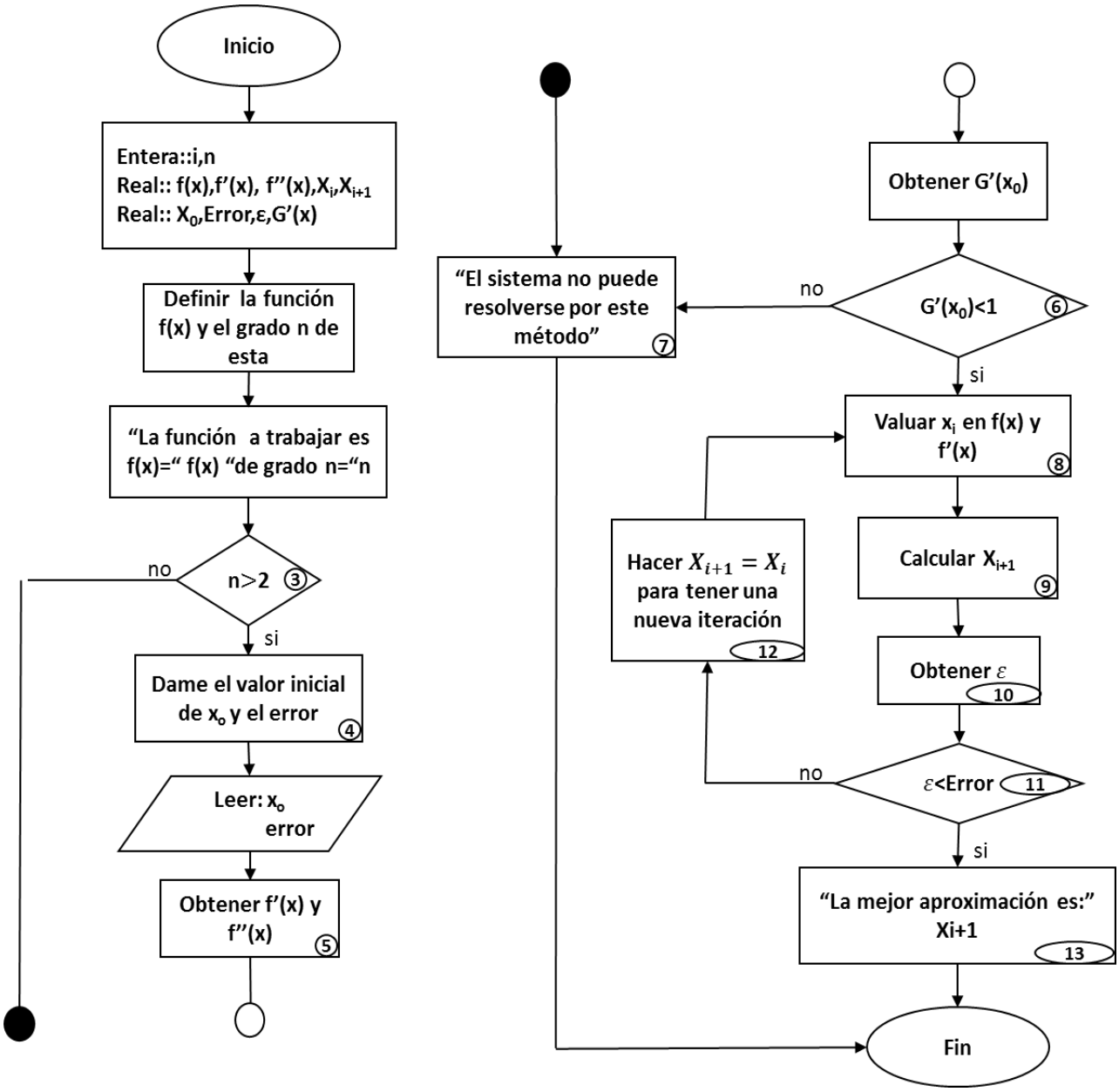


Fig. 3. 2
Método de Newton-Raphson

Capítulo 4

Interpolación numérica

La interpolación es una técnica muy antigua, que hasta la fecha sigue siendo utilizada para conocer los valores intermedios en una relación; donde se hacen aproximaciones a partir de un polinomio de interpolación de grado uno para el caso de la lineal, de grado dos cuando se trata de la interpolación de Lagrange y mínimos cuadrados, de tercer grado cuando se trabaja con el spline cúbico y de grado n cuando se trata de la interpolación polinomial; es importante no olvidar que se está trabajando con aproximaciones por lo que en sus resultados estará presente cierto grado de error.

4.1 Interpolación lineal y doble

La interpolación lineal es un método utilizado para estimar los valores que toma una función en cierto intervalo$[x_n, x_{n+1}]$, gráficamente consiste en conectar dos puntos por medio de una línea recta **Fig. 4. 1**.

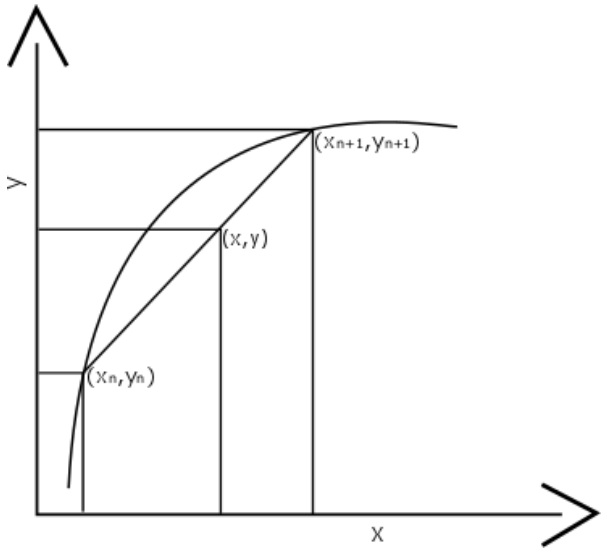


Fig. 4. 6
Esquema que representa la interpolación lineal

Dado que las variaciones en una relación lineal permanecen constantes, se puede estimar de relación de proporcionalidad mediante la **Ec. (4.1)**:

$$\frac{y - y_n}{x - x_n} = \frac{y_{n+1} - y_n}{x_{n+1} - x_n}, \quad \text{.....} \quad \textbf{(4.1)}$$

a partir de esta relación, puede ser despejada la variable que resulte de nuestro interés, que en este caso seria “y” resultando así la **Ec. (4.2)**:

$$y = y_n + \frac{(y_{n+1} - y_n)(x - x_n)}{x_{n+1} - x_n}, \quad \text{.....} \quad \textbf{(4.2)}$$

la cual es la representativa para llevar a cabo el proceso de Interpolación Lineal, siempre y cuando, sean conocidos los valores extremos de un comportamiento constante.

Además de la Interpolación Lineal también existe la Interpolación Doble, que resulta ser una extensión de la primera, pero que es utilizada para interpolar funciones de dos variables, prácticamente consiste en interpolar dos veces, donde deberán ser conocidos los valores de frontera, por lo que la expresión resulta ser la misma que para la Interpolación Lineal **Ec. (4.2)**.

Al utilizar este método de interpolación se presenta un pequeño error respecto al valor de la función verdadera, pero suele ser una muy buena aproximación a que si se tomara el valor más cercano que se tiene.

A continuación, se presenta el algoritmo de implementación, así como su diagrama correspondiente se incluye en la **Fig. 4. 2:**

1. Identificar los puntos en los extremos x_n y $x_{\mathrm{n+1}}$ así como el valor que nos interesa conocer x.
2. Es importante verificar que el valor de x dado se encuentre entre x_n y $x_{\mathrm{n+1}}$
3. En caso de no cumplirse la condición anterior el método no puede ser aplicado a menos que se introduzcan nuevamente valores que satisfagan dicha condición.
4. Sustituir los datos en **Ec. (4.2)**, que es la ecuación representativa de la Interpolación lineal.
5. Si se requiere realizar una nueva interpolación para conocer otra variable, volver a identificar los datos de entrada y sustituirlos en **Ec. (4.1)**.

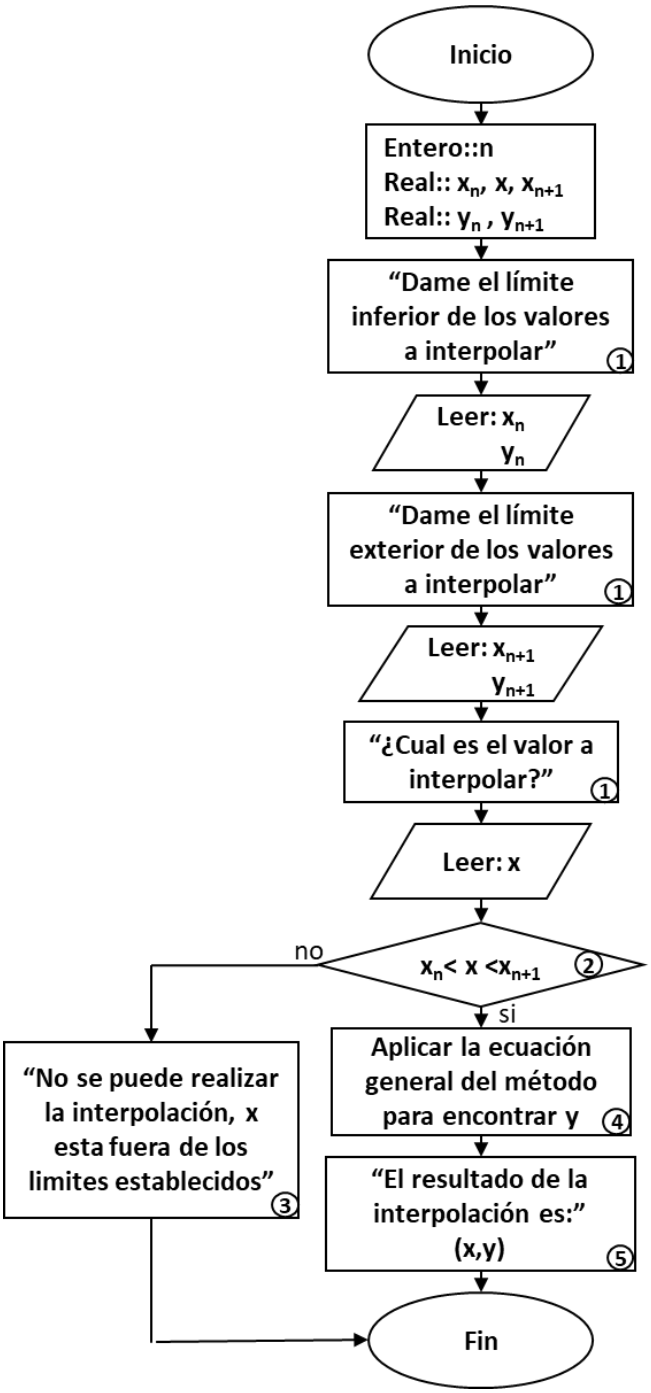


Fig. 4. 7
Interpolación Lineal

4.2 Método de Lagrange

El polinomio de Lagrange sirve para interpolar un conjunto de puntos de *n+1* datos.

$$(x_0, y_0),(x_1, y_1),\ldots,(x_n, y_n) \qquad \text{Para } j = 0,1, \ldots n$$

Donde todos los elementos x_i se consideran distintos, el polinomio interpolador en la forma de Lagrange es representado por la **Ec. (4.3)**:

$$P(x) = \sum_{i=0}^{n} y_i \; l_i(x) \;, \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \textbf{(4.3)}$$

donde l_i es la base polinómica de Lagrange y es evaluada mediante:

$$l_0(x) = \frac{(x - x_1)(x - x_2) \ldots (x - x_j)}{(x_0 - x_1)(x_0 - x_2) \ldots (x_0 - x_{j)}} \;,$$

$$l_1(x) = \frac{(x - x_0)(x - x_2) \ldots (x - x_j)}{(x_1 - x_0)(x_1 - x_2) \ldots (x_1 - x_{j)}} \;,$$

$$\vdots$$

$$l_i(x) = \prod_{j=0.i\neq j}^{n} \frac{(x - x_j)}{(x_i - x_j)} \;, \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \textbf{(4.4)}$$

resultando así la expresión general que representa a la Interpolación de Lagrange.

Se presenta a continuación el algoritmo para la implementación del método, así como el diagrama correspondiente es presentado en la **Fig. 4. 3:**

1. Identificar el conjunto de n pares ordenados (x, y), así como el valor correspondiente de x para el que se desea conocer su respectiva y.
2. Obtener los elementos $l_i(x)$ desde $j = 1$ hasta n, excepto para cuando $j = i$ recurriendo a **Ec. (4.4)**:
3. Una vez obtenidos los elementos $l_i(x)$ multiplicar uno a uno por su respectivo elemento y_i, para después sumarlo desde $i = 0$ hasta $i = n$ para completar el polinomio de Legendre, presentado por la **Ec. (4.3)**:
4. Sustituir el valor de x al cual se busca conocer su respectiva y en el polinomio obtenido en el paso anterior.

5. El resultado de la interpolación, mediante el método de Lagrange es (x, y).

Parte esencial para la elaboración del código (Para calcular una sola predicción de grado n-esimo, dode n+1 es el número de datos).

```
function lagrang(x,y,n,xx)
sum=0
do i=0,n
if (i.ne.j) then
product=product*(xx-xj)/(xi-xj)
end if
end do
end do
sum0sum+product
end do
lagrang=sum
end lagrang
```

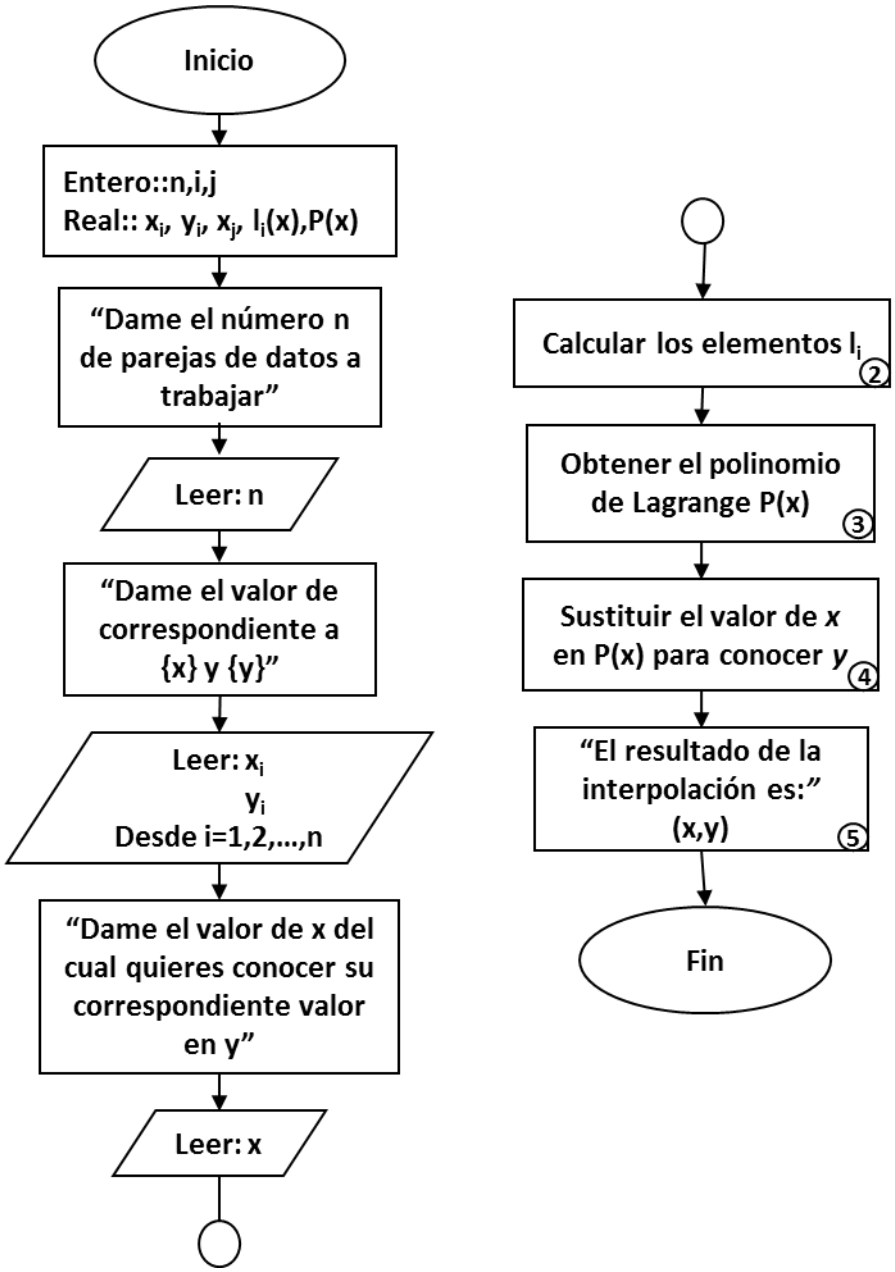


Fig. 4. 8
Método de Lagrange

4.3 Método del Spline Cúbico

La interpolación polinómica se basa en la sustitución de valores de una función por un polinomio que represente dichos valores, cuando en número de puntos aumenta, también lo hace el grado del polinomio de aproximación, arrastrando consigo un incremento en el error.

La interpolación con splines es utilizada cuando se presentan estos casos en que los polinomios son de grado alto, entonces son sustituidos por polinomios de grado menor con sus respectivos subintervalos, y en particular se recurre al Spline Cúbico cuando este es de grado tres.

La idea central de este método consiste que, en vez de usar un solo polinomio para interpolar datos, se puedan usar segmentos de polinomios y unirlos adecuadamente para formar la interpolación.

El Spline Cúbico es muy empleado, debido a que proporciona un excelente ajuste a los puntos tabulados.

Sea el intervalo $[t_0, t_n]$, dividido en pequeños intervalos $[t_0, t_1]$, $[t_1, t_2]$,…, $[t_{n-1}, t_n]$, y f una función definida por un polinomio cúbico, y f_i el polinomio cúbico que representa a la función f en el intervalo $[t_n, t_{n+1}]$ quedando representada en la **Ec. (4.5)**:

$$f(x) = \begin{cases} f_0(x) & x \in [t_0, t_1) \\ f_1(x) & x \in [t_1, t_2) \\ & \cdot \\ & \cdot \\ & \cdot \\ f_{n-1}(x) & x \in [t_{n-1}, t_n) \end{cases} \quad . \quad \text{.....} \quad \textbf{(4.5)}$$

Una vez que se forcé a que pasé por los puntos dados, se procede a calcular su primera derivada y evaluarla en tales puntos e igualarla con ellos.

Después se realiza el mismo proceso, para la segunda derivada calculada previamente, considerando que se busca formar un sistema de ecuaciones que sea

cuadrado, las ecuaciones faltantes se obtienen al evaluar la segunda derivada en los límites del intervalo e igualándolos a cero, quedando de la siguiente manera:

$$f(x) = \begin{cases} f_0(x) = f_0 \\ f_1(x) = f_1 \\ f_2(x) = f_2 \\ f_3(x) = f_3 \\ \vdots \\ f_{i-1}(x) = f_{i-1} \\ f'(x)_1 = f'(x)_2 \\ \vdots \\ f''(x)_1 = f''(x)_2 \\ \vdots \\ f''(x_0) = 0 \\ f''(x_{0i}) = 0, \end{cases} \quad \text{.....} \quad \textbf{(4.6)}$$

y

$$\begin{matrix} f_{i-1}(x_i) = f_i(x_i) \\ f'_{i-1}(x_i) = f_i'(x_i) \\ f''_{i-1}(x_i) = f_i''(x_i) \end{matrix} . \quad \text{.....} \quad \textbf{(4.7)}$$

La manera de obtener cada uno de los polinomios cúbicos $f(x)$ para cada intervalo es mediante el uso de la **Ec. (4.8):**

$$\begin{aligned} f_{n-1}(x) = {} & \frac{f''_i(x_{i-1})}{6(x_i - x_{i-1})}(x_i - x)^3 + \frac{f''_i(x_i)}{6(x_i - x_{i-1})}(x - x_{i-1})^3 \\ & + \left[\frac{f(x_{i-1})}{(x_i - x_{i-1})} - \frac{f''(x_{i-1})(x_i - x_{i-1})}{6}\right](x_i - x) \\ & + \left[\frac{f(x_i)}{(x_i - x_{i-1})} - \frac{f''(x_i)(x_i - x_{i-1})}{6}\right](x - x_{i-1}) \ , \end{aligned} \quad \text{.....} \quad \textbf{(4.8)}$$

en la cual, solo se tienen presentes dos incógnitas (las segundas derivadas en los extremos de cada intervalo) las cuales se obtienen al emplear la **Ec.(4.9)** :

$$\begin{aligned} & (x_i - x_{i-1})f''(x_{i-1}) + 2(x_{i+1} - x_{i-1})f''(x_i) + \\ & (x_{i+1} - x_i)f''(x_{i+1}) = \frac{6}{(x_{i+1} - x_i)}[f(x_{i+1}) - f(x_i)] \ , \end{aligned} \quad \text{.....} \quad \textbf{(4.9)}$$

de esta ecuación para cada nodo o intervalo trabajado surgirá un sistema de ecuaciones con dos incógnitas que puede resolverse usando el método de Eliminación Gaussiana.

De esta manera al resolverse la **Ec. (4.8)** se obtienen las $n-1$ ecuaciones simultaneas con $n-1$ incógnitas; es importante no olvidar que las segundas derivadas en los extremos son igual a cero, las demás expresiones se obtienen al aplicar las condiciones restantes de la **Ec. (4.6)** y **Ec. (4.7).**

Se presenta a continuación el algoritmo para la implementación de este método, así como su respectivo diagrama esquematizado en **Fig. 4. 4**:

1. Dado el conjunto de pares ordenados identificar los respectivos $(x_i, f(x)_i)$ así como el valor de x para el cual se quiere conocer su $f(x)$.
2. Aplicar la **Ec. (4.9) para obtener las dos funciones correspondientes de los extremos** y resolver el sistema para obtener las segundas derivadas.
3. Y junt
4. Sustituir las segundas derivadas obtenidas en el **paso 2**., en la **Ec. (4.8)** para obtener el polinomio correspondiente al intervalo.
5. Aplicar el **paso 2** y **paso 3** hasta los $n-1$ datos para obtener los polinomios de tercer grado representativos para cada intervalo (con esto se está forzando a que el Spline pase por cada uno de los puntos).
6. Calcular la primera derivada del sistema de ecuaciones $f(x)$, y evaluarla en el intervalo e igualarlas.

$$f'(x)_1 = f'(x)_2$$
$$\vdots$$

7. Calcular la segunda derivada y realizar lo mismo que con la primera derivada:

$$f''(x)_1 = f''(x)_2$$
$$\vdots$$

ahora tenemos *m* ecuaciones y *m+2* incógnitas

8. Obtener las segundas derivadas en las fronteras:

$$f''(x_0) = 0$$
$$f''(x_n) = 0$$

9. Ahora tenemos un sistema matricial cuadrado con *m* ecuaciones y *m* incógnitas que puede resolverse por el método adecuado para sistemas tridiagonales.
10. Si se quiere conocer $f(x)$ de una x dada, esa se obtendrá empleando el polinomio adecuado al intervalo donde se encuentre.

Parte esencial para la elaboración del código

subroutine spline(x,y,n,xu,yu,dy,d2y)	c1=d2xi-1/6/(xi-xi-1)
local en,fn,gn,rn,d2xn	c2=d2xi/6/(xi-xi-1)
call tridiag(x,y,n,e,f,g,r)	c3=yi-1/(xi-xi-1)-d2xi-1*(xi-xi-1)/6
call decomp(e,f,g,n-1)	c4=yi/(xi-xi-1)-d2xi*(xi-xi-1)/6
call subst (e,f,g,r,n-1,d2x)	t1=c1*(xi-xu)^3
call interpol (x,y,n,d2x,xu,yu,dy,d2y)	t2=c2*(xu-xi-1)^3
end spline	t3=c3*(xi-xu)
subroutine tridiag (x,y,n,e,f,g,r)	t4=c4*(xu-xi-1)
f1=2*(x2-x0)	yu=t1+t2+t3+t4
g1=(x2-x1)	t1=-3*c1*(xi-xu)^2
r1=6/(x2-x1)*(y2-y1)	t2=3*c2*(xu-xi-1)^2
r1=r1+6/(x1-x0)*(y0-y1)	t3=-c3
do i=2,n-2	t4=c4
ei=(xi-xi-1)	dy=t1+t2+t3+t4
fi=2*(xi+1-xi-1)	t1=6*c1*(xi-xu)
gi=(xi+1-xi)	t2=6*c2*(xu-xi-1)
ri=6/(xi+1-xi)*(yi+1-yi)	d2y=t1+t2
end do	flag=1
en-1=(xn-1-x-2)	else
fn-1=2*(xn-xn-2)	i=i+1

```
rn-1=6/(xn-xn-1)*(yn-yn-1)
rn-1=rn-1+6/(xn-1-xn-2)*(yn-2-yn-1)
end tridiag
soubroutine interpol (x,y,n,d2x,xu,yu,dy,d2y)
flag=0
i=1
do
if ( xu.ge.xi-1.and.xu.le.xi) then
end if
if (i.eq.n+1.or.flag.eq.1) then
exit
end do
if (flag.eq.0) then
print*,"fuera de rango"
pause
end if
end interpol
```

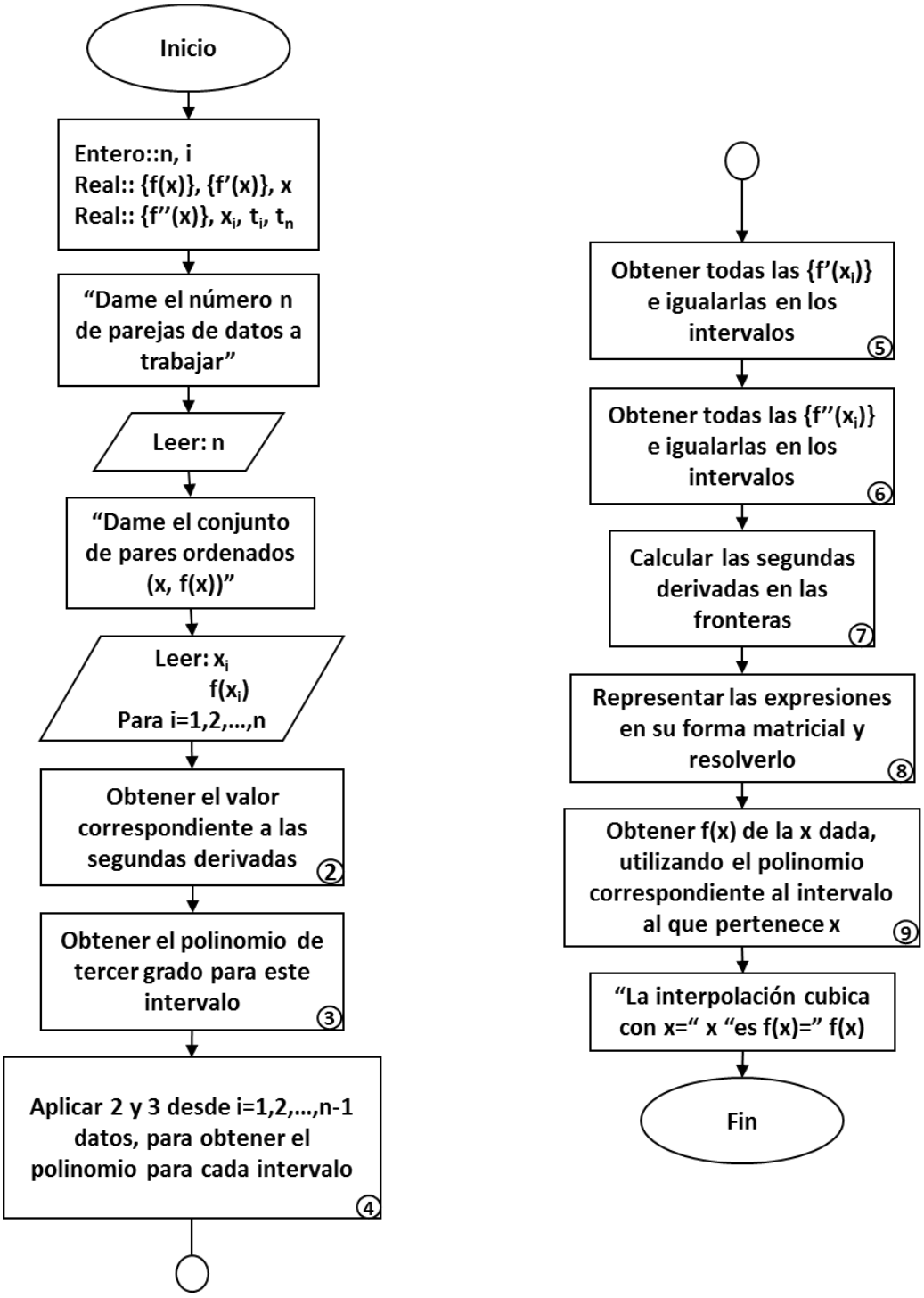


Fig. 4. 9
Método del Spline Cúbico

4.4 Mínimos cuadrados

El ajuste por mínimos cuadrados consiste en aproximar un polinomio de grado n a uno de menor grado, mientras que en el caso más sencillo y más utilizado consiste en aproximar un conjunto de datos establecidos a un polinomio de grado uno el cual corresponde a la expresión representativa de una recta.

Ajuste de una recta por mínimos cuadrados

Este método es aplicado cuando se tiene un conjunto de datos arbitrarios n, los cuales se quieren ser representarlos mediante su representación más simple, una recta, la cual representará al conjunto de puntos dados mediante 0su ecuación representativa **Ec. (4.10)**:

$$y = a_1 x + a_0 \ , \qquad \text{.....} \qquad \textbf{(4.10)}$$

dónde: a_1 corresponde a la pendiente de la recta y a_0 es la ordenada en el origen.

La pendiente a_1 se obtiene empleando la **Ec. (4.11)**:

$$a_1 = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum y_i^2)} \ , \qquad \text{.....} \qquad \textbf{(4.11)}$$

y la ordenada al origen a_0 se obtendrá mediante **Ec. (4.12)**:

$$a_0 = \bar{y} - a_1 \bar{x} \ , \qquad \text{.....} \qquad \textbf{(4.12)}$$

donde $\bar{x}$ y $\bar{y}$ son las medias de x y y.

Regresión Polinomial

Este método se aplica cuando la representación gráfica de los datos establecidos no es preciso que se aproxime a una recta, entonces lo que se busca es ajustar a los datos a un polinomio mediante una regresión polinomial.

Para esto, es necesario partir de la **Ec. (4.13)**, la cual representa a un polinomio de grado m:

$$y = a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m + e \ , \qquad \text{.....} \qquad \textbf{(4.13)}$$

Recurriendo a la expresión que representa la suma de los cuadrados de los residuos de la ecuación anterior resultando:

$$S_r = \sum_{i=1}^{n} (y_i - (a_0 + a_1 x_i + a_2 x_i^2 + \cdots + a_m x_i^2))^2 \ , \quad \text{.....} \quad \textbf{(4.14)}$$

una vez establecida la **Ec. (4.14),** se prosigue a derivarla con respecto a cada uno de los coeficientes desconocidos del polinomio $(a_0, a_1, a_2, \ldots, a_m)$.

$$\begin{aligned}
\frac{\delta S_r}{\delta a_0} &= -2 \sum (y_i - a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m) \ , \\
\frac{\delta S_r}{\delta a_1} &= -2 \sum x_i (y_i - a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m) \ , \\
\frac{\delta S_r}{\delta a_2} &= -2 \sum x_i^2 (y_i - a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m) \ , \\
&\vdots \\
\frac{\delta S_r}{\delta a_m} &= -2 \sum x_i^m (y_i - a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m) \ .
\end{aligned} \quad \text{.....} \quad \textbf{(4.15)}$$

Después se iguala a cero y se reordenan cada una de las ecuaciones para obtener un conjunto de ecuaciones normales de la forma:

$$\begin{aligned}
(n) a_0 + \left(\sum x_i\right) a_1 + \left(\sum x_i^2\right) a_2 + \cdots + \left(\sum x_i^m\right) a_m &= \sum y_i \ , \\
\left(\sum x_i\right) a_0 + \left(\sum x_i^2\right) a_1 + \left(\sum x_i^3\right) a_2 + \cdots + \left(\sum x_i^{m+1}\right) a_m &= \sum x_i y_i \ , \\
\left(\sum x_i^2\right) a_0 + \left(\sum x_i^3\right) a_1 + \left(\sum x_i^4\right) a_2 + \cdots + \left(\sum x_i^{m+2}\right) a_m &= \sum x_i^2 y_i \ , \\
&\vdots \\
\left(\sum x_i^m\right) a_0 + \left(\sum x_i^{m+1}\right) a_1 + \left(\sum x_i^{m+2}\right) a_2 + \cdots + \left(\sum x_i^{m+m}\right) a_m &= \sum x_i^m y_i \ ,
\end{aligned} \quad \text{....} \quad \textbf{(4.16)}$$

una vez obtenido el sistema de ecuaciones **(4.16),** éstas pueden ser representadas de forma matricial para resolverse mediante el método de *Eliminación Gaussiana* y así obtener el valor correspondiente a cada incógnita que nos interese conocer.

A continuación, se presenta el algoritmo de implementación para una recta por mínimos cuadrados, así como su diagrama correspondiente se incluye en la **Fig. 4. 5:**

1. Identificar el número n de pares ordenados $(x. y)$.

2. Sumar uno a uno los elementos de x e y desde i hasta n.

$$\sum x_i$$

y

$$\sum y_i$$

3. Obtener la sumatoria del producto xy, desde i hasta el elemento n.

$$\sum x_i y_i$$

4. Calcular la sumatoria del cuadrado de los elementos x e y.

$$\sum x_i^2$$

y

$$\sum y_i^2$$

5. Evaluar el valor de a_1 que corresponde a la pendiente mediante la **Ec. (4.11):**
6. Para la ordenada al origen a_0 deberà calcular las medias de x e y mediante:

$$\bar{x} = \frac{\sum x_i}{n}$$

y

$$\bar{y} = \frac{\sum y_i}{n}$$

7. Sustituir los valores en la expresión para obtener a_0 empleando la **Ec. (4.12).**
8. Sustituir los valores obtenidos de a_1 y a_0 en la **Ec. (4.10).**
9. Si se cuenta con un valor de x, este puede sustituirse en la **Ec. (4.10)** para conocer su respectiva y.

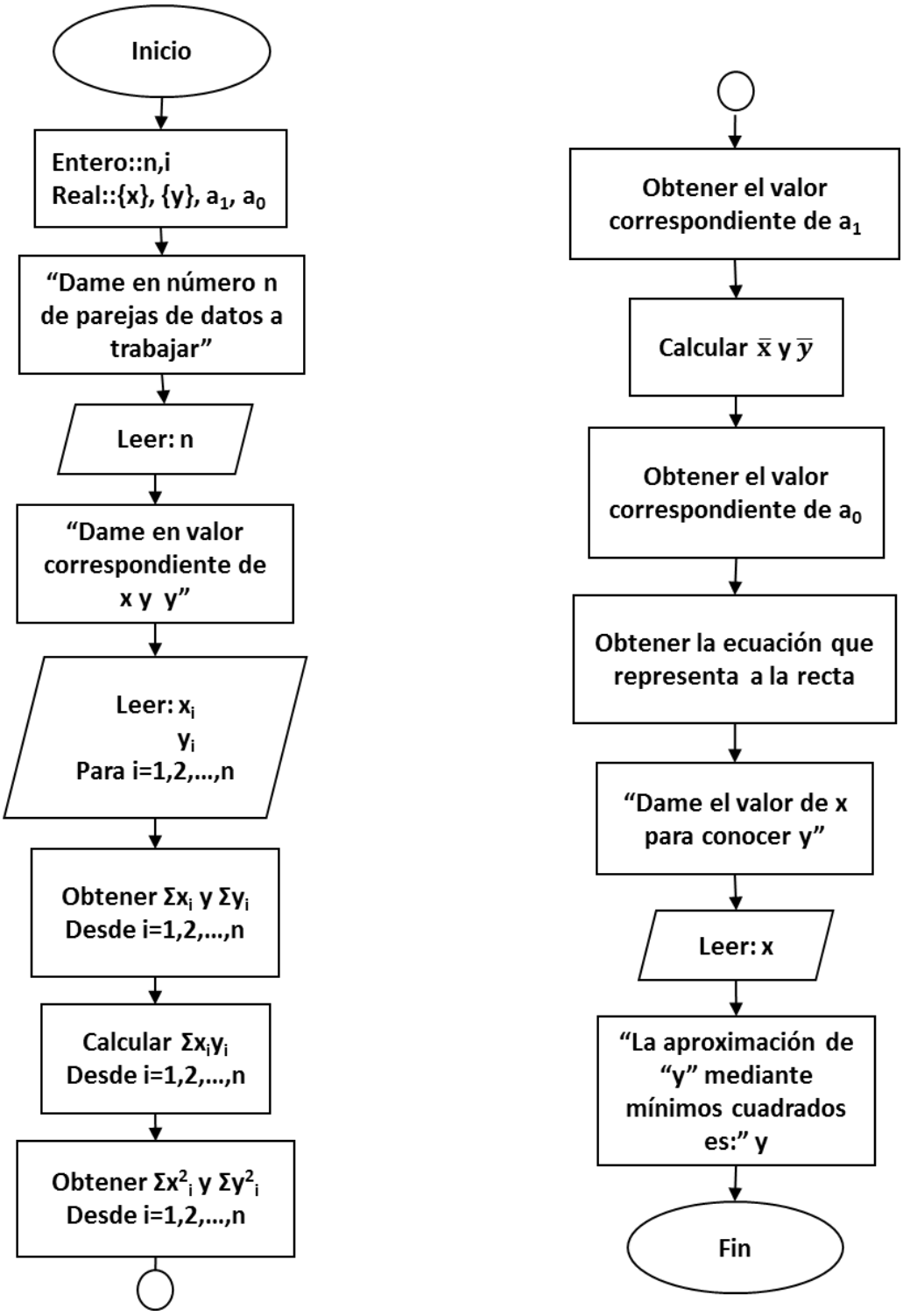


Fig. 4. 10
Método de mínimos cuadrados

Capítulo 5
Derivación e integración numérica

En este capítulo se abordará lo correspondiente a la derivación numérica mediante Diferencias Finitas e integración numérica recurriendo a los métodos de Newton-Cotes, Simpson 1/3 y 3/8 y Cuadratura Gaussiana, donde al implementar las correspondientes ecuaciones se puede conocer el valor de la derivada o la integral para un punto dado.

La función a diferenciar o integrar puede estar representada en cualquiera de las tres siguientes formas:

1. Como función continua simple (polinomio, exponencial o trigonométrica).
2. Como una función continúa complicada difícil o imposible de realizar directamente.
3. Como una función tabulada donde los valores de x y $\mathrm{f(x)}$ están representados como un conjunto de puntos (datos experimentales o de campo).

5.1 Derivación Numérica

En el ámbito matemático la derivada representa la razón de cambio de una variable dependiente con respecto a una variable independiente. Resultando ser una técnica de análisis numérico para calcular mediante una aproximación la derivada de una función $f(x)$, así, cuando la aproximación sea mayor a cero se trabaja con diferencias hacia a delante y si es menor a cero las diferencias son hacia atrás, sin olvidar considerar el error existente, por eso, si se quiere disminuir para una mejor aproximación debe recurrirse a las diferencias centrales que resulta ser un promedio de las dos anteriores.

5.1.1 Diferencias Finitas

El método de Diferencias finitas permite resolver ecuaciones diferenciales en derivadas parciales definidas. Las diferencias finitas divididas pueden ser representadas por la siguiente **Ec. (5.1)** y **Ec. (5.2)**:

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} + O(x_{i+1} - x_i) \quad \text{.....} \quad \textbf{(5.1)}$$

o

$$f'(x_i) = \frac{\Delta f_i}{h} + O(h) \ , \qquad \text{.....} \quad \textbf{(5.2)}$$

donde Δf_i corresponde a la primera diferencia hacia adelante y h es el incremento. A esta **Ec. (5.2)** se le conoce como diferencia "hacia adelante" ya que usa datos en i e $i+1$ para obtener la derivada, por lo que el término $\frac{\Delta f}{h}$ es conocido como primera diferencia finita dividida.

Esta diferencia dividida hacia adelante es solo una de tantas que pueden desarrollarse partiendo de la serie de Taylor a la aproximación de derivadas numéricas.

Aproximación de la primera derivada con diferencias finitas hacia atrás

Al expandir la serie de Taylor hacia atrás y truncando en la primera derivada para calcular un valor anterior conociendo un valor actual, es representada mediante la **Ec. (5.3)**:

$$f'(x_i) \cong \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} = \frac{\nabla f_1}{h} \ , \quad \text{.....} \quad \textbf{(5.3)}$$

cuya ecuación es conocida como primera diferencia dividida hacia atrás.

Aproximación de la primera derivada con diferencias centradas hacia atrás.

Una tercera forma de aproximar la primera derivada es mediante la **Ec. (5.4)**:

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - O(h^2) \ , \qquad \text{.....} \quad \textbf{(5.4)}$$

que representa a las diferencias centradas con espaciamiento uniforme, cabe destacar que esta ecuación presenta una mayor exactitud para el cálculo de la derivada que las expresiones anteriores, debido al error que maneja $(h2)$.

El algoritmo de implementación se presenta a continuación, así como el diagrama correspondiente se incluye en la **Fig. 5. 1:**

1. Definir la función $f(x)$,
2. Solicitar el espaciamiento h
3. identificar el punto x_i en el cual se quiere calcular la derivada.

4. elegir el método a utilizar

4.A Algoritmo de implementación de diferencias fintas hacia adelante

4.1 Obtener el valor x_{i+1}

$$x_{i+1} = x_i + h$$

4.2 Valuar la función en el punto x_i y x_{i+1}

4.3 Aplicar la expresión **(5.1)** para obtener $f'(x_i)$

4. B Algoritmo de implementación de diferencias fintas hacia atrás

4.1 Obtener el valor x_{i-1}

$$x_{i-1} = x_i - h$$

4.2 Obtener la función valuada en el punto x_i y x_{i-1}

4.3 Aplicar la Ec. **(5.3)** para obtener $f'(x_i)$

4.C Algoritmo de implementación de diferencias finitas centradas

4.1 Obtener el valor x_{i-1} y x_{i+1}

$$x_{i-1} = x_i - h$$

$$x_{i+1} = x_i + h$$

4.2 Obtener la función valuada en el punto x_{i-1} y x_{i+1}

4.3 Aplicar la **Ec. (5.4)** para obtener $f'(x_i)$

5 La aproximación de la deriva en diferencias finitas es: $f'(x_i)$

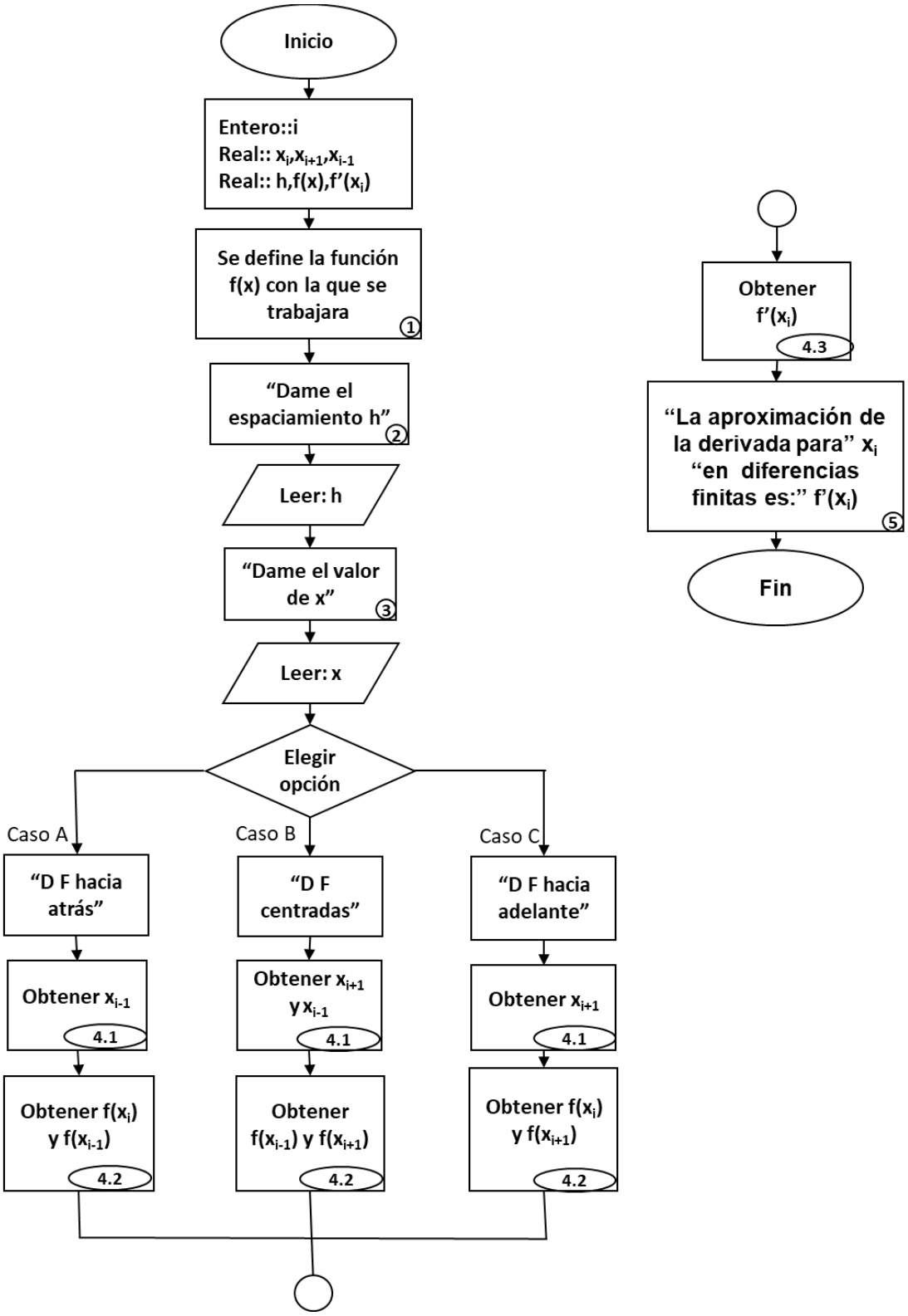


Fig. 5. 11
Método de Diferencias Finitas

5.2 Integración Numérica

La integración numérica consiste en encontrar la mejor aproximación del área bajo la curva que representa a una función $f(x)$, la cual ha sido determinada mediante datos experimentales o a partir de una expresión matemática establecida.

Estos métodos son empleados para integrar funciones dadas mediante una tabla o de forma analítica, la integración numérica puede ahorrar tiempo y esfuerzo si es que solo se desea conocer en valor numérico de la integral.

En este capítulo se explicarán los métodos de Newton-Cotes (Regla del rectángulo, Regla del trapecio, Método de Simpson 1/3 y Método de Simpson 3/8), siendo el caso particular, cuando los datos a integrar están separados de manera uniforme para dar solución al problema planteado.

5.2.1 Formula de Newton-Cotes

Corresponden a un conjunto de fórmulas de integración numérica del tipo interpolatorio, ya que, se busca reemplazar una función complicada por un polinomio de aproximación que resulte más fácil de integrar, evaluando la función en una serie de puntos equidistantes, para así hallar un valor aproximado a la integral. En cuanto más intervalos sea dividida la función, más preciso será el resultado.

Dentro de las fórmulas de Newton-Cotes se encuentra la regla del Rectángulo, la regla del Trapecio, la regla de Simpson 1/3 y Simpson 3/8.

5.2.1.1 Regla del Rectángulo

Este método es el más simple para hacer que una función interpoladora, una función constante, pase a través de un solo punto; mediante la **Ec. (5.5)**:

$$I = \int_a^b f(x) \sim (b-a) f(a) \quad . \quad \text{.....} \quad \textbf{(5.5)}$$

Este método consiste en dividir el área bajo la curva de la función en n partes iguales, mediante al dividir el intervalo de integración $[a, b]$, éstas serán representadas por pequeños rectángulos de base $dx = \frac{(b-a)}{n}$ y altura h.En donde h es calculada empleando la **Ec. (5.6).**

$$h = \left(\frac{a + dx}{2}\right) \quad \text{.....} \quad \textbf{(5.6)}$$

El área de estos pequeños rectángulos estará dada por:

$$A = dx * h \ , \quad \text{.....} \quad \textbf{(5.7)}$$

donde h corresponde al valor de la función calculada en el punto medio del área. El área total bajo la curva corresponderá a la suma del área de cada rectángulo:

$$I = \sum_{a}^{b} dx * h \ , \quad \text{.....} \quad \textbf{(5.8)}$$

este método resulta ser el más rápido en cuanto integración numérica, pero trae consigo un alto grado de error de truncamiento en cuanto a los resultados obtenidos.

A continuación, se presenta el algoritmo de implementación, así como el diagrama correspondiente es incluido en la **Fig. 5. 2:**

1. Elegir el número "n" en los que se dividirá 4l intervalo $[a, b]$.
2. Obtener el dx que será de la misma magnitud para cada rectángulo

$$dx = \frac{(b - a)}{n}$$

3. Calcular la altura para cada uno de los rectángulos mediante la **Ec.(5.6)**
4. Calcular el área para cada uno de los rectángulos empleando la **Ec. (5.7)**:
5. Obtener la integral mediante la suma de las áreas de cada rectángulo el área total bajo la curva, **Ec. (5.8)**:
6. La aproximación del área bajo la curva de la integral en el intervalo $[a, b]$ es: I

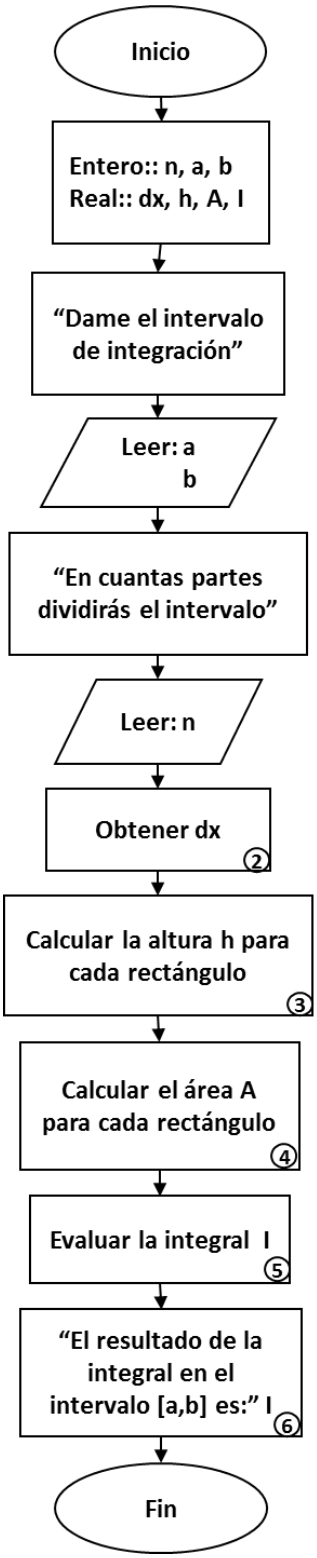


Fig. 5. 12
Regla del Rectángulo

5.2.1.2 Regla del Trapecio

Este método es empleado para obtener una aproximación de las integrales definidas, resultando ser equivalente a aproximar el área del trapecio bajo la línea recta, **Fig. 5. 3**, dentro de un intervalo $[a, b]$.

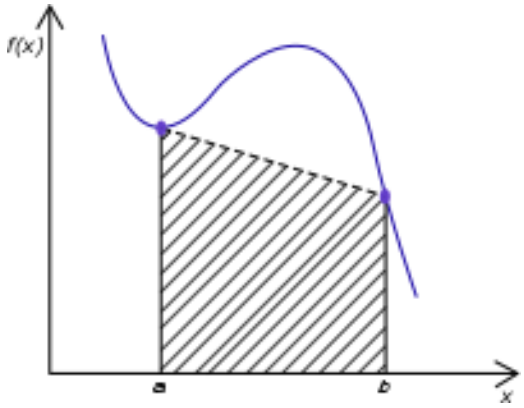


Fig. 5. 13
Representación gráfica de Regla del Trapecio

La aproximación de la integral es definida mediante la **Ec. (5.9)**:

$$I = \int_a^b \left[f(a) + \frac{f(b) - f(a)}{b - a}(x - a) \right] dx = (b - a)\frac{f(a) + f(b)}{2} \quad \text{.....} \quad \textbf{(5.9)}$$

Al obtener el resultado de la integral bajo la curva con esta ecuación, se arrastra un error de truncamiento, el cual puede ser disminuido con una aplicación múltiple de esta regla, dando origen a la **Ec. (5.10)**:

$$I = (b - a)\frac{f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)}{2n} \quad \text{.....} \quad \textbf{(5.10)}$$

A continuación, se presenta un algoritmo de implementación que facilite el uso de dicho método mediante su aplicación múltiple, también, se incluye su diagrama correspondiente en la **Fig. 5. 4.**

1. Dar el intervalo de integración $[a, b]$
2. Obtener el espaciamiento h dividiendo el intervalo de integración $[a, b]$ entre n

3. Obtener los subintervalos $(x_0, x_1, x_2, x_3, \dots x_{n-1},\ x_{n=b})$ para aplicar la regla del trapecio de manera múltiple, de tal forma que:
   $x_0 = a, x_1 = x_0 + h, x_2 = x_1 + h, \dots, x_i = x_{i-1} + h, x_n = b$
4. Definir la función $f(x)$
5. Valuar la función en cada uno de los subintervalos
   $$f(x_0), f(x_1), f(x_2), \dots, f(x_n)$$
6. Sustituir en **Ec. (5.10)**, para obtener la aproximación a la integral
7. La aproximación de la integral en el intervalo $[a, b]$ mediante la regla del Trapecio es: I

Parte esencial para la elaboración del código

```
function trapm(h,n,f)
sum=f0
do i=1,n-1
sum=sum+2*fi
end do
sum=sum+fn
trapm=h*sum/2
end trapm
```

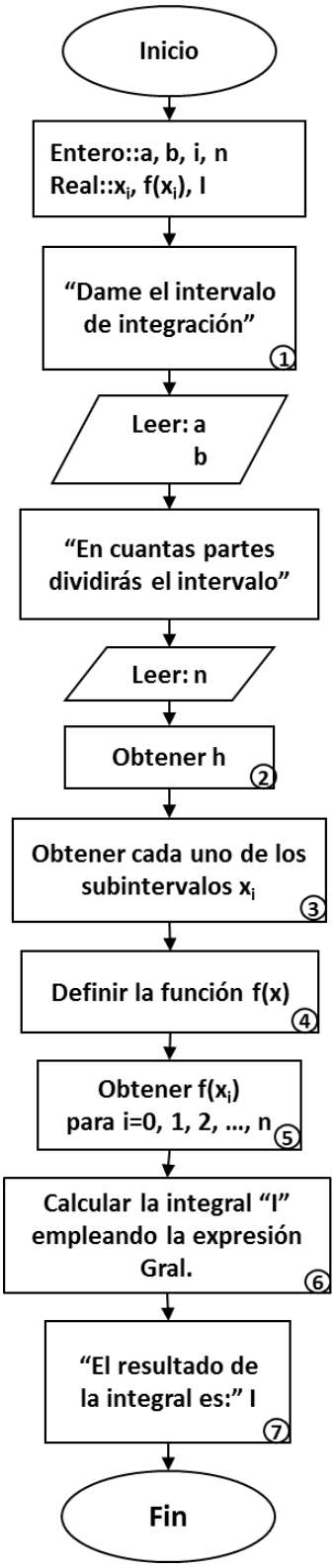


Fig. 5. 14
Método del Trapecio

5.2.1.3 Regla de Simpson 1/3

Este método es utilizado para evaluar integrales de una variable $\mathrm{I} = \int_{\mathrm{a}}^{\mathrm{b}} \mathrm{f(x)dx}$, así como integrales dobles $\mathrm{I} = \int_{\mathrm{a}}^{\mathrm{b}} \int_{\mathrm{u(x)}}^{\mathrm{v(x)}} \mathrm{f(x,y)dydx}$.

Una forma de obtener una aproximación más exacta de una integral consiste en utilizar polinomios de grado superior para unir los puntos, **Fig. 5. 5**, para este caso se toma el área bajo una parábola la cual une a tres puntos.

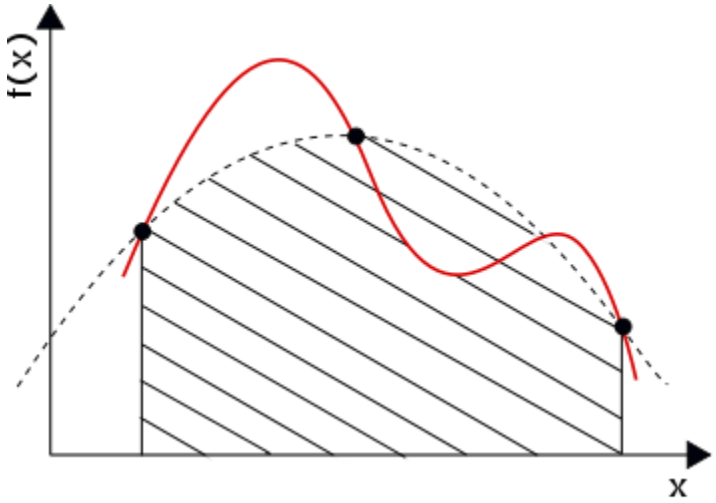


Fig. 5. 15
Representación gráfica del método de Simpson 1/3

Una de las expresiones resultantes al tomar la integral bajo el esquema de un polinomio de segundo grado corresponde precisamente a la regla de Simpson 1/3, la cual corresponde a la **Ec. (5.10)**:

$$I = \int_{x_0}^{x_2} f(x)dx = \frac{h}{3}[y_0 + 4y_1 + y_2] \ . \quad \text{.....} \qquad \textbf{(5.11)}$$

Al igual que la regla del rectángulo y del trapecio, la regla de Simpson puede mejorarse si se divide el intervalo de integración en varios segmentos de un mismo tamaño, **Fig. 5. 6**, para obtener resultados con mayor precisión.

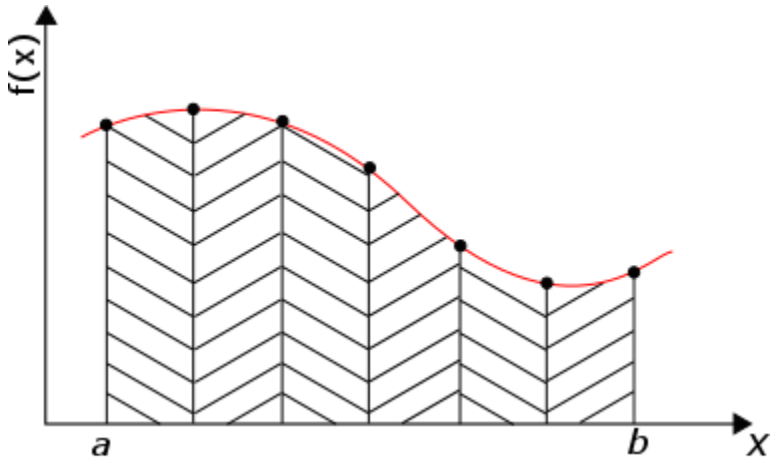


Fig. 5. 16
Representación gráfica del método de Simpson 1/3 múltiple

Algo importante que se debe recordar, es que el método puede ser empleado solo si el número de segmentos es par, de lo contrario no se podrá aplicar el método. La **Ec. (5.11)** es el resultado de esta mejora.

$$I = \int_{x_0}^{x_n} f(x)dx = \frac{h}{3}\left[y_0 + y_n + 4\sum_{indice\ impar}^{ordenadas\ con} + 2\sum_{indice\ par}^{ordenadas\ con}\right] \quad \text{.....} \quad \textbf{(5.12)}$$

A continuación, se presenta un algoritmo de implementación, así como el diagrama correspondiente se incluye en la **Fig. 5. 7**:

1. Definir la función $f(x)$
2. Solicitar el intervalo de integración $[a, b]$ y observar si la función a integrar numéricamente está en intervalos de $n = 2$ o $n = pares$, en caso de que no sea así definirlo.
3. Definir el intervalo de espaciamiento constante h de la siguiente manera:

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

dónde: $a = intervalo$ inferior de integración, $b = intervalo\ superior$ de integración, $n = intervalos\ (2\ o\ par)$.

4. Identificar la forma de la función a integrar (tabular o implícita), tabular ir a **paso 7**, implícita continuar.

5. Si la función está implícita definir los valores de x_i en el intervalo de integración de $[a, b]$, de la siguiente manera:
   $x_i = x_{i-1} + h$ para i=1, 2, …, n
   siendo $x_0 = a$ y $x_n = b$
6. Evaluar la función en cada uno de los puntos x_i para obtener y_i para $i = 0 = a, 1, 2, \dots, n = b$
7. Definir la formula a utilizar.
8. Si $n = 2$, estimar la integral mediante la **Ec. (5.11)**
9. Si n es un número par, estimar la integral mediante la **Ec.** (**5.12)**
10. La aproximación de la integral mediante Simpson 1/3 es: I

Parte esencial para la elaboración del código del Método de Simpson con aplicación múltiple, para segmentos pares e impares siempre que n>=1.

function simpint (a,b,n,f)
h=(b-a)/n
if (n.eq.1) then
sum=trap(h,fn-1,fn)
else (m.eq.n)
odd=n/2-iint(n/2)
if (odd.gt.0.and.n.gt.1) then
end simpint
	sum=sum+simp38(h,fn-3,fn-2,fn-1,fn)
	m=n-3
	end if
	end if
	simpint=sum
	end simpint

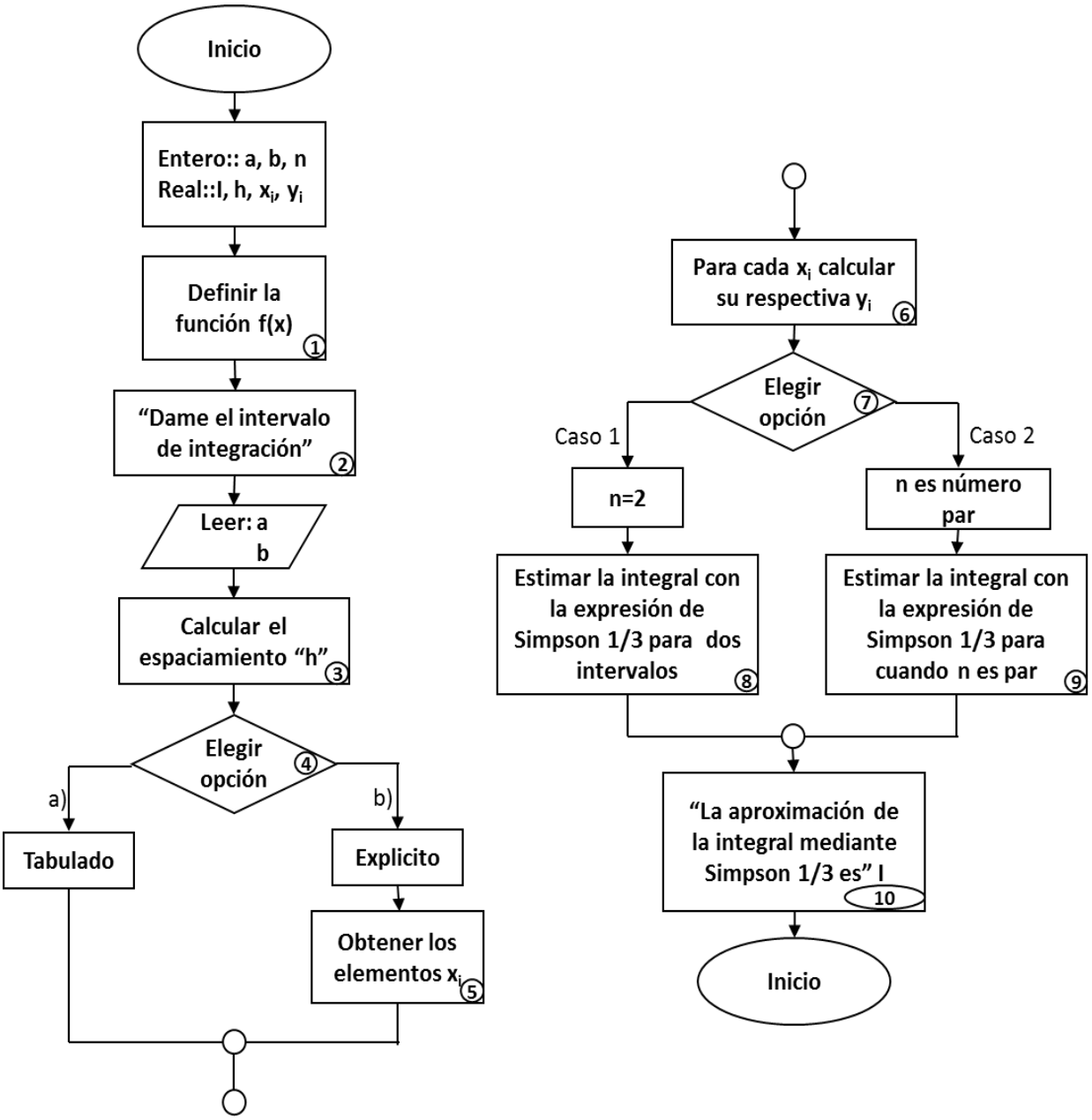


Fig. 5. 17
Método de Simpson 1/3

5.2.1.4 Regla de Simpson 3/8

Esta regla tiene la misma finalidad que la regla de Simpson 1/3, con la diferencia que el polinomio de aproximación no es de segundo grado sino de tercero, **Fig. 5. 8**, gráficamente lo que se presenta es tomar el área bajo una ecuación cúbica que une ahora a cuatro puntos.

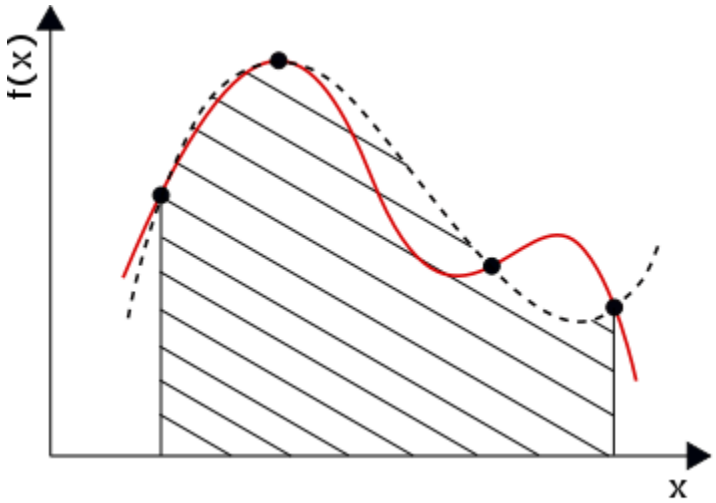


Fig. 5. 18
Representación gráfica de Método de Simpson 3/8

La regla se aplica cuando el número de intervalos n es igual a tres, en donde la **Ec. (5.13)** es empleada para obtener el valor de la integral.

$$I = \int_{x_0}^{x_2} f(x)dx = \frac{3}{8}h\,[y_0 + 3y_1 + y_2]\ . \quad \text{.....} \quad \textbf{(5.13)}$$

Dado que el denominador de la **Ec. (5.13)** es mayor que el de la **Ec. (5.12)** implica que Simpson 3/8 tiene un mayor grado de exactitud. La regla de 3/8 también puede tener aplicación múltiple que se presenta cuando el número de intervalos resulta ser múltiplo de tres, siendo la **Ec. (5.14)** la representativa para este caso.

$$I = \int_{x_0}^{x_n} f(x)dx = \frac{3}{8}h\left[y_0 + y_n + 2\sum_{multiplo\ de\ tres}^{ordenadas\ con\ indice} + 3\sum_{ordenadas}^{resto\ de}\right]. \quad \text{....} \quad \textbf{(5.14)}$$

Cuando no se sabe a qué regla recurrir, si a Simpson 1/3 o 3/8 debido a que el número de intervalos resulta no ser igual a dos, tres, un número par o un número

múltiplo de tres, lo recomendable es dividir el intervalo de integración en dos números o más partes de tal manera que utilicemos por separado las reglas y sumar el resultado de cada una para obtener el resultado final.

Se presenta continuación el algoritmo de implementación, así como el diagrama que corresponde al método se incluye en la **Fig. 5. 9**:

1. Definir la función $f(x)$ a integrar
2. Pedir el intervalo de integración $[a, b]$ y observar si la función tiene intervalos de tres o bien que sea múltiplo de tres, en caso de que no sea así este se deberá definir.
3. Al igual que en Simpson 1/3 definir el intervalo de espaciamiento constante h de la siguiente manera:

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

4. Identificar la forma de la función a integrar (tabular o implícita).
   4.1 Si la función está implícita definir los valores de x_i en el intervalo de integración de $[a, b]$, de la siguiente manera:

   $x_i = x_{i-1} + h$ para i=1, 2, …, n

   siendo $x_0 = a$ y $x_n = b$

5 Evaluar la función en cada uno de los puntos para obtener y_i para $i = a = 0, 1, 2, \ldots, n = b$

6 Definir la formula a utilizar:
   Si el número de intervalos n es igual a tres, estimar la integral empleando la **Ec. (5.13).**
   6.1 Si el número de intervalos n es múltiplo de tres, emplear la **Ec. (5.14)** para estimar el valor de la derivada

7 La aproximación de la integral mediante Simpson 3/8 es: I

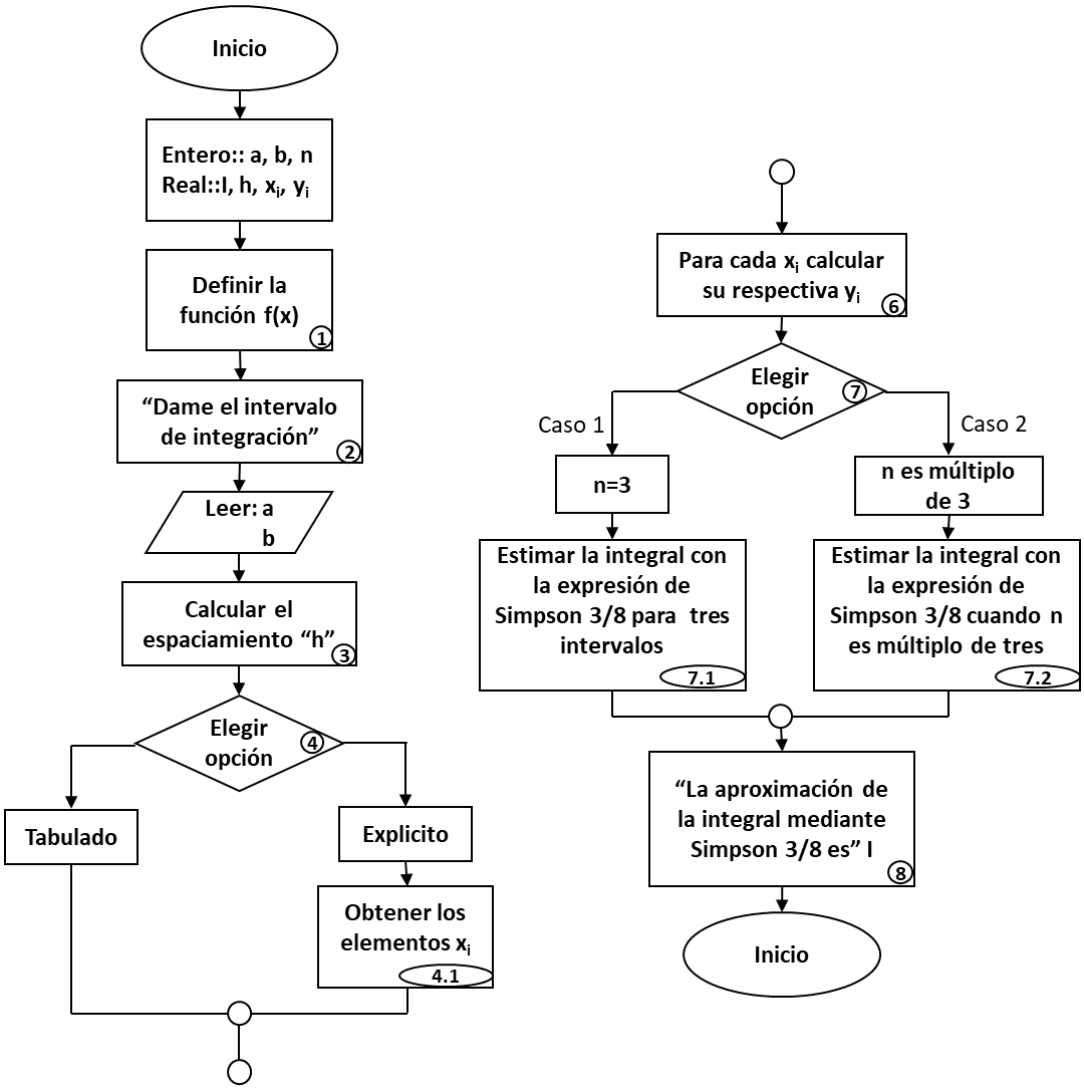


Fig. 5. 19
Método de Simpson 3/8

5.2.2 Cuadratura Gaussiana

Este método es uno de los más eficientes en cuanto a integración numérica, también es conocido como *Cuadratura de Gauss-Legendre*, ya que hace uso de los polinomios de *Legendre*.

Se utiliza cuando el intervalo de integración (a, b) es igual a $(-1,1)$ y e punto a buscar se encuentra en él, de lo contrario se lleva a cabo el proceso de sustitución de variables para acoplar dicha expresión al intervalo establecido.

La ecuación general que representa al método para n puntos en el intervalo (a, b) está dada por **Ec. (5.15)**:

$$\int_{-1}^{1} f(z)dx = w_1 f(z_1) + w_2 f(z) + w_3 f(z_3) + \cdots + w_n f(z_n) = \sum_{k=0}^{n} w_k\, f(z_k)\ . \quad \text{.....} \qquad \textbf{(5.15)}$$

Donde la **Ec. (5.16)** ayudara a obtener los w_k (factor de ponderación):

$$w_k = \frac{1}{P'_{n+1}(z_k)} \int_{-1}^{1} \frac{P_{n+1}(z)}{(z - z_k)} dz\ , \qquad \text{.....} \qquad \textbf{(5.16)}$$

en la **Tabla 5. 1**, presenta el valor correspondiente de w_ky z de acuerdo al número de puntos establecidos y $P_n(z)$ corresponde a los polinomios de *Legendre* expresados por:

$$\begin{aligned} P_{i=0}(z) &= 1 \\ P_1(z) &= z \\ P_2(z) &= \frac{1}{2}(3z^2 - 1) \\ P_3(z) &= \frac{1}{2}(5z^3 - 3z) \\ P_4(z) &= \frac{1}{8}(35z^4 - 30z^2 + 3) \\ &\vdots \\ P_n(z) &= \frac{1}{2^n n!}\frac{d^n}{dz^n}(z^2 - 1)^n\ , \end{aligned} \qquad \text{.....} \qquad \textbf{(5.17)}$$

para $i = 1, 2, \ldots, n$,

siendo las z_k son las raíces que dan solución a los polinomios de *Legendre*.

Tabla 5. 2
Coeficientes w_k y z empleados en la Cuadratura Gaussiana

Puntos	Coeficientes w_k	Abscisas z
2	$w_1 = w_2 = 1$	$-z_1 = z_2 = 0.5773502692$
3	$w_2 = 0.88888\ldots.$ $w_1 = w_3 = 0.55555\ldots$	$-z_1 = z_3 = 0.7745966692$ $z_2 = 0.0$
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	$w_1 = w_5 = 0.2369268851$ $w_2 = w_4 = 0.4786286705$ $w_3 = 0.56888\ldots$	$-z_1 = z_5 = 0.9061798459$ $-z_2 = z_4 = 0.5384693101$ $z_3 = 0.0$
6	$w_1 = w_6 = 0.1713244924$ $w_2 = w_5 = 0.3607615730$ $w_3 = w_4 = 0.4679139346$	$-z_1 = z_6 = 0.9324695142$ $-z_2 = z_5 = 0.6612093865$ $-z_3 = z_4 = 0.2386191861$

Cuando se presenta el caso en que el intervalo de integración es diferente de $(-1,1)$, el método puede aplicarse si se recurre a un cambio de variable para ajustar dicho intervalo mediante la **Ec. (5.18)**.

$$z = \frac{2x - (a+b)}{b-a}\ , \qquad \ldots\ldots\ldots\ldots \quad \textbf{(5.18)}$$

cambiando la variable y el diferencial:

$$x = a \rightarrow z = \frac{2(a) - (a+b)}{b-a} = -1$$

$$x = b \rightarrow z = \frac{2(b) - (a+b)}{b-a} = 1$$

resultando así la **Ec. (5.19)** para la integral de la siguiente manera:

$$\int_a^b f(x)dx = \frac{b-a}{2} \int_{-1}^{1} f\left(\frac{b-a}{2}z + \frac{a+b}{2}\right) dz\ , \qquad \ldots\ldots\ldots\ldots \quad \textbf{(5.19)}$$

la cual puede ser expresada de forma general como:

$$\int_a^b f(x)dx = \frac{b-a}{2}\sum_{i=1}^{n} w_i f\left(\frac{b-a}{2}z_i + \frac{a+b}{2}\right), \quad \text{.....} \quad \textbf{(5.20)}$$

donde los coeficientes w_i y z_i son estimados de la misma manera como cuando se tiene el intervalo de $(-1,1)$, por lo que la **Ec. (5.21)** será la empleada para el cálculo de la integral.

$$I = \int_{-1}^{1} f(x)dx = w_1 f(x_1) + w_2 f(x_2) + w_3 f(x_3) + \cdots + w_n f(x_n) = \sum_{k=0}^{n} w_k\, f(x_k)\ , \ \ .... \quad \textbf{(5.21)}$$

Será presentado a continuación el algoritmo de implementación, así como su diagrama correspondiente en la **Fig. 5. 10**:

1. Definir la función $f(x)$ a integrar en el intervalo de integración $(1,-1)$
2. Identificar el grado del polinomio presente en la integral
3. Seleccionar el polinomio de *Legendre* a utilizar y obtenerlas las respectivas raíces desde $i = 1, 2, \dots, n$; haciendo uso de la **Ec. (5.17)**.
4. Obtener el factor de ponderación w_k empleando la **Ec. (5.16)**.
5. Estimar el valor de la integral mediante la **Ec. (5.21)** representativa del método de *Cuadratura Gaussiana*.
6. La aproximación de la integral mediante Cuadratura Gaussiana es: I.

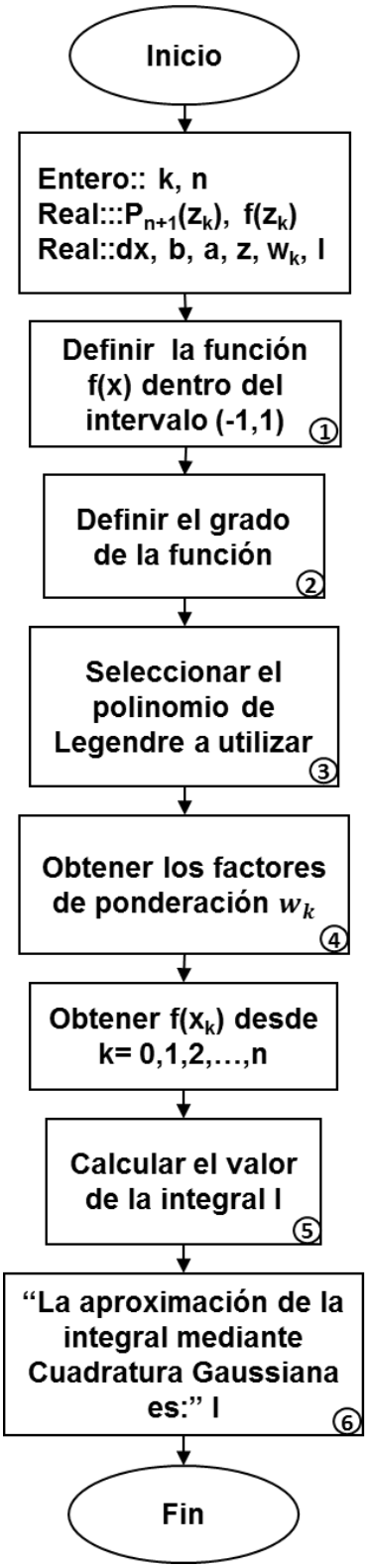


Fig. 5. 20
Método de Cuadratura Gaussiana

Capítulo 6

Ecuaciones diferenciales ordinarias

En este capítulo se abordará lo referente a solución a ecuaciones diferenciales ordinarias (EDO) o bien, a los sistemas que pueden formar estas ecuaciones.

Para cumplir con el objetivo se trabajará primeramente con el método de Euler y Euler mejorado, para así llegar a los métodos de Runge-Kutta de segundo, tercer y cuarto grado, siendo este último el más recurrido en la literatura para dar solución a este tipo de sistemas de EDO.

6.1 Que son las ecuaciones diferenciales ordinarias

En una ecuación diferencial interviene una función incógnita y una o varias de sus derivadas.

Una ecuación diferencial ordinaria de primer orden (EDO) corresponde a la forma:

$$y' = F(x, y) \ , \qquad \text{.....} \qquad \textbf{(6.1)}$$

donde la ecuación F depende de las variables x e y. Estas ecuaciones diferenciales son las más sencillas; ya que la derivada mayor corresponde a la primera derivada.

En este capítulo se explicarán algunos métodos numéricos para resolver problemas de valor inicial en EDO y los sistemas que éstas pueden llegar a formar.

6.2 Método de Euler

La idea del método de Euler es muy sencilla y está basada en el significado geométrico de la derivada de una función en un punto dado.

Este método parte de la suposición de tener la curva solución a la ecuación diferencial y trazar la recta tangente a la curva en el punto dado con la condición inicial.

Debido a que la recta tangente se aproxima a la curva en valores cercanos al punto de tangencia, **Fig. 6. 1,** puede ser tomado el valor de la recta tangente en el punto x_1 como una aproximación al valor deseado $y(x_1)$.

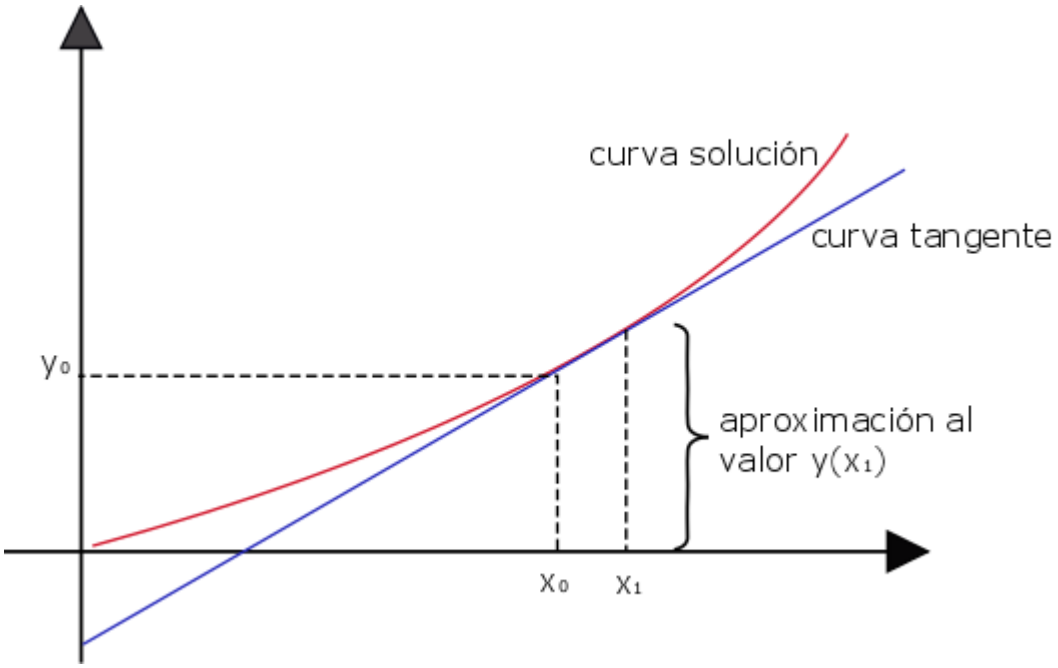


Fig. 6. 2
Representación gráfica del Método de Euler

Resultando más fácil obtener la ecuación de la recta tangente a la curva solución de la ecuación diferencial en el punto (x_0, y_0).

De aquí surge la expresión de aproximación **Ec. (6.2)**, que representa a la recta tangente en el punto $x_1 = x_0 + h$; donde:

$$y_1 \approx y_0 + h \cdot f(x_0, y_0) \ , \qquad \text{.....} \qquad \textbf{(6.2)}$$

esta aproximación es suficientemente buena si se toman valores de h suficientemente pequeños, de lo contrario arrojará un error.

De esta expresión resulta la Formula de Euler, **Ec. (6.3)**, usada para aproximar el valor de $y(x_1)$ aplicándola sucesivamente desde x_0 hasta x_1.en pasos de longitud h.

$$y_{n+1} = y_n + h \cdot f(x_n, y_n) \ , \quad \text{.....} \quad \textbf{(6.3)}$$

El método de Euler Mejorado, **Ec. (6.4)**, se basa en la misma idea del método anterior, solo que se hace un refinamiento en cuanto a la aproximación, ya que toma el promedio entre ciertas pendientes.

$$y_{n+1} = y_n + h \cdot \left[\frac{f(x_n, y_n) + f(x_{n+1}, y*_{n+1})}{2}\right] , \quad \text{.....} \quad \textbf{(6.4)}$$

donde:

$$y*_{n+1} = y_n + h \cdot f(x_n, y_n) \ . \quad \text{.....} \quad \textbf{(6.5)}$$

A continuación, se presenta el algoritmo de implementación, así como el diagrama correspondiente del método en la **Fig. 6.2**:

1. Dada la ecuación diferencial $\frac{dy}{dx}$, sus condiciones iniciales (x_n, y_n)
2. Establecer un espaciamiento h (se recomienda un valor pequeño para una mejor aproximación) para estimar el valor de y_{n+1}
3. Obtener la función valuada para las condiciones (x_n, y_n)
4. obtener y_{n+1} mediante la **Ec. (6.3)**:
5. Calcular el valor verdadero en el punto mediante la ecuación de solución $y = f(x)$
6. Calcular el error relativo porcentual entre el valor calculado en la iteración y el valor verdadero de la ecuación mediante:

$$\varepsilon = \frac{valor\ calculado - valor\ verdadero}{valor\ calculado}$$

7. Si $\varepsilon < tolerancia$, el resultado de la EDO es: y_{n+1}.
8. De no cumplirse la condición anterior obtener una nueva iteración (**paso 3**) haciendo $y_{n+1} = y_n$ para conocer un nuevo valor de y_{n+1}.

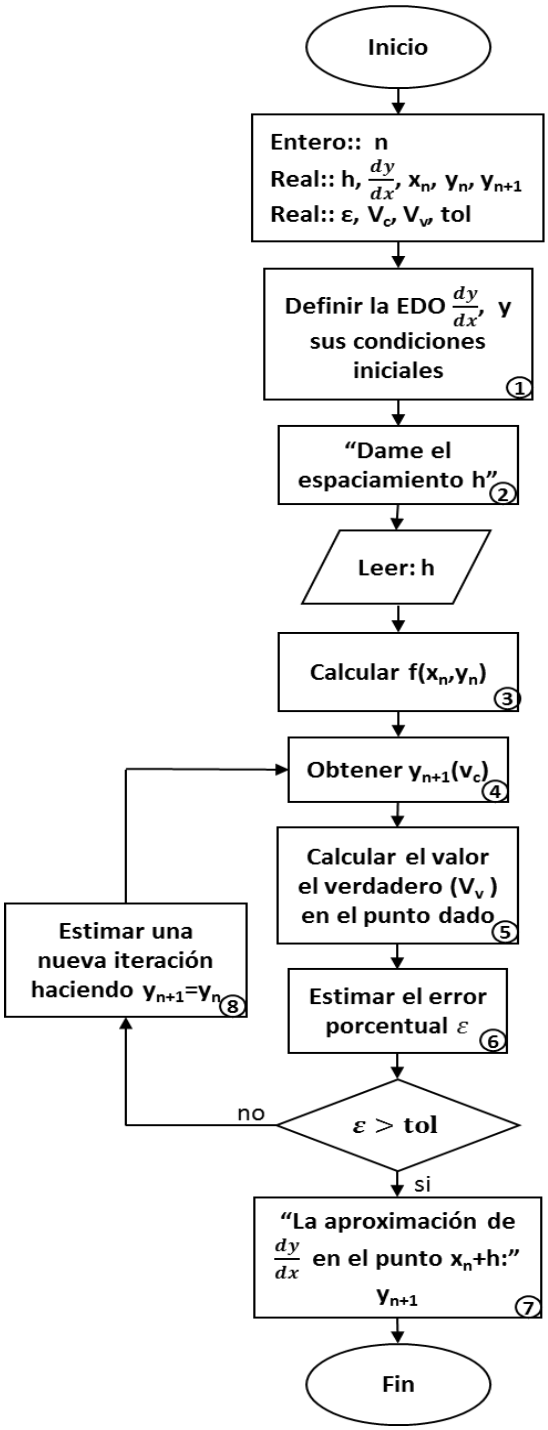


Fig. 6.2
Método de Euler

6.3 Métodos de Runge-Kutta

Los métodos de Runge-Kutta (RK) de segundo, tercer y cuarto orden dan solución a EDO, logrando una mejor exactitud al procedimiento de la serie de Taylor, pero sin recurrir al cálculo de derivadas de orden superior.

Aunque existen muchas versiones de estos métodos, todos presentan la forma generalizada de la **Ec. (6.6)** de forma:

$$y_{i+1} = y_i + \emptyset(x_i, y_i, h)h \ , \quad \text{.....} \quad \textbf{(6.6)}$$

donde $\emptyset(x_i, y_i, h = espaciamiento)$ es conocida como función incremento, y es interpretada como una pendiente representativa en el intervalo, cuya expresión general es representada mediante la **Ec. (6.7)**:

$$\emptyset = a_1k_1 + a_2k_2 + \cdots + a_nk_n \ , \quad \text{.....} \quad \textbf{(6.7)}$$

donde las $a_1, a_2, \ldots, a_n$ son constantes y las $k_1, k_2, \ldots, k_1$ son obtenidas mediante la **Ec. (6.8)**[8]**:**

$$\begin{gathered} k_1 = f(x_i, y_i) \\ k_2 = f(x_i + p_1h, y_i + q_{11}k_1h) \\ k_3 = f(x_i + p_2h, y_i + q_{21}k_1h + q_{22}k_2h) \\ \vdots \\ k_n = f\left(x_i + p_{n-1}h, y_i + q_{n-1,1}k_1h + q_{n-1,2}k_2h + \cdots + q_{n-1,n-1}k_{n-1}h\right) \ . \end{gathered} \quad \text{.....} \quad \textbf{(6.8)}$$

Para obtener cada uno de los modelos correspondientes al método de Runge-Kutta, solo basta con sustituir el valor de n. Así cuando sea $n = 1$ la expresión que se obtiene corresponde precisamente al Método de Euler.

Método de Runge-Kutta de segundo orden

La ecuación general representativa del *método de RK* cuando $n = 2$ queda expresada en la **Ec. (6.9)**:

[8] p_i y q_i son constantes y las $k's$ son relaciones de recurrencia.

$$y_{i+1} = y_i + h(a_1 k_1 + a_2 k_2) \ , \quad \text{.....} \quad \textbf{(6.9)}$$

donde

$$k_1 = f(x_i, y_i)$$
$$k_2 = f(x_i + p_1 h, y_i + q_{11} k_1 h) \ .$$

Existen tres variantes a este método, donde la diferencia entre cada uno consiste en los valores asignados a las variables $a_1, p_{1,} q_{11}$ y principalmente al asignado a la constante a_2, las expresiones para conocer cada una de estas variables son las siguientes:

$$a_1 = 1 - a_2$$
$$a_2 \cdot p_1 = \frac{1}{2} \ ,$$
$$a_2 \cdot q_{11} = \frac{1}{2}$$

donde en el Método de Heun asigna a la variable a_2 el valor de $\frac{1}{2}$ y a partir de esta encuentra los valores correspondientes a las demás variables; otra versión es el Método del punto medio en la cual se maneja a $a_2 = 1$ y la última variante es el Método de Ralston en la cual $a_2 = \frac{2}{3}$. El más utilizado de los tres en precisamente el Método de Ralston ya que maneja un menor error de truncamiento; por lo que sólo se presentaran las expresiones correspondientes a este método.

Cuando $a_2 = \frac{2}{3}$, $a_1 = \frac{1}{3}$ y $p_1 = q_{11} = \frac{3}{4}$ por lo que la Ec. (6.10) será la que represente al Método de Runge-Kutta de segundo:

$$y_{i+1} = y_i + h\left(\frac{1}{3} k_1 + \frac{2}{3} k_2\right) \ , \quad \text{.....} \quad \textbf{(6.10)}$$

En donde:

$$k_1 = f(x_i, y_i) \ ,$$
$$k_2 = f\left(x_i + \frac{3}{4} h, y_i + \frac{3}{4} k_1 h\right) \ .$$

Método de Runge-Kutta de tercer orden

Este método es planteado a partir de la recurrencia al Método de Simpson $\frac{1}{3}$, al igual que los anteriores, sirve para resolver EDO arrojando resultados exactos cuando la ED sea una cúbica y su solución sea de cuarto grado mediante la **Ec.(6.10)**:

$$y_{i+1} = y_i + \frac{h}{6}(k_1 + 4k_2 + k_3) \ , \quad \text{..... (6.11)}$$

donde

$$k_1 = f(x_i, y_i) \ ,$$
$$k_2 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1 h\right) \ ,$$
$$k_3 = f(x_i + h, y_i + k_1 h + 2k_2 h) \ .$$

Método de Runge-Kutta de cuarto orden.

El más popular de los métodos de RK es el de cuarto orden. Al igual que en los procedimientos de segundo orden existen un número infinito de versiones, por lo cual solo se presentará la forma más usada a cuál llamaremos método clásico de RK de cuarto orden. La **Ec. (6.12)** representa al método:

$$y_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \ , \quad \text{..... (6.12)}$$

donde

$$k_1 = f(x_i, y_i) \ ,$$
$$k_2 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1 h\right) \ ,$$
$$k_3 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_2 h\right) \ ,$$
$$k_4 = f(x_i + h, y_i + k_3 h) \ .$$

El método de RK de cuarto orden presenta similitud al planteamiento de Heun, en cuanto el cálculo de múltiples estimaciones de la pendiente para tener una mejor pendiente promedio en el intervalo.

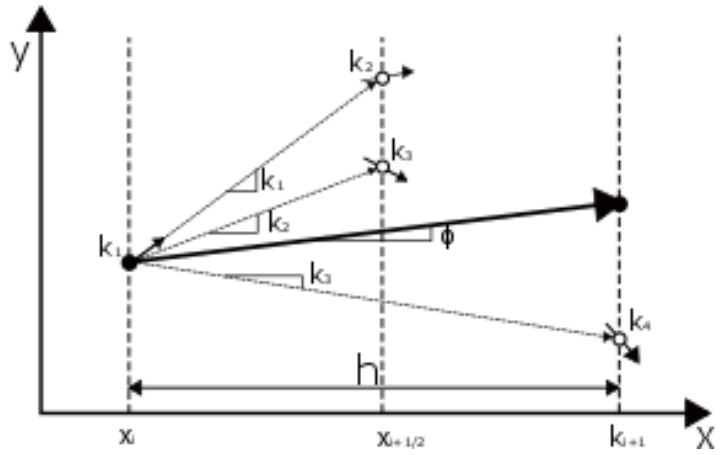


Fig. 6.3
Representación gráfica de las pendientes
Estimadas en el método de Runge-Kutta de cuarto orden

A continuación, se presenta el algoritmo para la implementación del método de RK de cuarto orden, así como el respectivo diagrama se esquematiza en la **Fig. 6.4**:

1. Definir la EDO de tercer orden, así como sus respectivas condiciones iniciales
2. Establecer el valor del espaciamiento h y decidir el número de iteraciones n a realizar.
3. Calcular el valor correspondiente de k_1, para lo cual basta con sustituir los valores iniciales x_i, y_i en la EDO.

$$k_1 = f(x_i, y_i)$$

4. Obtener el valor de k_2 empleando la siguiente expresión:

$$k_2 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1 h\right)$$

5. Obtener el valor correspondiente a k_3 con la expresión.

$$k_3 = f(x_i + h, y_i + k_1 h + 2k_2 h)$$

6. obtener el valor correspondiente de k_4 con la expresión.

$$k_4 = f(x_i + h, y_i + k_3 h)$$

7. Sustituir k_1, k_2, k_3, k_4 en la **Ec. (6.12)** para obtener el valor de y_{i+1} requerido.
8. Verificar si la iteración i<n
9. Si no se cumple la condición se deberá de repetir el proceso a partir del **paso 3** para volver a calcular $k_1, k_2, k_3\ y\ k_4$, pero ahora con la nueva x y la nueva y calculadas anteriormente. Para obtener el siguiente valor de y, en caso contrario ir a **paso 10**.
10. El resultado a la EDO es: y_{i+1}

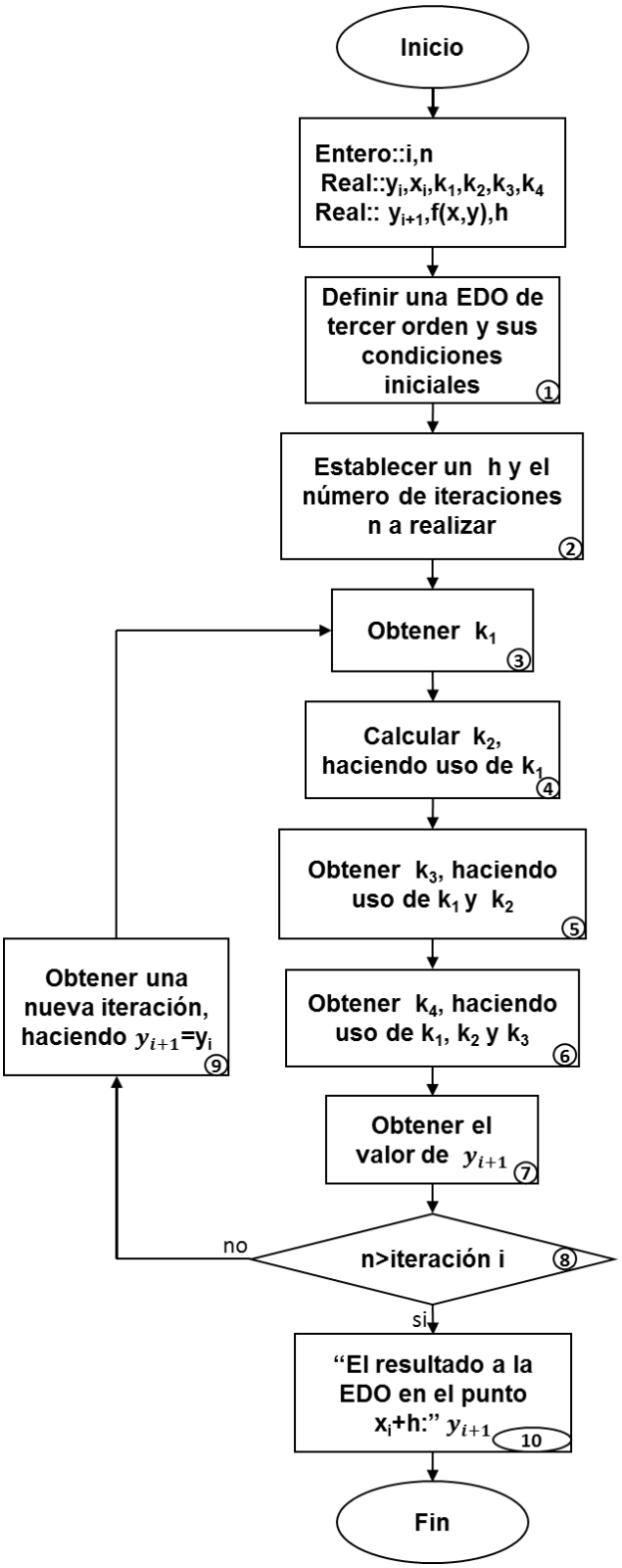


Fig. 6.4
Método de Runge Kutta de cuarto orden

Capítulo 7

Ecuaciones diferenciales parciales

En este capítulo se trabajará lo referente a ecuaciones parabólicas como lo son la ecuación de conducción de calor, método Explícito, Implícito y Crak-Nicolson dentro de las cuales también se trabaja con ecuaciones diferenciales parciales.

Este capítulo resulta ser de mayor complejidad, ya que, para dar solución al sistema de ecuaciones, se recurre a otros métodos vistos anteriormente, los cuales resultan ser indispensables para su manejo, entre los métodos a emplear se encuentran: diferencias finitas y el método de Thomas para dar solución a sistemas tridiagonales.

Los temas abordados en este capítulo son empleados en muchos de los procesos en ingeniería, y en Ingeniería Petrolera no puede ser acepción, puesto que son empleados en el área de yacimientos principalmente.

7.1 Que son las ecuaciones diferenciales parciales

Las ecuaciones diferenciales parciales (EDP) son aquellas ecuaciones diferenciales que dependen de dos o más variables independientes, por ejemplo:

$$\frac{\partial^2 u}{\partial x^2} + 2xy\frac{\partial^2 u}{\partial y^2} + u = 1 \ , \quad \text{.....} \quad \textbf{(7.1)}$$

$$\frac{\partial^3 u}{\partial x^2 \partial y} + x\frac{\partial^2 u}{\partial y^2} + 8u = 5y \ , \quad \text{.....} \quad \textbf{(7.1)}$$

$$\frac{\partial^2 u}{\partial x^2} + xu\frac{\partial u}{\partial y} = x \ . \quad \text{.....} \quad \textbf{(7.3)}$$

En donde el orden de una EDP se puede saber al identificar la derivada parcial de mayor orden presente en la ecuación; para esta sección se abordarán solamente EDP de segundo orden.

7.2 Ecuaciones Parabólicas

Son aquellas ecuaciones diferenciales parciales (EDP) que varían en el tiempo, por lo que en este capítulo se abordara la solución a la ecuación de Conducción de Calor mediante la aplicación de los métodos Explicito, Implícito y Crank-Nicolson, abordando también, las respectivas condiciones de frontera.

7.2.1 Ecuación de conducción de calor

Esta ecuación puede ser utilizada para conocer la distribución de calor que es almacenado a lo largo de una barra delgada para un tiempo Δt y parte del siguiente principio: $entradas - salidas = acumulación$.

Siendo así la **Ec. (7.4)** la representativa de conducción de calor:

$$k\frac{\partial^2 T}{\partial x^2} = \frac{\partial T}{\partial t}\ , \qquad \text{.....} \qquad \textbf{(7.4)}$$

la cual resulta ser una ecuación parabólica, por lo que deben ser considerados los cambios tanto en tiempo como en espacio, dos métodos que dan solución a esta ecuación son el método Explícito y el método Implícito, además del método de Cranck-Nicolson que disminuye notablemente el error para su solución.

7.2.2 Método explícito

Siendo que la ecuación de conducción de calor requiere de aproximaciones de la segunda derivada en el espacio y de la primera en el tiempo; la segunda derivada es representada mediante una diferencia dividida finita centrada **Ec. (7.5)**:

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i+1}^l - 2T_i^l + T_{i-1}^l}{\Delta x^2}\ , \qquad \text{.....} \qquad \textbf{(7.5)}$$

donde el cambio en la denotación de los superíndices se utiliza para denotar el tiempo.

Por otra parte, una diferencia dividida finita hacia adelante sirve para aproximar a la derivada con respecto al tiempo **Ec. (7.6)**:

$$\frac{\partial T}{\partial t} = \frac{T_{i+1}^{l} - T_{i}^{l}}{\Delta t} \ . \quad \text{.....} \quad \textbf{(7.6)}$$

Al sustituir las ecuaciones (**7.5**) y (**7.6**) en (**7.4**) se obtiene:

$$k\frac{T_{i+1}^{l} - 2T_{i}^{l} + T_{i-1}^{l}}{\Delta x^2} = \frac{T_{i+1}^{l} - T_{i}^{l}}{\Delta t} \ , \quad \text{.....} \quad \textbf{(7.7)}$$

resultando así:

$$T_{i+1}^{l} = T_{i}^{l} + \lambda\left(T_{i+1}^{l} - 2T_{i}^{l} + T_{i-1}^{l}\right) \ , \quad \text{.....} \quad \textbf{(7.8)}$$

donde $\lambda = k\Delta t/(\Delta x)^2$.

La **Ec. (7.8)** será usada para calcular la distribución de la temperatura en cada uno de los nodos para un tiempo posterior, tomando como referencia a los valores presentes del nodo y de sus respectivos vecinos.

El esquema explicito es convergente[9] y estable[10] solo si el valor de $\lambda \leq \frac{1}{2}$ o si $\Delta t \leq \frac{1}{2}\frac{\Delta x^2}{k}$.

A continuación, se presenta el algoritmo de implementación para el método explicito, el cual se explicará basado a la ecuación de conducción de calor, pero es importante mencionar que no solo se aplica para este caso, ya que hay muchos problemas a los cuales da solución; también se incluye el diagrama correspondiente en la **Fig. 7.1:**

1. Definir la longitud de la barra metálica, el tiempo total y las respectivas Δx y Δt
2. Solicitar las condiciones de frontera (temperatura al inicio y al final de la barra)
3. Dar el valor de la constante k correspondiente al material de la barra
4. Calcular el valor de λ mediante la expresión:

[9] *Convergencia*: conforme Δx y Δt tiendan a cero, los resultados de la técnica de diferencias finitas se aproximan a la solución verdadera.
[10] *Estabilidad*: los errores en cualquier parte del cálculo no se amplifican, si no que se acentúan conforme avanza el cálculo.

$$\lambda = k\Delta t/(\Delta x)^2$$

5. Calcular la distribución de la temperatura para el primer tiempo t, a lo largo de toda la distancia x evaluando uno a uno los nodos correspondientes en la **Ec. (7.8)** (l representa el tiempo e i representa el espacio x).
   para $l = 1$ e $i = 1$ hasta $i = m$
6. Aplicar nuevamente la **Ec. (7.8)** para obtener la distribución de temperatura en el siguiente lapso de tiempo, a lo largo de toda la barra.
   Para $l = 2$ hasta $l = t$ asi como para $i = 1$ hasta $i = n$
7. Finalmente se conoce la distribución de la temperatura a lo largo de la barra conforme avanza el tiempo (los resultados pueden graficarse para ver su respetiva variación en el tiempo y distancia).

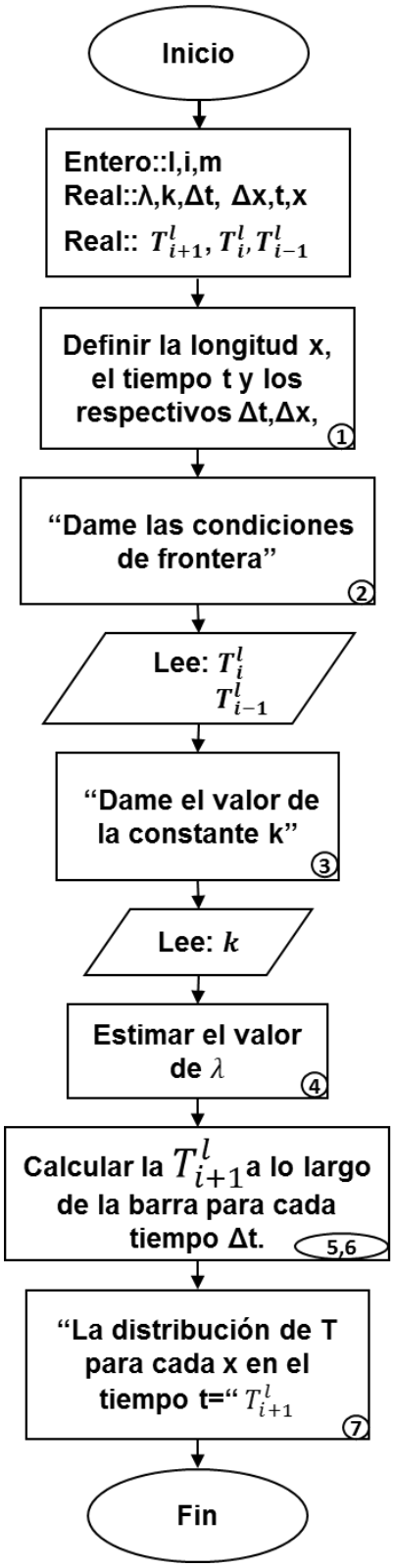


Fig. 7. 1
Método Explícito

7.2.3 Método implícito

Con este método, también se puede dar solución a la ecuación de conducción de calor. ya que la forma explícita por diferencias finitas presenta problemas relacionados con la estabilidad, las cuales se ven mejoradas con éste.

En el método implícito la derivada espacial es aproximada para un nivel de tiempo posterior $l+1$, ya que la segunda derivada es aproximada mediante la **Ec. (7.9)**:

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i+1}^{l+1} - 2T_i^{l+1} + T_{i-1}^{l+1}}{(\Delta x)^2} \quad \text{.} \qquad \text{.....} \quad \textbf{(7.9)}$$

Dado que en esta ecuación se presenta más de una incógnita, esta no puede resolverse como en el método explicito, por lo que es necesario crear un sistema de ecuaciones algebraicas lineales en cada punto del tiempo con el mismo número de incógnitas al considerar las condiciones de frontera el cual debe resolverse simultáneamente. La ecuación representativa de este método surge cuando se sustituye la **Ec**. **(7.9)** y **Ec. (7.7)** en **Ec. (7.4)** resultando la **Ec. (7.10)**:

$$-\lambda T_{i-1}^{l+1} + (1+2\lambda)T_i^{l+1} - \lambda T_{i+1}^{l+1} = T_i^l \ , \qquad \text{.....} \quad \textbf{(7.10)}$$

donde $\lambda = k\Delta t/(\Delta x)^2$.

Esta ecuación, puede ser utilizada para todos los nodos exceptuando el primero y último, utilizando las condiciones de frontera en los nodos exteriores, la **Ec. (7.11)** representa al primer nodo de la barra (cuando $i = 1$) es:

$$(1+2\lambda)T_1^{l+1} - \lambda T_2^{l+1} = T_1^l + f_0(t^{l+1}) \ , \quad \text{.....} \quad \textbf{(7.11)}$$

donde $f_0(t^{l+1})$ es una función que describe cómo cambia con el tiempo la temperatura de la frontera.

Y para el último nodo interior (cuando $i = m$) la **Ec. (7.12)** es utilizada:

$$-\lambda T_{m-1}^{l+1} + (1+2\lambda)T_1^{l+1} = T_m^l + \lambda f_{m+1}(t^{l+1}) \ , \quad \text{.....} \quad \textbf{(7.12)}$$

donde $\lambda f_{m+1}(t^{l+1})$ describe los cabios específicos de temperatura en el extremo derecho de la barra.

Con éstas dos últimas expresiones surge el sistema de m ecuaciones lineales con m incógnitas, este método tiene la ventaja de que el sistema que surge es tridiagonal, por lo que puede resolverse empleando el método descrito en capítulos anteriores. El método implícito descrito es estable y convergente.

A continuación, se presenta el algoritmo para dar solución a la ecuación de conducción de calor empleando el método implícito, ya que resulta ser más estable y a su vez convergente; también se incluye el diagrama correspondiente en la **Fig. 7. 2**.

1. Definir la longitud de la barra metálica, el tiempo total y las respectivas Δx y Δt
2. Definir las condiciones de frontera (temperatura al inicio y al final de la barra)
3. Dar el valor de la constante k correspondiente al material de la barra
4. Obtener el valor de λ mediante la expresión:

$$\lambda = k\Delta t/(\Delta x)^2$$

5. Obtener la primera ecuación para el nodo (frontera 1) uno empleando la **Ec. (7.11)**.
6. Obtener la ecuación para el último nodo con la **Ec. (7.12)**.
7. Obtener las ecuaciones para los nodos interiores al evaluar en **Ec. (7.10)**.

   para $l = t = 1$ y para $i = 1$ hasta m

8. Representar el sistema de ecuaciones de forma matricial
9. Resolver el sistema empleando el método más adecuado para sistemas tridiagonales (se recomienda emplear el método de Thomas visto en el **Capítulo 2.1.3**).
10. Repetir a partir del **paso 7.** para obtener los respectivos sistemas de ecuaciones para cada lapso de tiempo.
11. Finalmente se conoce la distribución de la temperatura a lo largo de la barra conforme avanza el tiempo con un método más confiable (los resultados pueden graficarse para ver su respetiva variación).

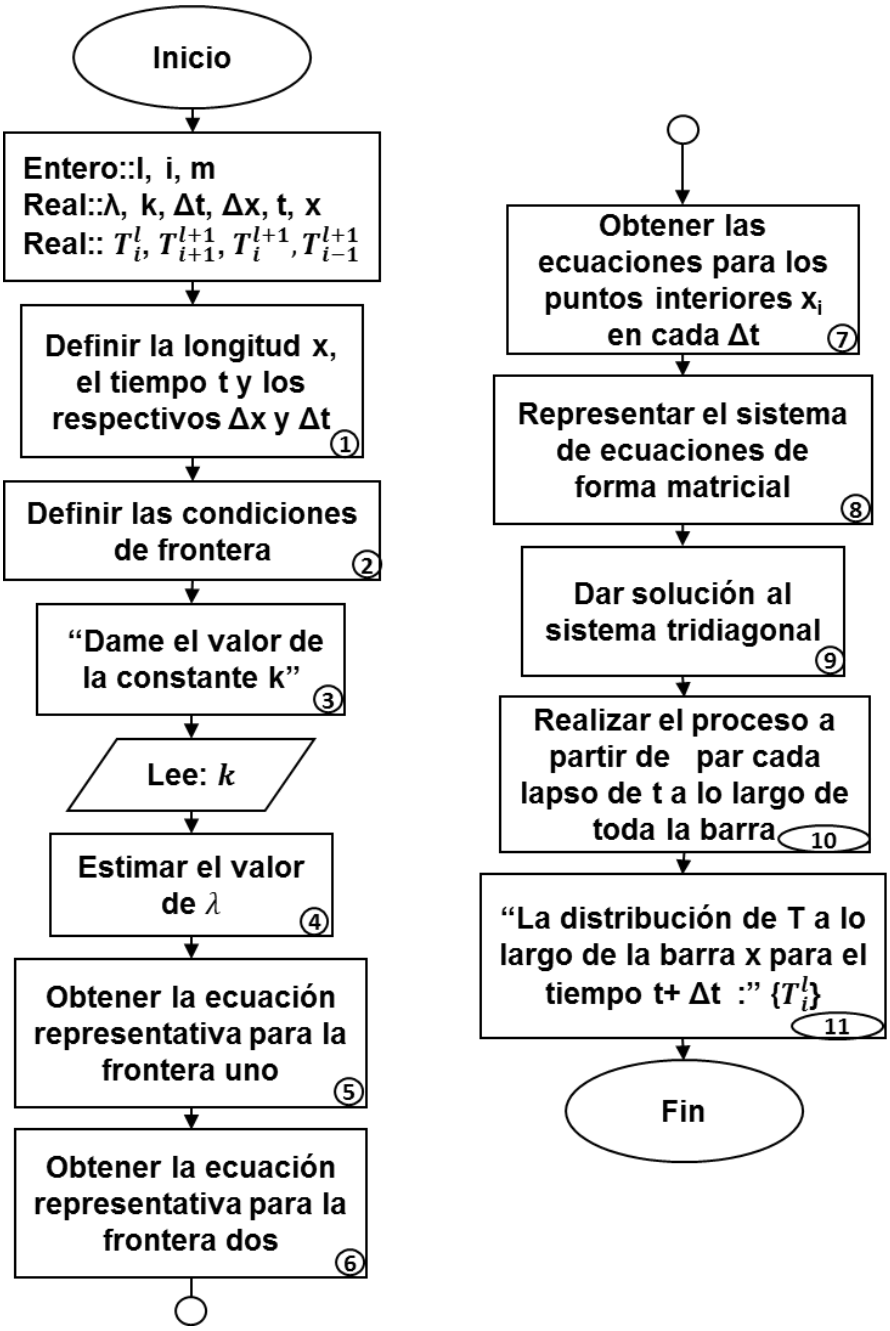


Fig. 7. 2
Método Implícito

7.2.4 Método de Crank-Nicolson

Este método resulta de ser una combinación de los métodos implícito y explícito de Euler. Además, ofrece un esquema implícito alternativo con mayor

exactitud que los métodos anteriores, para alcanzar tal exactitud se realizan aproximaciones por diferencias en el punto medio de incremento del tiempo; aproximando la primera derivada temporal para $t^{l+1/2}$ y la segunda derivada en el espacio puede determinarse en el punto medio al promediar las aproximaciones por difeencias al principio y al final del incremento en el tiempo, para llegar a la **Ec. (7.13)**:

$$-\lambda T_{i+1}^{l+1} + 2(1+\lambda)T_i^{l+1} - \lambda T_{i-1}^{l+1} = \lambda T_{i-1}^{l} + 2(1-\lambda)T_i^{l} + \lambda T_{i+1}^{l} \ , \quad \text{.....} \quad \textbf{(7.13)}$$

donde $\lambda = k\Delta t/(\Delta x)^2$, también es necesario adecuar dicha ecuación para las condiciones de frontera (primer y último nodo), resultando la **Ec. (7.14)** para el primer nodo:

$$2(1+\lambda)T_1^{l+1} - \lambda T_2^{l+1} = \lambda f_0(t^l) + 2(1-\lambda)T_1^{l} + \lambda T_2^{l} + \lambda f_0(t^{l+1}) \ , \quad \text{.....} \quad \textbf{(7.14)}$$

y **Ec. (7.15)** para el último nodo:

$$-\lambda T_{m+1}^{l+1} + 2(1+\lambda)T_m^{l+1} = \lambda f_{m+1}(t^l) + 2(1-\lambda)T_m^{l} + \lambda T_{m-1}^{l} + \lambda f_{m+1}(t^{l+1}) \ , \quad \text{....} \quad \textbf{(7.15)}$$

resultando de manera similar al método implícito un sistema de ecuaciones lineales de forma tridiagonal, el cual deberá solucionarse empleando el método adecuado.

El algoritmo de implementación se presenta enseguida, así como el diagrama correspondiente se incluye en la **Fig. 7. 3**:

1. Establecer cada valor de las variables involucradas en el método
2. Calcular el valor de λ mediante la expresión:

$$\lambda = k\Delta t/(\Delta x)^2$$

3. Para encontrar las ecuaciones representativas a cada frontera utilizar la **Ec. (7.14)** y **Ec. (7.15)**.
4. Obtener las ecuaciones que representan a los nodos internos al emplear la **Ec. (7.13)**:

   para $l = 1$ y para $i = 1$ hasta m.

5. Representar el primer sistema de ecuaciones en su forma matricial.

6. Resolver el sistema empleando un método para resolver sistemas tridiagonales.
7. Repetir a partir del **paso 4.,** para formar y resolver cada sistema de ecuaciones para cada lapso de tiempo a lo largo de toda la barra.
8. Finalmente se conoce la distribución de la temperatura a lo largo de la barra conforme avanza el tiempo con un método aún más confiable que el explícito e implícito (los resultados pueden graficarse para ver su respetiva variación en tiempo y espacio).

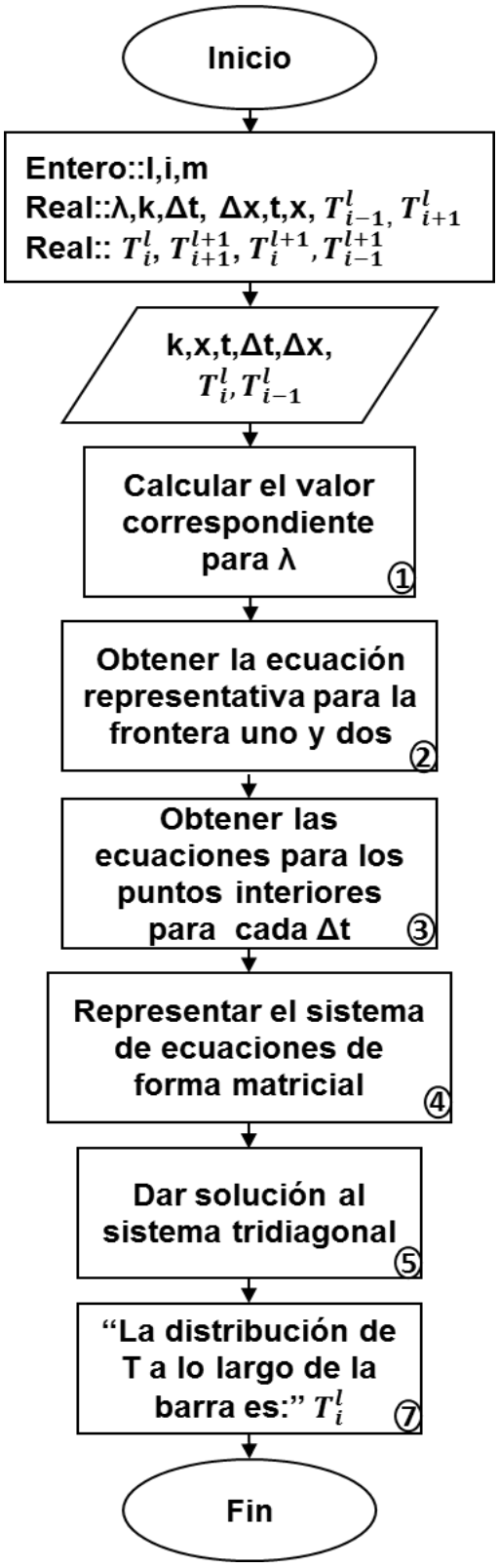


Fig. 7. 3
Método de Crank-Nicolson

7.2.4 Ecuaciones Elípticas

Estas ecuaciones son usadas cuando se busca dar solución a problemas en estado estacionario con valores en la frontera, entre ellos se encuentra el Método de Condiciones de Frontera. Representado esencialmente por la ecuación de Poisson la cual representa la distribución de calor en una placa que se encuentra aislada en la parte media quedando expuestos solo ambos extremos.

La **Ec. (7. 16)** es la ecuación que representa a la ecuación de Laplace, en la cual se asume solamente la distribución de calor e la placa a lo largo del eje *x* y *y*, mientras que en la **Ec. (7. 17)** se consideran fuentes o pérdidas de calor dentro del dominio bidimensional, siendo $f(x,y)$ la función que describe dichas pérdidas, resultando así la ecuación de Poisson.

$$\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}=0 \quad \text{.....} \quad \textbf{(7. 16)}$$

$$\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}=f(x,y) \quad \text{.....} \quad \textbf{(7. 27)}$$

7.2.5 Transformada de Laplace

Es una herramienta matemáticas empleada para la solución de ecuaciones diferenciales, se emplea para aquellas que resultan imposibles de resolver mediante los métodos convencionales, ya que lo que se busca es convertir una ecuación diferencial en una ecuación algebraica; esta sección abordará algunas de sus propiedades, así como el modelo aplicado a las ecuaciones de flujo.

7.2.5.1 Definición de la transformada de Laplace

Sea la función $f(t)$ definida en el intervalo$[0,\infty)$, su transformada $(L\{f(t)\})$ queda definida como la **Ec. (7.18)** que es impropia.

$$L\{f(t)\}(Z)=\bar{F}(Z)=\int_0^{\infty} e^{-Zt}f(Z)dt\ , \quad Z\in R \quad \text{.....} \quad \textbf{(7. 38)}$$

Cuando se trabaja con la transformada de Laplace se pasa del dominio en t a uno de Z, facilitando así el manejo de las ecuaciones pues convierte las

integrales y derivadas en operaciones algebraicas de menor complejidad como lo son multiplicaciones y divisiones; facilitando las soluciones a los problemas planteados.

7.2.5.1 Propiedades fundamentales de la transformada de Laplace

Para dar solución a un problema que ha sido trasladado al espacio de Laplace, es necesario contar con conocimientos de algebra y sus propiedades. En la **Tabla 7. 1** se presentan algunas propiedades para facilitar su manejo en las ecuaciones

Tabla 7. 2
Propiedades elementales de la transformada de Laplace

1	Linealidad	$L\{af(t)+bg(t)\}=aL[f(t)]+bL[g(t)]$
	Cambio de escala	$L\{f(at)\}=\frac{1}{a}F\left(\frac{Z}{a}\right)$
3	Primera propiedad de traslación	$L\{e^{at}f(t)\}=F(Z-a)$
4	Segunda propiedad de translación	$L\{u(t-a)f(t-a)\}=e^{-aZ}L\{f(t)\}$
5	Transformada de una derivada	$L\{f^{(n)}(t)\}=Z^nF(Z)-Z^{n-1}f(0)-Z^{n-2}f'(0)-\cdots-f^{(n-1)}(0)$
6	Derivada de una transformada	$L\{t^nf(t)\}=(-1)^nF^{(n)}(Z)$
7	Transformada de una integral	$\frac{F(Z)}{Z}=L\left\{\int_0^t f(u)du\right\}$
8	Integral de una transformada	$L\left\{\frac{f(t)}{t}\right\}=\int_0^\infty F(u)du$
9	Convolución	$L\{f(t)*g(t)\}=\int_0^\infty e^{-Zt}\left\{\int_0^t f(\tau)g(t-\tau)d\tau\right\}dt$

7.2.5.1 Uso de la transformada de Laplace en problemas de flujo radial

Cuando se presenta el caso de flujo radial, el problema a resolver se basa en la ecuación de difusividad, la cual en términos con variables adimensionales resulta:

$$\frac{1}{r_D}\frac{\partial}{\partial r_D}\left(r_D\frac{\partial p_D}{\partial r_D}\right)=\frac{\partial p_D}{\partial t_D}\ , \qquad \text{..... (7.19)}$$

la cual se encuentra sujeta a la condición inicial y de frontera establecidas para el problema.

Si se define a la transformada de la presión adimensional como (propiedad 6):

$$L\{p_D(r_D,t_D)\}(Z)=\overline{P_D}=\int_0^\infty e^{-Zt_D}p_D\,dt_D\ , \qquad \text{..... (7.20)}$$

aplicando las propiedades 1 y 5, la transformación de **Ec. 7.19** es:

$$\frac{d^2\overline{P_D}}{dr_D^2}+\frac{1}{r_D}\frac{d\overline{P_D}}{dr_D}=s\overline{P_D}-p_D(r_D,0)\ , \qquad \text{..... (7.21)}$$

considerando condiciones iniciales:

$$p_D(r_D,0)=0$$

se obtiene:

$$\frac{d^2\overline{P_D}}{dr_D^2}+\frac{1}{r_D}\frac{d\overline{P_D}}{dr_D}-Z\overline{P_D}=0\ . \qquad \text{..... (7.22)}$$

La **Ec. (7.21)** es una forma de la ecuación de Bessel modificada, donde la solución general se encuentra dada por:

$$\overline{P_D}(r_D,Z)=Al_0\left(\sqrt{Z}r_D\right)+BK_0(\sqrt{Z}r_D) \qquad \text{..... (7.23)}$$

En donde A y B son constantes de integración conocidas al evaluar las condiciones de frontera para cada problema, éstas condiciones también deben ser transformadas al espacio de Laplace.

Ya conocidos A y B, se llega a la solución particular en términos de las variables de Laplace, recordar que la solución final es obtenida invirtiendo la

expresión resultante para regresar al dominio del tiempo, en la **Fig. 7. 4** se representa dicho proceso.

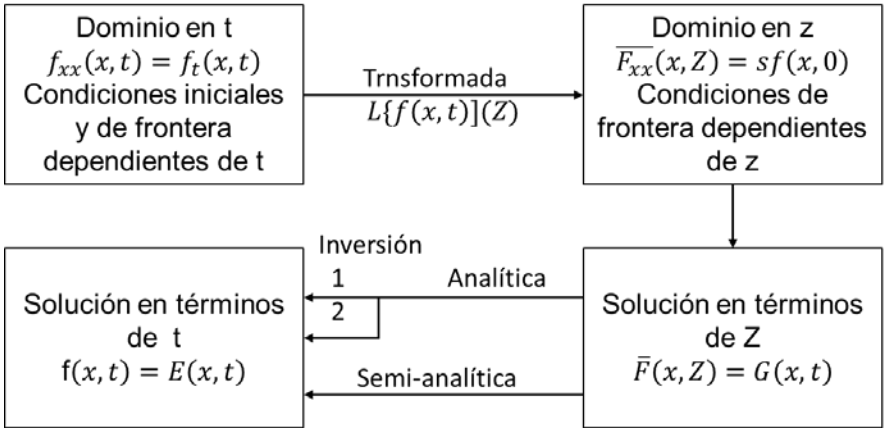


Fig. 7. 4
Proceso de solución de un problema de ecuaciones diferenciales parciales mediante el método de transformada de Laplace. En la trayectoria analítica: 1 indica la inversión completa de la ecuación obtenida en el espacio de Laplace, y 2 las aproximaciones asintóticas por tramos.

La inversión del resultado obtenido puede realizarse de forma analítica, ya sea con función completa o a pedazos, o semi-analítica, en caso de usar un inversor numérico.

Soluciones Analíticas

La obtención de una solución analítica para un problema resuelto mediante la transformada de Laplace, requiere definir a la transformada inversa de Laplace como:

$$L^{-1}\{\bar{F}(Z)\} = L^{-1}\{L\{f(t)\}\} = f(t) \ , \qquad \text{.....} \quad \textbf{(7.24)}$$

y que puede ser obtenido mediante la fórmula de inversión de Mellin (**Ec.(7.25)**).

$$L^{-1}\{\bar{F}(Z)\}(t) = \frac{1}{1\pi i}\int_{\gamma-i\infty}^{\gamma+i\infty} e^{\lambda t} f(\lambda) d\lambda \ , \qquad \text{.....} \quad \textbf{(7.25)}$$

para que exista una solución única $f(t)$ para cada $\bar{F}(Z)$, cumpliéndose que:$f(t) \leftrightarrow \bar{F}(Z)$.

La integración de la **Ec. (7.25)** ocurre en el plano complejo $\lambda = x + iy$, a lo largo de una línea paralela al eje de las ordenadas, la cual se extiende de $-\infty$ a ∞, encontrándose desplazada del origen a una distancia $\boldsymbol{\gamma}$. Debido a la complejidad de la **Ec. (7.25)**, el primer recurso para la inversión de funciones en el espacio de Laplace es mediante tablas.

De acuerdo con la naturaleza, las ecuaciones analíticas pueden ser:

Soluciones completas, cuando se puede invertir la función completa de Laplace y se obtiene una solución continua en todo el dominio del problema.

Soluciones asintóticas, cuando la función en el espacio de Laplace no se puede ser invertida completamente, y pueden ser identificados como comportamientos que característicos a lo largo de una o más secciones del dominio del problema.

Soluciones semi-analíticas

Dado que no siempre pueden obtenerse ecuaciones inversas al problema analizado en el espacio de Laplace, existen diversas razones para que esto ocurra, entre las que se encuentran: que la función no pueda ser expresada como un producto simple, o bien que no se encuentre en alguna tabla, también puede suceder que el problema no esté bien planteado, y que difiera de alguna de éstas condiciones: que exista una solución, que la solución sea única, o que el comportamiento de la solución cambie continuamente con el de las condiciones iniciales.

Cuando no se pueden obtener inversiones analíticas, es necesario emplear algún método numérico para aproximar el resultado con una buena precisión; estos son denominados “métodos de inversión numérica”; para esto.

Métodos de inversión

Estos métodos son dependientes de la sensibilidad del procedimiento, llegando a presentarse casos de divergencia por errores de precisión.

En la **Tabla 7.2** se encuentran las transformaciones necesarias para transformar una función al espacio de Laplace, con la finalidad de poder resolverla de manera óptima.

Tabla 7. 3
Transformada de Laplace

Función $f(t)$	**Transformada**	**Observaciones**
$f(t)$	$L\left\{f(t)\right\} = F(s) = \int_0^\infty e^{-st} f(t)\,dt$	
1	$1/s$	$s > 0$
t	$1/s^2$	$s > 0$
t^n, con $n = 1, 2, 3, \ldots$	$n!/s^{n+1}$	$s > 0$
e^{at}	$1/(s-a)$	$s > a$
$\left(e^{at} - e^{bt}\right)/(a-b)$	$1/(s-a)(s-b)$	$(a \neq b)$
$\left(ae^{at} - be^{bt}\right)/(a-b)$	$s/(s-a)(s-b)$	$(a \neq b)$
$\operatorname{sen} \omega t$	$\omega/\left(s^2 + \omega^2\right)$	$s > 0$
$\cos \omega t$	$s/\left(s^2 + \omega^2\right)$	$s > 0$
$\operatorname{senh} \omega t$	$\omega/\left(s^2 - \omega^2\right)$	$s > \lvert\omega\rvert$
$\cosh \omega t$	$s/\left(s^2 - \omega^2\right)$	$s > \lvert\omega\rvert$

Tabla 7. 3
Transformada de Laplace

Función $f(t)$	Transformada	Observaciones
$e^{at}\,\text{sen}\,\omega t$	$\omega/\left[(s-a)^2+\omega^2\right]$	$s>a$
$e^{at}\cos\omega t$	$(s-a)/\left[(s-a)^2+\omega^2\right]$	$s>a$
te^{at}	$1/(s-a)^2$	$s>a$
$t^n e^{at}$	$n!/(s-a)^{n+1},\ n=1,2,...$	$s>a$
$t\,\text{sen}\,\omega t$	$2\omega s/\left(s^2+\omega^2\right)^2$	$s>0$
$t\cos\omega t$	$\left(s^2-\omega^2\right)/\left(s^2+\omega^2\right)^2$	$s>0$
$(1-\cos\omega t)/\omega^2$	$1/(s^2+\omega^2)$	$s>0$
$y'(t)$	$sY-y(0)$	$Y=L\{y(t)\}$
$y''(t)$	$s^2Y-sy(0)-y'(0)$	$Y=L\{y(t)\}$
$y^{(n)}(t)$	$s^nY-s^{n-1}y(0)-\cdots-y^{(n-1)}(0)$	$n=1,2,3,\ldots$ $Y=\mathrm{L}\{y(t)\}$
$e^{at}f(t)$	$F(s-a)$	$s>a$
$t^n f(t)$	$(-1)^n f^{(n)}(s)$	$n=1,2,3,\ldots$
$\int_0^t f(u)g(t-u)du$	$F(s)\cdot G(s)$	$F(s)=L\{f(t)\}$ $G(s)=L\{g(t)\}$
$\int_0^t f(u)du$	$F(s)/s$	$F(s)=L\{y(t)\}$

Ejercicios de aplicación

Se presentan los ejercicios de algunos métodos numéricos orientados a la ingeniería petrolera.

Ejercicio 1

En el método de entrada de agua de Schilthuis, el área bajo la curva representa la integral $\int_0^t (p_t - p)dt$. La cual puede ser resuelta con cualquier método numérico de integración. La función para evaluar es la siguiente:

$$f(t_k) = (p_i - p_k)$$

La historia de un yacimiento con empuje hidráulico es el siguiente:

t [días]	Presión [psia]	ΔP
0	3500	0
100	3450	50
200	3410	90
300	3380	120
400	3340	160

El comportamiento del acuífero es bajo régimen permanente, con una constante C de entrada de agua de 130 (bpd/psi). Calcular la entrada de agua a los 400 días empleando el método de Simpson 1/3 compuesto. La entrada de agua está definida por la ecuación:

$$We = C \int_0^t (p_i - p)dt$$

Pseudocódigo

1. Identificar la función $f(t_k) = (p_i - p_k)$ a integrar
2. El intervalo de integración $[a, b] = [0{,}400]$, además se trabajará con cuatro intervalos de integración siendo $n = 4$ el cual es un número par.
3. Se puede calcular el espaciamiento h de la siguiente manera $h = \frac{400-0}{4} = 100$

4. La información que se tiene se encuentra de forma tabular
5. La ecuación que se utilizara es la de Simpson 1/3 compuesta
6. Ya que $n = 4 = par$, la integral se obtendrá mediante la **Ec. (5.11)**

$$I = \int_{x_0}^{x_n} f(x)dx = \frac{h}{3}\left[y_0 + y_n + 4\sum_{indice\ impar}^{ordenadas\ con} + 2\sum_{indice\ par}^{ordenadas\ con}\right]$$

Sustituyendo datos:

$$I = \int_{x_0}^{x_n} (\Delta p)dt = \frac{100}{3}[0 + 160 + 4(50 + 120) + 2(90)] = \frac{100}{3}(1020)$$

$$= 34000\ psi$$

7. La aproximación de la integral mediante Simpson 1/3 es 34000.
8. Una vez obtenido el valor de la integral, sustituirlo para calcular la entrada de agua al yacimiento a los 400 días.

$$We = C\int_0^t (p_i - p)dt = \left(130\frac{bls}{psi}\right)(34000\ psi) = 4420000\ bls$$

9. La entrada de agua a los 400 días es de 4 420 000 bls

Código del método de Simpson con menú para seleccionar el caso que se presente.

- Inicio del programa y declaración de variables

```
program ReglasSimpsonterm
implicit none
real::SumPar=0,SumImp=0,SumPri=0,SumUlt=0,SumUlt2=0,Simp1_3,Simp3_8,h
real::SumMulti=0,SumDemas1=0,SumDemas2=0,SumDemas=0,a,b,We,c
real::y0,y1,y2,y3,y4,y5,yn
real::Simp_Comp
integer::opcion,i,n,Int,Int1,Int2,Intervalos,Res
real,allocatable,dimension(:)::Vector,Pares,Impares,Multi,Demas1,Demas2
real,allocatable,dimension(:)::Primer,Ultimo,Ultimo2
```

- Menú, para seleccionar la opción del método a utilizar

```
print*,'|        Selecciona la regla que quieres efectuar (con numero)              |'
print*,'|                                                                            |'
print*,'|                          1) Simpson 1/3                                    |'
print*,'|                          2) Simpson 3/8                                    |'
print*,'|                          3) Simpson 1/3 Compuesta                          |'
print*,'|                          4) Simpson 3/8 Compuesta                          |'
print*,'|                          5) Simpson Combinado Simple                       |'
print*,'|                          6) Simpson Combinado Compuesta                    |'
```

- Método de Simpson 1/3

```
 print*,'Dame el intervalo de Integracion Superior (b)'
 read*,b !LEE EL INTERVALO DE INTEGRACION B
 print*,'Dame el intervalo de Integracion Inferior (a)'
 read*,a !LEE EL INTERVALO DE INTEGRACION A
 h=(b-a)/2 !DEFINE A H PARA SIMPSON 1/3 SIMPLE
 n=3 !DEFINE A N COMO 3
 print*,'Ingresa los valores de y (por espacios o enter)'
 read*,y0,yl,yn !LEE LOS VALORES DE Y
 Simpl_3=(h*(y0+4*yl+yn))/3 !DEFINE LA REGLA DE SIMPSON 1/3 SIMPLE
 Write(*,*)'|           EL VALOR DE LA INTEGRAL NUMERICA POR SIMPSON 1/3 ES:
 print*,'|                          << ',Simpl_3,'  >>                              |'
```

- Método de Simpson 3/8

```
print*,'Dame el intervalo de Integracion Superior (b)'
read*,b !LEE EL INTEVALO DE INTEGRACION B
print*,'Dame el intervalo de Integracion Inferior (a)'
read*,a !LEE EL INTERVALO DE INTEGRACION A
  h=(b-a)/3 !DEFINE H PARA SIMPSON 3/8 SIMPLE
  n=4 !DEINE A N COMO 4
  print*,'Ingresa los valores de y (separados por espacio o enter)'
  read*,y0,yl,y2,yn !LEE LOS VALORES DE Y RESPECTIVAMENTE
  print*
  Simp3_8=(3*h*(y0+3*yl+3*y2+yn))/8 !DEFINE REGLA DE SIMPSON 3/8 SIMPLE
  Write(*,*)'|             EL VALOR DE LA INTEGRAL NUMERICA POR SIMPSON 3/8 ES:
  print*,'|
  write(*,5)'|                              << ',Simp3_8,' >>
```

- Método de Simpson 1/3 múltiple

```
print*,'Cuantos intervalos quieres integrar?'
read*,Int !LEE EL NUMERO DE INTERVALOS
Res=mod(Int,2) !DEFINE AL RESIDUO
if (Res.eq.0)then !CONDICION SI RESIDUO ES IGUAL A CERO
 print*,'ES UN INTERVALO PAR' !IMPEIRME QUE ES UN NUMERO PAR
  print*,'Dame el intervalo de Integracion Superior (b)'
  read*,b !LEE EL INTERVALO DE INTEGRACION B
  print*,'Dame el intervalo de Integracion Inferior (a)'
  read*,a !LEE EL INTERVLO DE INTEGRACION A
  h=(b-a)/Int !DEFINE H PARA SIMPSON 1/3 COMPUESTA
  write(*,*)'El valor de h es:',h !IMPRIME H CON EL FORMATO
  n=Int+1 !DEFINE N PARA SIMPSON 1/3 COMPUESTA
  Allocate(Vector(n)) !LLAMA AL VECTOR "VECTOR" DE TAMAÑO N
  do i=1,n !HACE EL CICLO DESDE 1 HASTA N
  print*,'Ingresa los valores de y',i
  read*,Vector(i) !LEE LOS VALORES DE Y UNO A UNO
  end do !TERMINA EL CICLO
  Allocate(Primer(n)) !LLAMA AL VECTOR PRIMER DE TAMAÑO N
  do i=1,1 !HACE EL CICLO DO DESDE 1 HASTA 1
      Primer(i)=Vector(i) !DEFINE EL ELEMENTO DEL VECTOR PRIMER CON LOS VALORES DEL CICLO
      SumPri=SumPri+Primer(i) !HACE LA SUMA PARA EL CONTADOR SUMPRI
  end do !TERMINA EL CICLO
  Allocate(Ultimo(n)) !LLAMA AL VECTOR ULTIMO CON TAMAÑO N
  do i=n,n !INICIA EL CICLO DO DESDE N HASTA N
      Ultimo(i)=Vector(i) !IGUALA LOS VALORES DE ULTIMO CON EL VALOR DEL CICLO
      SumUlt=SumUlt+Ultimo(i) !HACE LA SUMA PARA EL CONTADOR SUMULT
  end do !FINALIZA EL CICLO
  Allocate(Pares(n)) !LLAMA EL VECTOR PARES
  do i=3,n-1,2 !HACE EL CICLO DESDE 2 HASTA N-1, CON INCREMENTOS DE DOS (ELEMENTOS PARES DEL VECTOR)
      Pares(i)=Vector(i) !IGUALA EL VECTOR PARES CON LOS VALORES DEL CICLO
      SumPar=SumPar+Pares(i) !HACE LA SUMA PARA EL CONTADOR PARES
  end do !TERMINA EL CICLO
  Allocate(Impares(n)) !LLAMA AL VECTOR IMPARES CON TAMAÑO N
  do i=2,n-1,2 !HACE EL CICLO DO DESDE 2 HASTA N -2 CON INCREMENTOS DE 2 (ELEMENTOS IMPARES)
      Impares(i)=Vector(i) !IGUALA EL VECTOR IMPAR CON LOS VALORES DEL CICLO

     SumImp=SumImp+Vector(i) !HACE LA SUMA PARA EL CONTADOR SUMIMP
 end do
 Simpl_3=h*((SumPri+(2*SumPar)+(4*SumImp)+SumUlt)/3) !DEFINE LA REGLA DE SIMPSON 1/3 COMPUESTA
 Write(*,*)'|       EL VALOR DE LA INTEGRAL NUMERICA POR SIMPSON 1/3 MULTIPLE ES:       |'
 print*,'|                          << ',Simpl_3,' >>                          |'
```

- Método de Simpson 3/8 múltiple

```
print*,'Cuantos intervalos quieres integrar?'
read*,Int !LEE LOS INTERVALOS
Res=mod(Int,3) !DEFINE EL RESIDUO DEL INTERVALO ENTRE 3
if (Res.eq.0)then !SI EL RESIDUO ES CERO
  print*,'ES UN INTERVALO MULTIPLO DE 3'
  print*,'Dame el intervalo de Integracion Superior (b)'
  read*,b !LEE EL INTERVALO B
  print*,'Dame el intervalo de Integracion Inferior (a)'
  read*,a !LEE EL INTERVALO A
  h=(b-a)/Int !DEFINE H PARA SIMPSON 3/8 COMPUESTA
  write(*,10)'El valor de h es:',h
  10 format(A17,f6.2,/) !SIGUE EL FORMATO CON ETIQUETA 10
  n=Int+1 !DEFINE A N
  Allocate(Vector(n)) !LLAMA AL ARREGLO VECTOR DE DIMENSIONES N
  do i=1,n !INICIALIZA EL CICLO DESDE 1 HASTA N PARA LOS VALORES DE VECTOR
  print*,'Ingresa los valores de y',i
  read*,Vector(i) !LEE LOS VALORES UNO A UNO
  end do !FINALIZA EL CICLO
  print*,'|                    Los valores de las ordenadas (Yn) son:                  |'
  write(*,8)Vector !IMPRIME EL ARREGLO VECTOR CON ETIQUETA 8
  8 format (1x,100f8.2) !SIGUE EL FORMATO CON LA ETIQUETA 8
  Allocate(Primer(n)) !LLAMA AL ARREGLO PRIMER DE DIMENSIONES N
  do i=1,1 !INICIA EL CICLO DE 1 HASTA 1
      Primer(i)=Vector(i) !IGUALA LOS VALORES DE VECTOR A PRIMER
      SumPri=SumPri+Primer(i) !HACE LA SUMA PARA EL CONTADOR SUMPRI
  end do !FINALIZA EL CICLO
  Allocate(Ultimo(n)) !LLAMA AL ARREGLO ULTIMO CON DIMENSION N
  do i=n,n !HACE EL CICLO  DESDE N HASTA N
      Ultimo(i)=Vector(i) !IGUALA EL VALOR DEL VECTOR AL ARREGLO ULTIMO
      SumUlt=SumUlt+Ultimo(i)!HACE EL CONTADOR SUMULT
  end do !FINALIZA EL CILCO DO
  Allocate(Multi(n))!LLAMA AL ARREGLO MULTI CON DIMENSIONES N
  do i=4,n-1,3 !HACE CICLO PARA I=4 HASTA N-1, CON INCREMENTOS DE TRES, ELEMENTOS MULTIPLOS DE 3
      Multi(i)=Vector(i) !IGUALA LOS VALORES DE VECTOR A MULTI CON LAS CONDICIONES DE DO

    SumMulti=SumMulti+Multi(i) !HACE LA SUMA DEL CONTADOR DE LA SUMA DE LOS MULTIPLOS
end do !FINALIA EL CICLO
Allocate(Demas1(n)) !LLAMA AL AREGLO DEMAS 1 DE DIMENSION N
do i=2,n-1,3 !HACE EL CICLO DO PARA I DESDE 2 HASTA N-1 CON INCREMENTOS DE 3
  Demas1(i)=Vector(i) !IGUALA EL VALOR DE VETOR A DEMAS1 CON LAS CONDICIONES DE DO
  SumDemas1=SumDemas1+Demas1(i) !HACE LA SUMA DEL CONTADOR SUMDEMAS1
end do !TERMINA EL CILCO
Allocate(Demas2(n)) !LLAMA AL ARREGLO DEMAS2 CON DIMENSIONES N
do i=3,n,3 !HACE EL CICLO DO PARA I DESDE 3 HASTA N CON INCREMENTOS DE 3
  Demas2(i)=Vector(i) !IGUALA LOS VALORES DE VECTOR CON DEMAS2 CON LAS CONDICIONES DE DO
  SumDemas2=SumDemas2+Demas2(i) !HACE LA SUMA CON EL CONTADOR SUMDEMAS2
end do !TERMINA EL CICLO DO
SumDemas=SumDemas1+SumDemas2 !HACE LA SUMA DE TODOS LOS DEMAS ELEMENTOS QUE NO SON MULTIPLOS DE 3
Simp3_8=(3*h*(SumPri+2*SumMulti+3*SumDemas+SumUlt))/8 !DEFINE LA REGLA SIMPSON 3/8 COMPUESTA
Write(*,*)'|       EL VALOR DE LA INTEGRAL NUMERICA POR SIMPSON 3/8 MULTIPLE ES:       |'
write(*,9)'|                              << ',Simp3_8,' >>                              |'
```

- Método de Simpson compuesto simple

```
    print*,'Dame el intervalo de Integracion Superior (b)'
    read*,b !LEE EL INTERVALO B
    print*,'Dame el intervalo de Integracion Inferior (a)'
    read*,a !LEE EL INTERVALO A
    h=(b-a)/5 !DEFINE H PARA SIMPSON SIMPLE COMBIANDO
write(*,11)'El valor de h para Simpson 1/3 es:',h
11 format (A35,f5.2) !SIGUE EL FORMATO CON ETIQUETA 11
    print*,'Ingresa los valores de y para Simpson 1/3 (por espacio o enter)'
    read*,y0,y1,y2,y3,y4,y5 !LEE LOS VALORES DE Y PARA ADA VARIABLE
    Simp1_3=(h*(y0+4*y1+y2))/3  !DEFINE REGLA SIMPSON 1/3 SIMPLE
    Simp3_8=(3*h*(y2+3*y3+3*y4+y5))/8 !DEFINE LA REGLA DE SIMPSON 3/8
    Simp_Comp=Simp1_3+Simp3_8 !HACE LA SUMA DE LAS DOS REGLAS
    Write(*,*)'|     EL VALOR DE LA INTEGRAL NUMERICA POR SIMPSON COMBINADO SIMPLE ES:
    print*,'|                          << ',Simp_Comp,'  >>                          |'
```

- Método de Simpson compuesto múltiple

```
502 print*,'Dame el nuemero de intervalos'
read*,Int !LEE EL NUMERO DE INTERVALOS
500 print*,'Dame el numero par para Regla Simpson 1/3 Compuesto'
read*,Int1 !LEE EL NUMERO PAR
Res=mod(Int1,2) !DEFINE EL RESIDUO
    if(Res.eq.0)then !SI RESIDUO ES IGUAL A CERO
    elseif(Res.ne.0)then !SI RESIDUO NO ES IGUAL A CERO
    goto 500 !MANDA A LA LINEA CON ETIQUETA 500
    end if !TERMINA LAS CONDICIONES PARA NUMERO PAR
501 print*,'Dame el numero multiplo de 3 para Simpson 3/8 Compuesto'
read*,Int2 !LEE INTERVALO
Res=mod(Int2,3) !DEFINE EL RESIDUO
    if (Res.eq.0)then !SI RESIDUO ES IGUAL A CERO
    elseif(Res.ne.0)then !SI RESIDUO NO ES CERO
    goto 501 !MANDA A ETIQUETA 502
    end if !TERMINA LAS CODICIONES PARA LOS INTERVALOS
Intervalos=Int1+Int2 !DEFINE INTERVALOS
if(Intervalos.eq.Int)then !SI INTERVALOS ES IGUA A LA VARIABLE INT
print*,'Dame el intervalo de integracion Superior'
read*,b !LEE EL INTRVALO B
print*,'Dame el intervalo de integracion Inferior'
read*,a !LEE EL INTERVALO A
h=(b-a)/Int !DEFINE H
n=Int+1 !DEFINE N
Allocate(Vector(n)) !LLAMA AL ARREGLO VECTOR DE DIMENSIONES N
do i=1,n !INICIALIA EL CICLO DO PARA I DESDE 1 HASTA N
print*,'Ingresa los valores de y',i !PIDE LOS VALORES DE I UNO A UNO
read*,Vector(i) !LEE LO VALORES DE Y UNO A UNO
end do !FINALIZA EL CICLO DO
Allocate(Primer(n)) !LLAMA AL ARREGLO PRIMER CON DIMENSIONES N
do i=1,1 !INICIA EL CICLO DO PARA UNO
Primer(i)=Vector(i) !IGUALA EL VALOR DE VECTOR PARA LA CONDICION DE DO
SumPri=SumPri+Primer(i) !HACE LA SUMA PARA EL CONTADOR SUMPRI
end do
```

```
Allocate(Pares(n)) !LLAMA AL ARREGLO PARES DE DIMENSION N
do i=3,n-(Int2+1),2 !INICIA EL CICLO PARA I=3 HASTA N-(INTE2+1) CON INCREMENTOS DE 2
Pares(i)=Vector(i) !IGUALA LOS VALORES DE VECTOR CON PARES PARA LA CONDICION DO
SumPar=SumPar+Pares(i) !HACE LA SUMA CON EL CONTADOR DE LOS PARES
end do
Allocate(Impares(n)) !LLAMA AL ARREGLO IMPARES CON DIMENSION N
do i=2,n-(Int2+1),2 !INICIA CICLO PARA I=2 HASTA N-(INT2+1) CON INCREMENTOS EN DOS
Impares(i)=Vector(i) !IGUALA LOS ELEMENTOS DE VECTOR A IMPARES DE LA CONDICION ANTERIOR
SumImp=SumImp+Impares(i) !HACE LA SUMA CON EL CONTADOR SUMIMP
end do
Allocate(Ultimo(n)) !LLAMA AL ARREGLO ULTIMO CON DIMENSIONES DE N
do i=n-Int2,n-Int2 !HACE EL CICLO DO PARA I DESDE N-INT2, HASTA N-INT2
Ultimo(i)=Vector(i) !IGUALA EL VELOR DE VECTOR A ULTIMO CON LA CONDICION ANTERIOR
SumUlt=SumUlt+Ultimo(i) !HACE LA SUMA CON EL CONTADOR SUMULT
end do !TERMINA EL CICLO
Simpl_3=(h*(SumPri+2*SumPar+4*SumImp+SumUlt))/3 !DEFINE REGLA SIMPSON 1/3
Y0=SumUlt !DEFINE YO PARA SIMPSON 3/8
Allocate(Multi(n)) !LLAMA AL ARREGLO MULTI DE DIMENSIONES N
do i=(Int2+3),(n-1),3 !HACE EL CICLO PARA I=iNT2+3 HASTA N-1 CON INCREMENTOS DE 3
Multi(i)=Vector(i) !IGUALA LOS VALORES DE VECTOR CON MULTI PARA LA CONDICION ANTERIOR
SumMUlti=SumMulti+Multi(i) !hACE LA SUMA CON EL CONTADOR SUMMULTI
end do
Allocate(Demas1(n)) !LLAMA AL ARREGLO DEMAS1 CON DIMENSIONES N
do i=(Int2+1),(n-1),3 !HACE EL CICLO PARA DO
Demas1(i)=Vector(i) !IGUALA LOS VALORES DE VECTOR CON DEMAS1 CON LA CONDICION ANTERIOR
SumDemas1=SumDemas1+Demas1(i) !HACE LA SUMA CON EL CONTADOR SUMDEMAS1
end do
Allocate(Demas2(n)) !LLAMA AL ARREGLO DEMAS2 CON DIMENSIONES N
do i=(Int2+2),n-1,3 !INICIO DEL CICLO
Demas2(i)=Vector(i) !IGUALA LSO VALORES DE VECTOR CON DEMAS2 CON LA CONDICION ANTERIOR
SumDemas2=SumDemas2+Demas2(i) !HACE LA SUMA CON EL CONTADOR SUMDEMAS2
end do
SumDemas=SumDemas1+SumDemas2 !DECLARA SUM-DEMAS
Allocate(Ultimo2(n)) !LLAMA AL ARREGLO ULTIMO2 CON DIMENSIONES N
do i=n,n !HACE EL CICLO DO PAA I CON EL ULTIMO VALOR

Ultimo2(i)=Vector(i) !IGUALA EL VALOR DEL VECTOR CON ULTIMO2
SumUlt2=SumUlt2+Ultimo2(i) !HACE LA SUMA CON EL CONTADOR SUMULT2
end do
Simp3_8=(3*h*(Y0+SumUlt2+3*SumDemas+2*SumMulti))/8 !DEFINE REGLA SIMPSON 3/8 COMPUESTO
print*,'|                EL RESULTADO TOTAL DE LA INTEGRAL ES:                    |'
Simp_Comp=Simp1_3+Simp3_8 !HACE LA SUMA DE LAS DOS REGLAS
print*,'|                          ((',Simp_Comp,'))                             |'
```

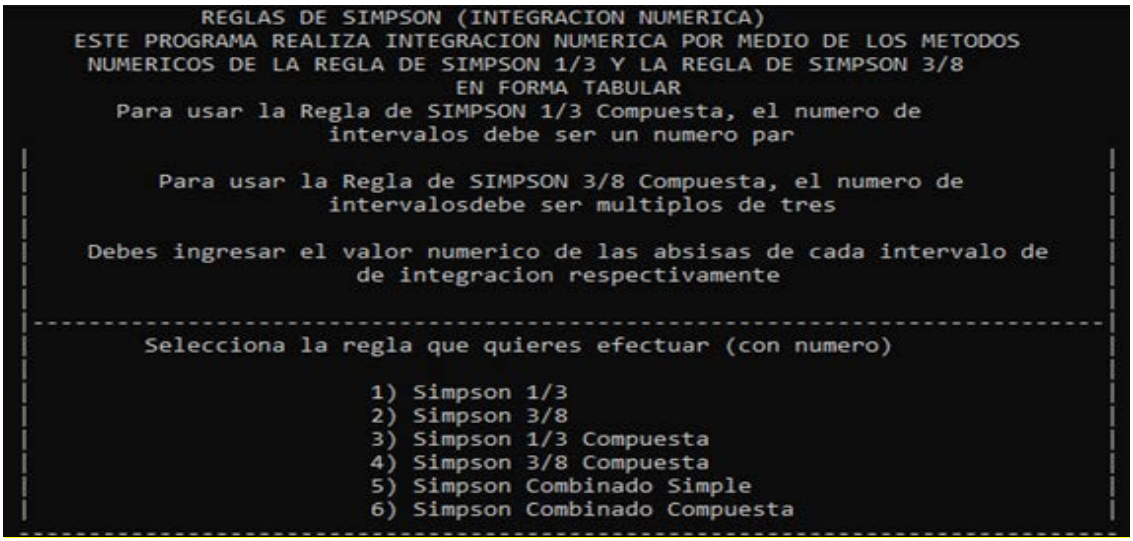


Figura 7. 5
Salida de pantalla del menú del método de Simpson

Figura 7.6
Salida de pantalla del ejercicio 1 resuelto mediante el método de Simpson 1/3 compuesta

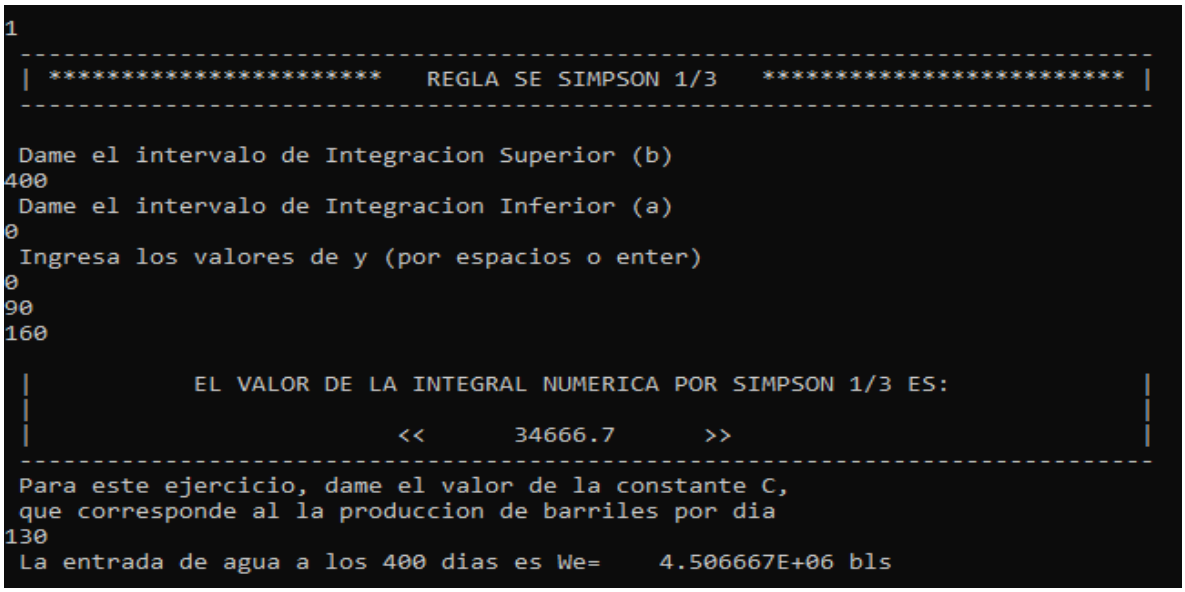


Fig. 7.7

Salida de pantalla del ejercicio 1 resuelto mediante el método de Simpson 1/3 simple

Como puede observarse, al realizar el ejercicio mediante el método de Simpson 1/3 simple y compuesto el resultado es variable. Esto debido a la precisión en el manejo de los intervalos tal como se explica en el **Capítulo 5**.

Ejercicio 2 y 3

En un cilindro cerrado se encuentra propano puro a 100°F, presentándose en fase liquida y vapor. Empleando la ecuación cúbica de estado de Van der Walls y de Redlich-Kwong, calcular la densidad de las fases líquida y vapor (gas).

La presión de vapor obtenida mediante la carta de Cox se obtiene una Pv=185 [psia] Tc=666 [°R], Pc=616.3 [psia], M=44 y una Tabs=560 [°R].

Ecuación de estado cubica de Van der Walls: $z^3 - (1 + B)z^2 + Az - Ab$

Ecuación de estado cubica de Redlich Kwong: $z^3 - z^2 + (A - B - B^2)z - Ab = 0$

Pseudocódigo para Van der Walls

1. Obtener el valor de z_v y z_l, mediante la ecuación de Van der Walls
2. Obtener el valor de las constantes A y B, mediante las siguientes ecuaciones:

$$a = \mho_a \frac{R^2 T^2}{P_c}$$

$$= (0.421875) \frac{(10.732\ [lb/pg^2 abs - ft^3/lbm - mol°R])^2 (666\ [°R])^2}{(616.3\ [lb/pg^2 abs]}$$

$$= 34\ 970.5 \left[\frac{lb/pg^2 abs - (ft^3)^2}{(lbm - mol)^2}\right]$$

$$b = \mho_a \frac{RT_c}{P_c} = (0.125) \frac{(10.732\ [lb/pg^2 abs - ft^3/lbm - mol°R])(666\ [°R])}{(616.3\ [lb/pg^2 abs]}$$

$$= 1.44968 \left[\frac{ft^3}{lbm - mol}\right]$$

$$A = \frac{aP_v}{R^2 T_{abs}^2} = \frac{\left(34\ 970.5 \left[\frac{lb/pg^2 abs - (ft^3)^2}{(lbm - mol)^2}\right]\right)\left(185 \left[\frac{lb}{pg^2 abs}\right]\right)}{(10.732\ [lb/pg^2 abs - ft^3/lbm - mol°R])^2 (560\ [°R])^2}$$

$$= 1.179117$$

$$B = \frac{bP_v}{RT_{abs}} = \frac{(1.44968\ [ft^3/lbm - mol])(185\ [lb/pg^2 abs]}{(10.732\ [lb/pg^2 abs - ft^3/lbm - mol°R])(560\ [°R])} = 0.04462$$

3. Sustituyendo A y B en la ecuación de estado cubica de Van der Walls

$$z^3 - (1 + B)z^2 + Az - Ab = 0$$

$$z^3 - 1.04462z^2 + 0.179117z - 0.007993 = 0$$

4. la primera raíz se obtendrá a partir de emplear el método de Newton-Raphson.
5. Mediante división sintética, obtener una ecuación de segundo grado, la cual se resolverá mediante la ecuación general obteniendo así las dos raíces faltantes del polinomio.
6. La raíz de valor más grande corresponderá a la fase vapor, mientras que la más pequeña a la fase líquida.

   $z_v = 0.8435$ y $z_l = 0.07534$
7. Obtener la densidad de la fase vapor mediante la siguiente ecuación:

$$\rho^v = \frac{P_v M_{c3}}{z_v RT} = \frac{(185\ [lb/pg^2 abs])(44\ lbm/lbm - mol)}{0.8435\ \ (10.732\ [lb/pg^2 abs - ft^3/lbm - mol°R])(560\ [°R])}$$

$$= 1.6057\ (lbm/ft^3)$$

8. Obtener la densidad de la fase líquido mediante la siguiente ecuación:

$$\rho^l = \frac{P_v M_{c3}}{z_l RT} = \frac{\left(185\ \left[\frac{lb}{pg^2 abs}\right]\right)\left(44\frac{lbm}{lbm} - mol\right)}{0.07534\ \ (10.732\ [lb/pg^2 abs - ft^3/lbm - mol°R])(560\ [°R])}$$

$$= 17.9765\ \left(\frac{lbm}{ft^3}\right)$$

Pseudocódigo para Redlich Kwong

1. Obtener el valor de z_v y z_l, empleando la ecuación de Redlich Kwong
2. Obtener el valor de las constantes A y B, mediante las siguientes ecuaciones:

$$a = \mho_a \frac{R^2 T_C^{2.5}}{P_c}$$

$$= (0.42747)\frac{(10.732\ [lb/pg^2 abs - ft^3/lbm - mol°R])^2 (666\ [°R])^{2.5}}{(616.3\ [lb/pg^2 abs]}$$

$$= 914\ 450.86\left[\frac{lb/pg^2 abs - (ft^3)^2}{(lbm - mol)^2}\right]$$

$$b = \mho_b \frac{RT_c}{P_c} = (0.08664)\frac{(10.732\ [lb/pg^2 abs - ft^3/lbm - mol°R])(666\ [°R])}{(616.3\ [lb/pg^2 abs])}$$

$$= 1.0048\left[\frac{ft^3}{lbm - mol}\right]$$

$$A = \frac{aP_v}{R^2 T_{abs}^{2.5}} = \frac{\left(914\ 450.865\left[\frac{lb/pg^2 abs - (ft^3)^2}{(lbm - mol)^2}\right]\right)\left(185\left[\frac{lb}{pg^2 abs}\right]\right)}{(10.732\ [lb/pg^2 abs - ft^3/lbm - mol°R])^2 (560\ [°R])^2}$$

$$= 0.197925$$

$$B = \frac{bP_v}{RT_{abs}} = \frac{(1.0048[ft^3/lbm - mol])(185\ [lb/pg^2 abs]}{(10.732\ [lb/pg^2 abs - ft^3/lbm - mol°R])(560\ [°R])} = 0.03093$$

3. Sustituyendo A y B en la ecuación de estado cubica de Redlich Kwong

$$z^3 - z^2 + (A - B - B^2)z - AB = 0$$

$$z^3 - z^2 + 0.166038z - 0.0061218 = 0$$

4. la primera raíz se obtendrá a partir de emplear el método de Newton-Rapson.
5. Mediante división sintética, obtener una ecuación de segundo grado, la cual se resolverá mediante la ecuación general obteniendo así las dos raíces faltantes del polinomio.

6. La raíz de valor más grande corresponderá a la fase vapor, mientras que la más pequeña a la fase líquida.

   $z_v = 0.802637$ y $z_l = 0.05237$

7. Obtener la densidad de la fase vapor mediante la siguiente ecuación:

$$\rho^v = \frac{P_v M_{c3}}{z_v RT} = \frac{(185\ [lb/pg^2 abs])(44\ lbm/lbm-mol)}{0.802637\ \ (10.732\ [lb/pg^2 abs - ft^3/lbm-mol°R])(560\ [°R])}$$
$$= 1.68747\ (lbm/ft^3)$$

8. Obtener la densidad de la fase líquido mediante la siguiente ecuación:

$$\rho^l = \frac{P_v M_{c3}}{z_l RT} = \frac{(185\ [lb/pg^2 abs])(44\ lbm/lbm-mol)}{0.05237\ \ (10.732\ [lb/pg^2 abs - ft^3/lbm-mol°R])(560\ [°R])}$$
$$= 25.6823\ (lbm/ft^3)$$

Código de la solución

- Declaración de variables a emplear y menú para seleccionar la ecuación a trabajar

```
real::a,b,R,T,Tc,c,d,Pv,Aa,Bb,Pc,fz,dfz,f,e,i,zl,n,z,mm
real::q,z2,z3,zr,zi,M,densva,densli,zl,zv,opcion,menu,h
real::disc,xa,xb,xi,xr,xl,si
10 print*, "     Solucion a ecuacion de estado cubica"
print*,"Emplear ecuacion de Van der Walls, presionar 1"
print*,"Emplear ecuacion de Redlich-Kwong, presionar 2"
```

- solicitud de las propiedades de la mezcla

```
Print*,"Ecuacion de Van der Walls"
Print*,""
Print*,"ingresar la presion de vapor, [psia]"
read*, Pv
Print*,"ingresar la presion critica, [psia]"
read*, Pc
Print*,"ingresar la temperatura del sistema, [°R]"
read*, T
Print*,"ingresar la temperatura critica, [°R]"
read*, Tc
Print*,"ingresar la masa molecular del fluido, [lbm/lbm-mol]"
read*, M
```

- Si del menú se elige trabajar con la ecuación de Van der Walls (1), se calcula A y B, y son sustituidas en la ecuación general

```
!calculos para obtener variables para calcular el valor de las
!incognitas que emplea esta ecuacion
a=(c*(R**2)*(Tc**2))/Pc
b=(d*R*Tc)/Pc
Aa=(a*Pv)/((R**2)*(T**2))
Bb=(b*Pv)/(R*T)
print*, "los valores de las variables utilizadas en la ecuacion son:"
Print*, "     A= ",Aa,"      B= ",Bb
Print*,""
print*,"La ecuacion de estado de Van der Walls es"
print*,"  z^3-(1+B)(z^2)+Az-AB=0"
print*, " Sustituyendo A y B..."
Print*, "z^3-",1+Bb,"z^2+",Aa,"z",-Aa*Bb
```

- Cálculo de la primera raíz mediante el método de Newton-Raphson empleando un ciclo hasta que el error entre la z sea menor a 0.000001

```
Print*,"PARA OBTENER LA PRIMER RAIZ DE LA ECUACION SE EMPLEA EL METODO"
print*,"  DE NEWTON-RAPHSON, USANDO COMO VALOR INICIAL UNA z=0.01"
z=0.01
i=1
do
  2 fz=(z**3)-((1+Bb)*z**2)+(Aa*Z)-Aa*Bb
  dfz=(3*z**2)-(2*(1+Bb)*z)+Aa
  f=z-(fz/dfz)   !Ecuacion general del metodo de NEWTON-RAPHSON
  e=abs(z-f)/z
  if(e>0.000001)then
    i=i+1
    z=f
go to 2
else
  go to 3
  end if
  end do
3 print*,"La primera raiz es Z1= ",z
```

- Obtención de las dos raíces faltantes

```
Print*,"Para obtener las siguiente raiz serealiza una divicion sintetica"
print*,"a la funcion original, para obtener una ecuacion de segundo grado"
print*,"           y se resolvera por la ecuacion general"
mm=1          !A
n=z-(1+Bb)   !B
q=(z**2)+Aa-z*(1+Bb)   !C
!Resolviendo ecuacion
disc=(n*n)-(4*mm*q)
if(disc.GT.0.0)then
  z2=(-n+(sqrt(disc)))/(2*mm)
   h=z2
  z3=(-n-(sqrt(disc)))/(2*mm)
 densva=(185*44)/(h*R*T)
print*,""
print*,"el valor de la segunda raiz Z2= ",z2
print*,"el valor de la tercera raiz Z3= ",z3
else if(disc.EQ.0)then
  Z2=(-n)/(2.0*mm)
Print*, " la ecuacion solo tiene una raiz Z2= ",z2
densli=(Pv*M)/(z2*R*T)
else
  z2=(-n/(2.0*mm))
  Zi=(sqrt(-disc)/(2.0*mm))
print*,"Se tiene una raiz real(Z2)= ",zr
print*,"Se tiene una raiz imaginaria (Z3)= ",zi
h=zr
```

- Cálculo de la densidad del líquido y vapor

```
densli=(Pv*M)/(z*R*T)    densva=(Pv*M)/(z2*R*T)
```

- Si del menú se elige trabajar con la ecuación de Redlinch Kwong (2), se calcula A y B, y son sustituidas en la ecuación general

```
a=(c*(R**2)*(Tc**2.5))/(Pc)
b=(d*R*T)/(Pc)
Aa=(a*Pv)/((R*2)*(T**2.5))
Bb=(b*Pv)/(R*T)
print*, "los valores de las variables utilizadas en la ecuacion son:"
Print*, "     A= ",Aa,"      B= ",Bb
print*,"La ecuacion de estado de Redlich-Kwong es"
print*,"  z^3-z^2+(A-B-B^2)z-AB=0"
print*, " Sustituyendo A y B..."
Print*, " z^3-z^2+",(Aa-Bb-(Bb**2)),"z",-A*B
```

- Cálculo de la primera raíz mediante el método de Newton-Raphson empleando un ciclo hasta que el error entre la z sea menor a 0.000001

```
Print*,"PARA OBTENER LA PRIMER RAIZ DE LA ECUACION SE EMPLEA EL METODO"
print*,"  DE NEWTON-RAPHSON, USANDO COMO VALOR INICIAL UNA z=0.01"
z=0.01
i=1
do
4 fz=(z**3)-(z**2)+((Aa-Bb-(Bb**2))*z)-(Aa*Bb)
  dfz=(3*z**2)-(2*z)+(Aa-Bb-(Bb**2))
  f=z-(fz/dfz)    !Ecuacion general de Newton-Raphson
  e=abs(z-f)/z
  if(e>0.000001)then
    i=i+1
    z=f
    goto 4
    else
      go to 5
  end if
  end do
 5 print*," La primera raiz de la ecuacion es Z1= ",z
```

- Obtención de las dos raíces faltantes

```
Print*,"Para obtener las siguiente raiz serealiza una divicion sintetica"
print*,"a la funcion original, para obtener una ecuacion de segundo grado"
print*,"            y se resolvera por la ecuacion general"
mm=1          !A
n=-1+z   !B
q=(Aa-Bb-(Bb**2))+z*(-1+z)   !C
disc=(n*n)-(4*mm*q)
if(disc.GT.0.0)then
  z2=(-n+(sqrt(disc)))/(2*mm)
  Zv=z2
  z3=(-n-(sqrt(disc)))/(2*mm)
```

- Cálculo de la densidad del líquido y vapor

```
densva=(185*44)/(Zv*R*T)    densli=(Pv*M)/(z*R*T)
```

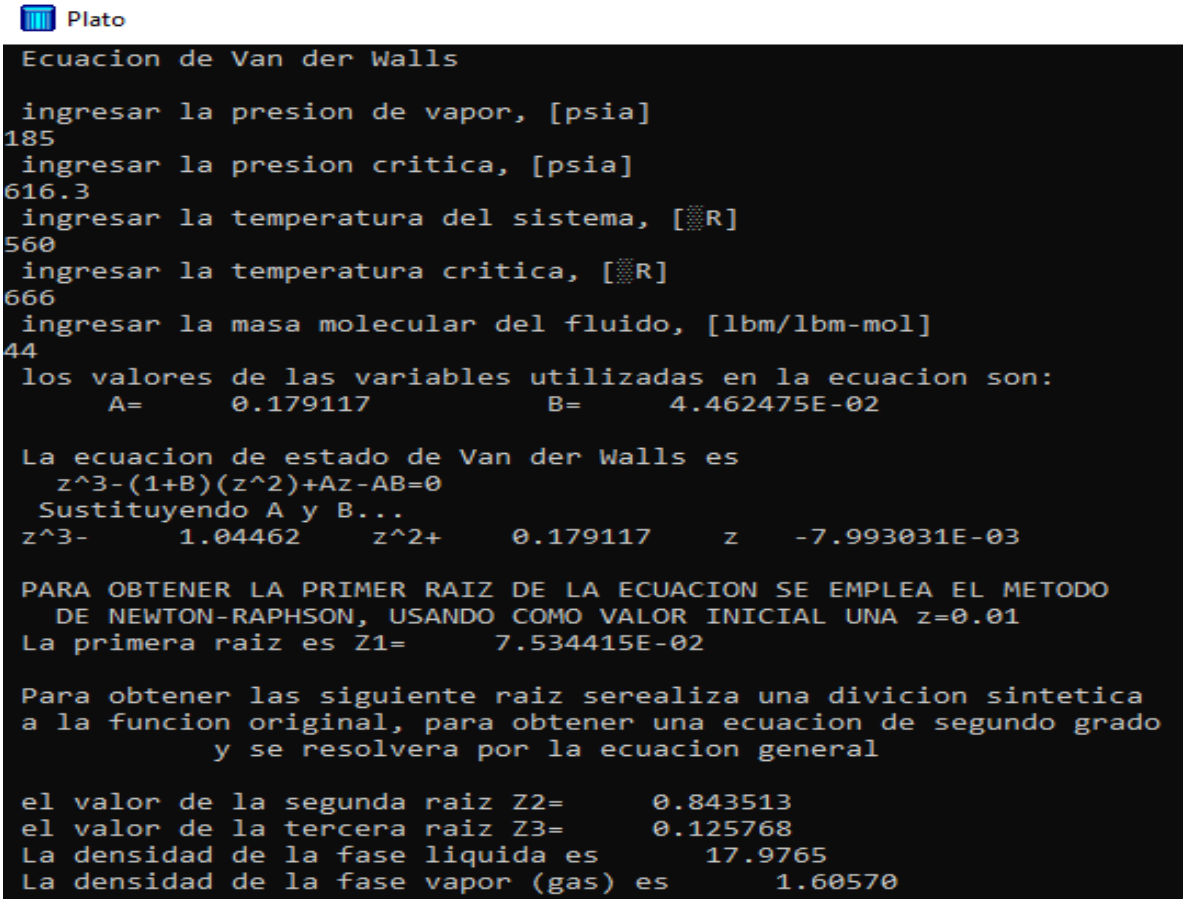


Figura 7.8
Salida de pantalla del ejercicio 2. Ecuación de Van der Walls, empleando el método de Newton-Raphson

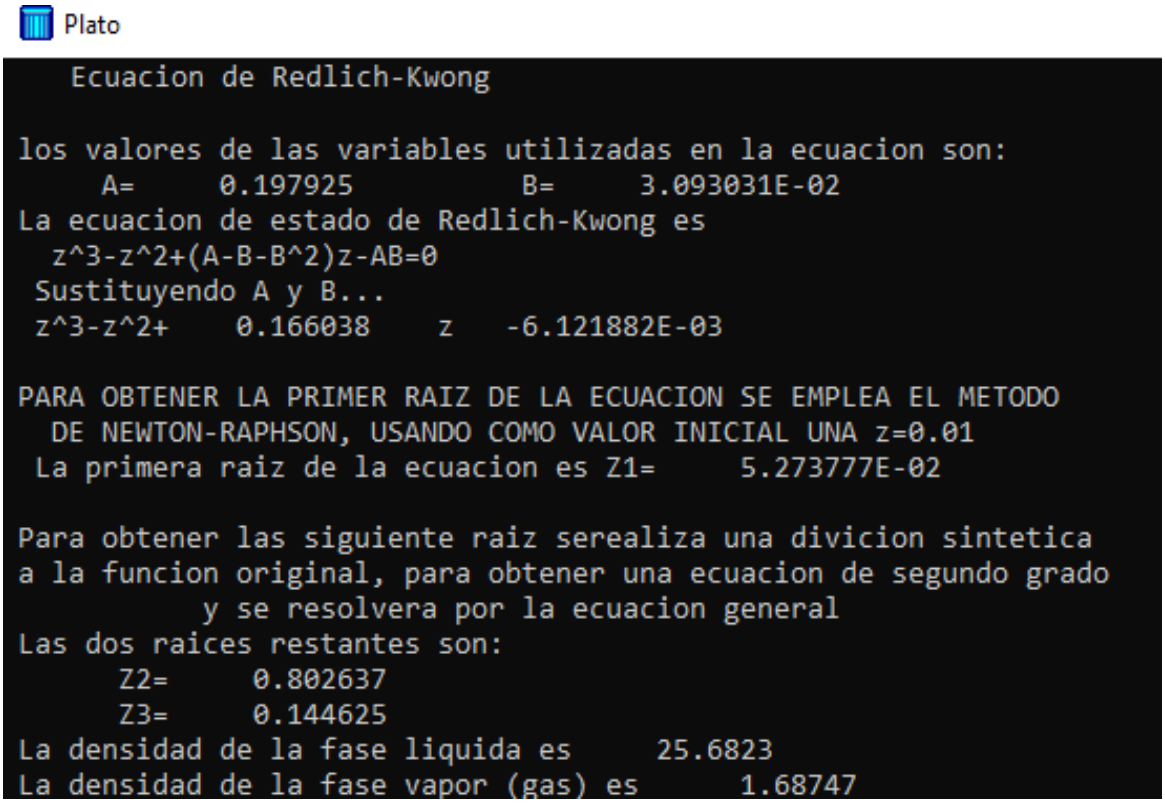


Figura 7.9
Salida de pantalla del ejercicio 3. Ecuación de Redlich-Kwong Walls, empleando el método de Newton-Raphson

Ejercicio 4

Aplicar el método de Crank-Nicolson para dar solución a la ecuación de calor en su forma unidimensional y obtener la temperatura cuando la longitud *x* de una barra de aluminio es de **2 [cm]** a un tiempo **t=10 [s]** en una barra larga y delgada, la cual se encuentra aislada exceptuando sus extremos, las propiedades de la barra son las siguientes:

Longitud	10 [cm]	C_p	0.2174 [Cal/g°C]
k	0.49 [Cal/scm°C]	ρ	2.7 [g/cm^3]
Δx	2 [cm]	α= ρ/ C_p	0.835 [cm^2/s]
Δt	5 [s]		

Cuando t=0, la temperatura de la barra es de 0 [°C] y las condiciones de frontera se fijan para todos los tiempos en $T\ (0)\ = 100\ [°C]$ y $T(10) = 50\ [°C]$.

Pseudocódigo

1. Obtener el valor de lamda

$$\lambda = \frac{\alpha \Delta t}{(\Delta x)^2} = \frac{0.835 * 5}{2^2} = 1.04375$$

2. Obtener cada ecuación del sistema para el primer tiempo $t = 0$ y x varia de dos en dos hasta ocho centímetros

2.1 Cuando $t = 0$ y $x = 4\ [cm]$

$$2(1 + 1.04375)T_1^1 - 1.04375T_2^1 =$$

$$1.04375(100) + (2(1 - 1.04375)0) + 1.04375(0) + 1.04375(0) + 1.04375(100)$$

$$4.0875T_1^1 - 1.04375T_2^1 = 208.75$$

2.2 Cuando $t = 0$ y $x = 4\ [cm]$

$$-1.04375T_1^1 + 2(1 + 1.04375)T_2^1 - 1.04375T_3^1 =$$

$$1.04375(0) + 2(1 - 1.04375)(0) + 1.04375(0)$$

$$-1.04375T_1^1 + 4.0875T_2^1 - 1.04375T_3^1 = 0$$

2.3 Cuando $t = 0$ y $x = 6\ [cm]$

$$-1.04375T_2^1 + 2(1 + 1.04375)T_3^1 - 1.04375T_4^1 =$$

$$1.04375(0) + 2(1 - 1.04375)(0) + 1.04375(0) =$$

$$-1.04375T_2^1 + 4.0875T_3^1 - 1.04375T_4^1 = 0$$

2.4 Cuando $t = 0$ y $x = 8\ [cm]$

$$-1.04375T_3^1 + 2(1 + 1.04375)T_4^1 =$$

$$1.04375(50) + 2(1 - 1.04375)(0) + 1.04375(0) + 1.04375(50) =$$

$$-1.04375T_3^1 + 4.0875T_4^1 = 104.375$$

3. Con los resultados anteriores se obtiene el siguiente sistema de ecuaciones

$$\begin{bmatrix} 4.0875 & -1.04375 & 0 & 0 \\ -1.04375 & 4.0875 & -1.04375 & 0 \\ 0 & -1.04375 & 4.0875 & -1.04375 \\ 0 & 0 & -1.04375 & 4.0875 \end{bmatrix} \begin{bmatrix} T_1^1 \\ T_2^1 \\ T_3^1 \\ T_4^1 \end{bmatrix} = \begin{bmatrix} 208.750 \\ 0 \\ 0 \\ 104.375 \end{bmatrix}$$

4. Resolviendo el sistema de ecuaciones mediante el método de Thomas para $t = 5\ [s]$ se tiene

$$\begin{bmatrix} T_1^1 \\ T_2^1 \\ T_3^1 \\ T_4^1 \end{bmatrix} = \begin{bmatrix} 55.4456 \\ 17.1345 \\ 11.6558 \\ 28.5115 \end{bmatrix}$$

5. Siendo que $\Delta t = 5$, falta obtener el otro sistema de ecuaciones por lo que se repiten los pasos 2 al 4 para tiempo $t = 5$ para obtener los valores en $t = 10$

2. Obtener cada ecuación del sistema para el primer tiempo $t = 0$ y x varia de dos en dos hasta ocho centímetros

   2.1 Cuando $t = 5$ y $x = 4\ [cm]$

$$2(1 + 1.04375)T_1^2 - 1.04375T_2^2 =$$

$$1.04375(100) + (2(1 - 1.04375)55.4456) + 1.04375(0) + 1.04375(17.1345) + 1.04375(100)$$

$$4.0875T_1^2 - 1.04375T_2^2 = 221.7826$$

   2.2 Cuando $t = 5$ y $x = 4\ [cm]$

$$-1.04375T_1^2 + 2(1 + 1.04375)T_2^2 - 1.04375T_3^2 =$$

$$1.04375(55.4456) + 2(1 - 1.04375)(17.1345) + 1.04375(11.6558)$$

$$-1.04375T_1^2 + 4.0875T_2^2 - 1.04375T_3^2 = 68.5378$$

   2.3 Cuando $t = 5$ y $x = 6\ [cm]$

$$-1.04375T_2^2 + 2(1 + 1.04375)T_3^2 - 1.04375T_4^2 =$$

$$1.04375(17.1345) + 2(1 - 1.04375)(11.6558) + 1.04375(28.5115) =$$

$$-1.04375T_2^2 + 4.0875T_3^2 - 1.04375T_4^2 = 46.6231$$

2.4 Cuando $t = 5$ y $x = 8\ [cm]$

$$-1.04375T_3^2 + 2(1 + 1.04375)T_4^2 =$$

$$1.04375(50) + 2(1 - 1.04375)(24.5115) + 1.04375(11.6558) + 1.04375(50) =$$

$$-1.04375T_3^2 + 4.0875T_4^2 = 114.046$$

3. Con los resultados anteriores se obtiene el siguiente sistema de ecuaciones

$$\begin{bmatrix} 4.0875 & -1.04375 & 0 & 0 \\ -1.04375 & 4.0875 & -1.04375 & 0 \\ 0 & -1.04375 & 4.0875 & -1.04375 \\ 0 & 0 & -1.04375 & 4.0875 \end{bmatrix} \begin{bmatrix} T_1^2 \\ T_2^2 \\ T_3^2 \\ T_4^2 \end{bmatrix} = \begin{bmatrix} 221.7826 \\ 68.53780 \\ 46.62310 \\ 114.0460 \end{bmatrix}$$

4. Resolviendo el sistema de ecuaciones mediante el método de Thomas para $t = 5\ [s]$ se tiene

$$\begin{bmatrix} T_1^2 \\ T_2^2 \\ T_3^2 \\ T_4^2 \end{bmatrix} = \begin{bmatrix} 64.7925 \\ 41.2519 \\ 31.0920 \\ 35.8405 \end{bmatrix}$$

5. Siendo que ya se acabó concluyó en tiempo y longitud, y que el ejercicio pide obtener la temperatura cuando la longitud x de la barra de aluminio es de 2 [cm] a un tiempo t=10 [s]

   $T(2,10)$ =64.7925 [°C]

Código de la solución

- Declaración del tipo de variables a emplear

```
real,allocatable::x(:),u(:),a(:),b(:),c(:),d(:)
real::long,l,dif,dx,dt,ti,td,tin
integer::j,ni,m,it
```

- Solicitud del ingreso de algunas variables y propiedades por el usuario

```
print*,"Cual es la longitud de la barra [cm]?"
read*,long
Print*,"Cual es la tempetatura inicial de la barra [°C]?"
read*,tin
print*,"Cual es el valor de los incrementos temporales [s]?"
read*,dt
20 print*,"Cual es el valor de los incrementos espaciales [cm]?"
read*,dx
m=ceiling((long/dx)-1)!numero de nodos, que determinana el numero de ecuaciones
!e incognitas a resolver en cada iteracion
if((dx.ge.long).or.(m.le.2))then!restringe el uso de dx mayor a la longitud de la barra
  print*,"El incremento es demasiado grande, introducir un nuevo valor"
  goto 20
  else
print*,"Cual es el valor de la difusividad termicadel"
print*,"    material de la barra[cm2/s]?             "
read*,dif
print*,"Cual es la temperatura en el extremo izquierdo [°C]?"
read*,ti
print*,"Cual es la temperatura en el extremo derecho [°C]?"
read*,td
print*,"Cual es el numero de iteraciones a realizar en el timepo"
read*,it
```

- Cálculo de lambda y declaración de los vectores a emplear

```
l=dif*(dt/dx**2)!Calculo de lambda
allocate(x(0:m),u(0:m+1),a(m),b(m),c(m),d(m))!vectores a utilizar
open(10,file="crank_nicolson.txt",status='unknown')!se crea archivo que
!guarda los datos Para condiciones iniciales
do j=1,m
  x(j)=j*d
  u(j)=tin
  end do
```

- Se hace un ciclo para asignar a cada matriz las diagonales correspondientes

```
!haciendo las iteraciones
do ni=1,it
  !asignacion de los valores de la matriz tridiagonal
  a=-l
  b=2*(1+l)
  c=-l
!asignacion de valores de los coeficientes independientes de la matriz
  d(1)=l*ti+2*(1+l)*u(1)+l*u(2)+l*ti !frontera izquierda
  d(m)=l*u(m-1)+2*(1-l)*u(m)+l*td+l*td !frontera derecha
  do j=2,m-1 !para nodos intermedios
    d(j)=l*u(j-1)+(2*(1-l))*u(j)+l*u(j+1)
    end do
```

- Solución del sistema tridiagonal resultante mediante el método de Thomas

```
!SOLUCION DEL SISTEMA TRIDIAGONAL RESULTANTE EMPLEANDO EL METODO DE THOMAS
 call thomas(a,b,c,d) !subroutine thomas (a,b,c,d)
 do j=1,m
u(j)=d(j)!el vector resultado de thomas se asigna a U(j) para para la sig iteracion
 end do
Print*,"cuando el tiempo t=", dt*ni,"[s]"
print*,"   x [cm]       T[°C]"
 write(10,*)"0.00000",ti
 do j=1,m
print*,x(j),u(j)
write(10,*)x(j),u(j)
end do
print*,""
write(10,*)long,td
print*,""
end do
close (10)
call system ("gnuplot curva.txt")
```

- Construcción de la matriz final y solución mediante Thomas, finalización del programa

```
!Metodo de thomas para dar solucion al sistema de ecuaciones
! con matriz de coeficientes de forma tridiagonal
subroutine thomas (a,b,c,d)
  real,intent(inout)::a(:),b(:),c(:),d(:)
  integer j
  do j=2,m
    a(j)=a(j)/b(j-1)
    b(j)=b(j)-a(j)*c(j-1)
    d(j)=d(j)-a(j)*(d(j-1))
    end do

d(m)=d(m)/b(m)
do j=m-1,1,-1<
  d(j)=(d(j)-c(j)*d(j+1))/b(j)
  end do
  end subroutine thomas
    end program Cranknicolson
```

```
 Ejemplo de aplicacion del metodo implicito de Crank-Nicolson
 Calculo de la ecuacion de calor en su forma unidimensional

 Calculo de la distribucion de la temperatura(de izquierda a derecha)
 con condiciones de temperatura iniciales y en la frontera para una
 barra larga y delgada, aislada en todos los puntos, menos en los
 extremos
 Cual es la longitud de la barra [cm]?
10
 Cual es la tempetatura inicial de la barra [ C]?
0
 Cual es el valor de los incrementos temporales [s]?
5
 Cual es el valor de los incrementos espaciales [cm]?
2
 Cual es el valor de la difusividad termicadel
     material de la barra[cm2/s]?
0.835
 Cual es la temperatura en el extremo izquierdo [ C]?
100
 Cual es la temperatura en el extremo derecho [ C]?
50
 Cual es el numero de iteraciones a realizar en el timepo
4
```

Fig. 7. 10

Salida de pantalla del ejercicio 4, Ecuación de calor

```
resultados

cuando el tiempo t=     5.00000    [s]
   x [cm]       T[ C]
     2.00000         55.4457
     4.00000         17.1345
     6.00000         11.6558
     8.00000         28.5115


cuando el tiempo t=     10.0000    [s]
   x [cm]       T[ C]
     2.00000         64.7925
     4.00000         41.2519
     6.00000         31.0919
     8.00000         35.8405


cuando el tiempo t=     15.0000    [s]
   x [cm]       T[ C]
     2.00000         73.9301
     4.00000         53.7022
     6.00000         43.9507
     8.00000         43.9302


cuando el tiempo t=     20.0000    [s]
   x [cm]       T[ C]
     2.00000         79.1827
     4.00000         62.5881
     6.00000         52.5439
     8.00000         49.2348
```

Fig. 7. 11

Variación de la temperatura en el tiempo y espacio

Ejercicio 5

Calcular la distribución de la presión en un yacimiento a los 360 días de producción, empleando el método de Explicito considerar las siguientes características y un $\Delta t = 100$ días:

$B = 1.05\ [bbl/STB]$	$\phi = 0.22$	$\Delta x = 1000\ [ft]$
$B_0 = 1.03\ [bbl/STB]$	$q_1 = q_2 = q_3 = q_5 = 0$	$\Delta z = 900\ [ft]$
$\mu = 3\ [cp]$	$q_4 = -125$	$\Delta y = 85\ [ft]$
$Co = 0.0000035\ [psi^{-1}]$	$Pi = 6500\ [psia]$	$\beta_c = 0.001127$
$k = 8\ [mD]$	$T_f = 360\ [°C]\ y\ \Delta T = 10$	$\alpha_c = 5.615$

Pseudocódigo

Cálculo de la transmisibilidad

$$Tx_{i+\frac{1}{2}} = \left(\beta_c \frac{A_x\ k_x}{\mu B \Delta x}\right)_{i+\frac{1}{2}} = \frac{\beta_c}{\mu B} \frac{2A_{xi}\ A_{xi+1}\ k_{xi}\ \ k_{xi+1}}{A_{xi}\ \ k_{xi}\ \Delta x_{\ i+1} + \ A_{xi+1}\ k_{xi+1}\ \Delta x_{\ i}}$$

$$= \frac{0.001127}{3(1.05)} \frac{2[(900)(85)][900)(85)](8)(8)}{[(900 * 85)(8)(1000)] + [(900 * 85)(8)(1000)]} = 0.21896$$

$$Tx_{i-\frac{1}{2}} = \left(\beta_c \frac{A_x\ k_x}{\mu B \Delta x}\right)_{i-\frac{1}{2}} = \frac{\beta_c}{\mu B} \frac{2A_{xi}\ A_{xi-1}\ k_{xi}\ \ k_{xi-1}}{A_{xi}\ \ k_{xi}\ \Delta x_{\ i-1} + \ A_{xi-1}\ k_{xi-1}\ \Delta x_{\ i}}$$

$$= \frac{0.001127}{3(1.05)} \frac{2[(900)(85)][(0)](8)(0)}{[(900 * 85)(0)] + [(0)(0)(1000)]} = 0$$

Para $i = 1$ y $\Delta t = 100$ días

$$P_1^{n+1} = 6500 + \left(\frac{(5.615)(1.03)(100)}{[(900)(85)(1000)](0.22)(3.5x10^{-6})}\right)(0)$$
$$+ \left(\frac{(5.615)(1.03)(100)}{[(900)(85)(1000)](0.22)(3.5x10^{-6})}\right)$$

- Declaración de variables a utilizar

```
program difusibilidad
implicit none
real::B,B0,mu,Co,K,fi,q,Pi,dias,deltaT,Bc,Dy,Dz,Dx,P1,pter,constante,P1,P2,P3
real::q1,q2,q3,q4,q5,DelT,alfaC,Tf,Tximas12,Ximen12,num,nume,den,deno,P4,P5
integer::i
```

- Variables constantes empleadas para este ejercicio

```
B=1.05
B0=1.03
mu=3
Co=0.0000035
k=8
fi=0.22
q1=0
q2=0
q3=0
q5=0
q4=-125
Pi=6500
Tf=360.00
deltaT=10.00
Dx=1000
Dy=85
Dz=900
Bc=0.001127
alfaC=5.615
```

- Calculo de la transmisibilidad en las fronteras

```
!Calculo de la transmisibilidad en las fronteras
pter=Bc/(mu*B)
num=pter*(2*(Dy*Dz)*(Dy*Dz)*k*k)
den=((Dy*Dz*k*Dx)+(Dy*Dz*k*Dx))
Tximas12=num/den
nume=pter*(2*(Dy*Dz)*(0)*k*0)!48
deno=((Dy*Dz*k*0)+(0*0*1000))
Ximen12=0
```

- Cálculo de la presión en cada bloque mediante un ciclo conforme aumenta el tiempo

```
    !Calculo de la presion en cada bloque

  i=1

print*,"     P1          |       P2      |       P3       |       P4      |       P5       "
    do DelT=deltaT,Tf,10.00
  constante=(alfaC*B0*DelT)/((Dx*Dy*Dz)*fi*Co)
P1=Pi+(constante*q1)+constante*((Tximen12*Pi)-(Tximen12+Tximas12)*Pi+(Tximas12 *Pi))
P2=Pi+(constante*q2)+constante*((Tximas12*Pi)-(Tximas12+Tximas12)*Pi+(Tximas12 *Pi))
P3=Pi+(constante*q3)+constante*((Tximas12*Pi)-(Tximas12+Tximas12)*Pi+(Tximas12 *Pi))
P4=Pi+(constante*q4)+constante*((Tximas12*Pi)-(Tximas12+Tximas12)*Pi+(Tximas12 *Pi))
P5=Pi+(constante*q5)+constante*((Tximas12*Pi)-(Tximas12+Timen12)*Pi+(Timen12 *Pi))
```

- Se manda a imprimir los resultados en pantalla y se termina el ciclo y el programa

```
    print*,"",P1,"| ",P2,"| ",P3,"| ","| ",P4,"| ",P5,""
pause
    end do

end program difusibilidad
```

```
Calculo de la difusibilidaden un yacimiento
      _____________________________
     /     /     /     /     /     /|
    /     /     /     /     / ||  /     / |
   /     /     /     /     /  || /     /  |
  /     /     /     /     /     /     /   |
  -----------------------------     |
  |     |     |     |     |     |    /
  |  1  |  2  |  3  |  4  |  5  |   /
  |     |     |     |     |     |  /
  |     |     |     |     |     | /
  |     |     |     |     |     |/
  -----------------------------
 Calcular la distribucion de  la presion en el yacimiento
 a los 360 dias con un deltaT de 10 dias
     P1          |      P2      |      P3      |      P4      |      P5
     intervalo numero                1
     6500.00     |      6500.00     |      6500.00     | |      6377.27     |      6500.00
***Pause:
Enter system command or press ENTER key to restart:
     intervalo numero                2
     6500.00     |      6500.00     |      6500.00     | |      6254.54     |      6500.00
***Pause:
Enter system command or press ENTER key to restart:
     intervalo numero                3
     6500.00     |      6500.00     |      6500.00     | |      6131.81     |      6500.00
***Pause:
Enter system command or press ENTER key to restart:
     intervalo numero                4
     6500.00     |      6500.00     |      6500.00     | |      6009.09     |      6500.00
***Pause:
Enter system command or press ENTER key to restart:
     intervalo numero                5
     6500.00     |      6500.00     |      6500.00     | |      5886.36     |      6500.00
***Pause:
Enter system command or press ENTER key to restart:
     intervalo numero                6
     6500.00     |      6500.00     |      6500.00     | |      5763.63     |      6500.00
***Pause:
Enter system command or press ENTER key to restart:
     intervalo numero                7
     6500.00     |      6500.00     |      6500.00     | |      5640.90     |      6500.00
***Pause:
```

Fig. 7. 12
Salida de pantalla del ejercicio 5

Ejercicio propuesto:

Equilibrio Liquido vapor

Realizar algoritmo y Programa que dé solución al siguiente ejercicio, mediante el uso del método de Newton-Raphson obtener la mejor aproximación para calcular el equilibrio liquido vapor de la siguiente mezcla. *Los valores de K son obtenidos al considerar P=300 psi.

Recordar que la suma de la fase liquida más la fase gaseosa de una mezcla debe ser igual a uno $L + V = 1$

Componente	z_i	$V_{supuesto}$ k_1	0.5 x_i	x_i	x_i	x_i	y_i
H2S	0.0157	1.18					
Co2	0.0214	4.0					
N2	0.0037	9.3					
C1	0.4921	5.2					
C2	0.1038	1.5					
C3	0.0594	0.64					
i-C4	0.012	0.32					
n-C4	0.0283	0.25					
i-C5	0.0121	0.132					
n-C5	0.017	0.11					
C6	0.0246	0.05					
C7+	0.2099	0.023					
Suma=							

Partiendo de la función $f(v)$, que es la relación entre la fase liquida y gaseosa se obtiene su derivada para poder ser sustituidas en la Ecuación de Newton-Raphson

$$f(v) = \sum_{i=1}^{n} \frac{z_i(k_i - 1)}{1 + (k_i - 1)v} = 0$$

$$f'(v) = -\sum_{i=1}^{n} \frac{z_i(k_i - 1)^2}{[1 + (k_i - 1)v]^2} = 0$$

$$V_{k+1} = V_k - \frac{f(V_k)}{f'(V_k)}$$

Donde el Volumen de la fase vapor V_{k+1} servirá para obtener el volumen correspondiente de la fase liquida, y para obtener la fracción liquido (x_i) y vapor (y_i) de cada uno de los componentes de la mezcla mediante las siguientes ecuaciones:

$$x_i = \frac{z_i}{1 + (k_i - 1)v}$$

$$y_i = \frac{k_i z_i}{1 + (k_i - 1)v}$$

No olvidar que: $\sum_{i=1}^{n} \boldsymbol{x_i} = \sum_{i=1}^{n} \boldsymbol{y_i} = \mathbf{1}$

Componente	z_i	$V_{supuesto}$ $*k_1$	0.5 $f(v)$	$f'(v)$	V_{k+1}	x_i	y_i
H2S	0.0157	1.18					
Co2	0.0214	4					
N2	0.0037	9.3					
C1	0.4921	5.2					
C2	0.1038	1.5					
C3	0.0594	0.64					
i-C4	0.012	0.32					
n-C4	0.0283	0.25					
i-C5	0.0121	0.132					
n-C5	0.017	0.11					
C6	0.0246	0.05					
C7+	0.2099	0.023					
Suma=							

Bibliografía

Chapra, STEVEN C. y Canale, RAYMOND P. (2011). ***"Métodos numéricos para ingenieros".*** (6a ed.). México, México: Mc Graw Hill.

Nieves, ANTONIO. y Domínguez, FEDERICO C. (1998). ***"Métodos Numéricos Aplicados a la Ingeniería".*** (4a ed.). México, México: Compañía Editorial Continental, S.A. de C.V.

Enríquez, Adriana (2016). ***"Apuntes de Clase de Programación Avanzada"***. Ciudad de México: Facultad de Ingeniería-UNAM

Bravo, Aidee (2016). ***"Apuntes de Clase de Programación Avanzada"***. Ciudad de México: Facultad de Ingeniería.

Sabido, Juan Carlos (2018). ***"Apuntes de Clase de Programación Avanzada".*** Ciudad de México: Facultad de Ingeniería.

Ramírez, Andrés B. (2015). ***"Apuntes de Clase de Métodos Numéricos".*** Ciudad de México: Facultad de Ingeniería.