Contenido

[UNIDAD II 3](#_Toc18394305)

[II. ECUACIONES NO LINEALES 3](#_Toc18394306)

[2.1. Comentarios Generales 3](#_Toc18394307)

[2.1.1. Introducción 6](#_Toc18394308)

[2.2. Método de punto fijo o iteración funcional 7](#_Toc18394309)

[2.2.1. Programación en Matlab 12](#_Toc18394310)

[Introducción 12](#_Toc18394311)

[1.Estructura de un programa, 13](#_Toc18394312)

[2.Estilos del programa. 13](#_Toc18394313)

[3.Control de flujo. 14](#_Toc18394314)

[4.Condicionales 14](#_Toc18394315)

[5.Ciclos o Bucles (LOOP) 17](#_Toc18394316)

[6.Lectura y escritura iterativa de variables (input, disp, fprintf) 19](#_Toc18394317)

[Función Input. 19](#_Toc18394318)

[Función Disp 20](#_Toc18394319)

[Función fprintf 21](#_Toc18394320)

[**1.** **FORMATO** 22](#_Toc18394321)

[**2.** **Las Variables** 23](#_Toc18394322)

[Función Pause(n) 25](#_Toc18394323)

[2.Ficheros de comandos o scripts 25](#_Toc18394324)

[2.2.2. Criterios de convergencias 27](#_Toc18394325)

[2.3. Método de newton Raphson 31](#_Toc18394326)

[2.3.1. Algoritmo de seudocódigo 32](#_Toc18394327)

[2.3.2. Programa en Matlab para el método de newton 32](#_Toc18394328)

[2.3.3. Interpretación gráfica del método 33](#_Toc18394329)

[2.4. Método de la secante 35](#_Toc18394330)

[2.4.1. Interpretación gráfica 36](#_Toc18394331)

[2.4.2. Ejemplos 36](#_Toc18394332)

[2.5. Método de la bisección 38](#_Toc18394333)

[2.5.2. Programa en Matlab 39](#_Toc18394334)

[2.5.3. Ejemplos 40](#_Toc18394335)

[2.6. Practica nº 4 41](#_Toc18394336)

# UNIDAD II

# II. ECUACIONES NO LINEALES

# 2.1. Comentarios Generales

En esta oportunidad nuestras reflexiones estarán direccionado a obtener la suficiente información para abordar con éxito una amplia variedad de problemas de ingeniería, relacionados con las raíces de ecuaciones. En general, se dominarán las técnicas, se habrá aprendido a determinar su confiabilidad y se tendrá la capacidad de elegir el mejor método (o métodos) para cualquier problema particular.

En particular objetivaremos:

1. Comprender la interpretación gráfica de una raíz

2. Conocer la interpretación gráfica del método de la falsa posición y por qué, en general, es mejor que el método de bisección

3. Entender la diferencia entre los métodos cerrados y los métodos abiertos para la localización de las raíces

4. Entender los conceptos de convergencia y de divergencia; usar el método gráfico como de las dos curvas para tener una idea visual de los conceptos

5. Saber por qué los métodos cerrados siempre convergen, mientras que los métodos abiertos algunas veces pueden divergir

6. Observar que la convergencia en los métodos abiertos es más segura si el valor inicial está cercano a la raíz verdadera

7. Entender los conceptos de convergencia lineal y cuadrática, así como sus implicaciones en la eficiencia de los métodos de iteración de punto fijo y de Newton-Raphson

8. Conocer las diferencias fundamentales entre el método de la falsa posición y el método de la secante, y cómo se relacionan con la convergencia

9. Comprender los problemas que presentan raíces múltiples y las modificaciones que se pueden hacer para reducir dichos problemas

10. Saber cómo extender el método de Newton-Raphson de una sola ecuación no lineal con el propósito de resolver **sistemas de ecuaciones no lineales.**

Sobre las raíces de ecuaciones se debe de tener en consideración que una función cambia de signo en la vecindad de una raíz. A estas técnicas se les llama **métodos cerrados, o de intervalos, p**orque se necesita de dos valores iniciales para la raíz. Como su nombre lo indica, dichos valores iniciales deben “encerrar”, o estar a ambos lados de la raíz. Estos métodos emplean diferentes estrategias para reducir sistemáticamente el tamaño del intervalo y así converger a la respuesta correcta.

Inicialmente analizaremos los métodos gráficos para representar tanto a las funciones como a sus raíces. Además de la utilidad de los métodos gráficos para determinar valores iniciales, también son útiles para visualizar las propiedades de las funciones y el comportamiento de los diversos métodos numéricos.

Aunque las raíces de ecuaciones aparecen en el contexto de diversos problemas, son frecuentes en el área de diseño en ingeniería. Es de conocimiento que los principios fundamentales que se utilizan comúnmente en trabajos de diseño, las ecuaciones matemáticas o modelos provenientes de estos principios se utilizan para predecir los valores de variables dependientes en función de variables independientes y los valores de parámetros. Observe que en cada caso las variables dependientes representan el estado o desempeño del sistema; mientras que los parámetros representan sus propiedades o su composición.

**Un ejemplo** de tales modelos es la ecuación obtenida a partir de la segunda ley de Newton, para la velocidad del paracaidista:

, ………………………………….(2.1)

Donde la velocidad:

v = variable dependiente,

t = tiempo, variable independiente,

g = constante de gravitación en función de fuerza y el coeficiente de arrastre c

**m=** masa,

Si se conocen los parámetros, la ecuación (2.1) se utiliza para predecir la velocidad de un paracaidista como una función del tiempo. Estos cálculos se pueden llevar a cabo de manera directa, ya que **v** se expresa explícitamente como una función del tiempo. Es decir, queda despejada en el lado izquierdo del signo igual.

Suponga que se tiene que determinar el coeficiente de arrastre de un paracaidista con una masa dada, para alcanzar una velocidad determinada en un periodo preestablecido. Aunque la ecuación (2.1) ofrece una representación matemática de la interrelación entre las variables del modelo y los parámetros, no es posible obtener explícitamente el coeficiente de arrastre. No hay forma de reordenar la ecuación para despejar el parámetro **c.** En tales casos, se dice que c está en forma implícita.

Esto representa un verdadero dilema, ya que en muchos de los problemas de diseño en ingeniería hay que especificar las propiedades o la composición de un sistema (representado por sus parámetros) para asegurar que esté funcionando de la manera deseada (representado por las variables). Así, a menudo dichos problemas requieren la determinación de parámetros implícitos.

La solución del dilema es proporcionada por los métodos numéricos para raíces de ecuaciones.

Para resolver el problema con métodos numéricos es conveniente **re expresar** la ecuación (2.1), esto se logra restando la variable dependiente v de ambos lados de la ecuación,

, ……………………………………………….(2.2)

Por lo tanto, el valor de c que hace f(c) = 0 es la raíz de la ecuación. Este valor también representa el coeficiente de arrastre que resuelve el problema de diseño.

Más adelante analizaremos una gran variedad de métodos numéricos y gráficos para determinar raíces de relaciones tales como en la ecuación (2.2). Dichas técnicas se pueden aplicar a problemas de diseño en ingeniería con base en los principios fundamentales de muchos problemas que se encuentran de manera rutinaria en la práctica de la ingeniería, como tenemos:

Tabla N° 2.1 Principios fundamentales usados en los problemas de ingeniería.

|  |  |  |  |
| --- | --- | --- | --- |
| Principio fundamental | Variable dependiente | Variable independiente | Parámetros |
| Balance de calor | Temperatura | Tiempo y posición | Propiedades térmicas del material y geometría del sistema |
| Balance de masa | Concentración o cantidad de masa | Tiempo y posición | El comportamiento químico del material: coeficientes de transferencia de masa  y geometría del sistema |
| Balance de fuerzas | Magnitud y dirección de fuerzas | Tiempo y posición | Resistencia del material,  propiedades estructurales  y geometría del sistema |
| Balance de energía | Cambios en los,  estados de energía y  cinética y potencial del sistema | Tiempo y posición | Propiedades térmicas masa del material geometría  de un sistema |
| Leyes de Newton del  movimiento | Aceleración, velocidad y posición | Tiempo y posición | Masa del material, geometría del sistema y parámetros disipadores, tales como fricción y rozamiento |
| Leyes de Kirchhoff | Corriente y voltaje del  en circuitos eléctricos | Tiempo | Propiedades eléctricas sistema, tales como resistencia, capacitancia e  inductancia |

Fuente , Chapra

### 2.1.1. Introducción

En esta oportunidad trataremos sobre la solución de ecuaciones no lineales analizado de diferentes maneras desde la óptica analítica y su interpretación geométrica.

En el campo de la tecnología principalmente en la ingeniería nos encontramos generalmente con el siguiente problema determinar las raíces de la ecuación f(x) = 0. Donde f(x) puede ser una función de variable real x, como es un polinomio en x, o como una función trascendente es decir:





Para dar solución a estos problemas existen distintos algoritmos o métodos para encontrar las raíces de f(x) = 0, pero debemos tener en cuenta que ninguno es general, pues en otras palabras no existe un método que funcione con todas las ecuaciones perfectamente.

Pero sólo en un reducido caso será posible obtener las raíces exactas de f(x) = 0, es decir cuando se trata de f(x) factorizable, en tal sentido tenemos:



Donde r1, r2; r3;…. rn; son las raíces de la ecuación que pueden ser reales o complejas, es decir la solución al problema planteado.

En consecuencia, los métodos numéricos estándares para encontrar raíces se encuentran en dos áreas de problemas relacionados, pero fundamentalmente distintos:

1. La determinación de raíces reales de ecuaciones algebraicas y trascendentes. Dichas técnicas se diseñaron para determinar el valor de una sola raíz real basándose en un conocimiento previo de su posición aproximada.

2. La determinación de todas las raíces reales y complejas de polinomios. Estos métodos están diseñados especialmente para polinomios; determinan sistemáticamente todas las raíces del polinomio en lugar de sólo una raíz real dada una posición aproximada, y nuestras reflexiones estarán direccionados en analizar los dos enfoques.

**Objetivos de esta unidad.** Después de terminar la unidad uno se debe tener la suficiente información para abordar con éxito una amplia variedad de problemas de ingeniería, relacionados con las raíces de ecuaciones. En general, se dominarán las técnicas, se habrá aprendido a determinar su confiabilidad y se tendrá la capacidad de elegir el mejor método (o métodos) para cualquier problema particular. Además de estas metas generales, deberá haber asimilado los conceptos específicos de la tabla 2.1 para comprender mejor el material de la parte dos.

En el caso general se pueden obtener soluciones muy próximas a dichas raíces, esto utilizando métodos numéricos que serán visto en esta oportunidad iniciando con el Método de Punto fijo, que se conoce también como aproximaciones sucesivas de iteración funcional.

## 2.2. Método de punto fijo o iteración funcional

Supongamos que f(x) es una función continua en [a, b] y se tiene que la ecuación

f(x)=0, (2.3)

De la cual nos interesa determinar sus raíces, es decir un valor o valores de x = ri, i = 0, 1,2,…n, de las raíces reales, que al sustituirse se transforma en una igualdad dicha ecuación.

Primero: Lo que se debe de hacer es transformar la ecuación dada, en una ecuación equivalente usando el álgebra, es decir obtenemos

x = g(x), (2.4)

Entonces r es una raíz de ambas funciones de g(x) y f(x) sí solo sí, r es un punto fijo de g(x) en decir se cumple que r=g(r). Esto nos inducirá a recordar el teorema de punto fijo, que dice:

Dada una función, continúa y derivable en , con para todo y dado un , la sucesión , convergente, tal que , entonces r es una raíz de la ecuación x= g(x).

Veamos algunos ejemplos como es que se realiza este paso:

Supongamos que se tienen:

a) ; b) 

De estas dos ecuaciones podemos obtener las expresiones equivalentes usando para ello el álgebra:

Para el caso a:

,

,

.

Segundo caso b

,

*.*

Segundo: Una vez determinado con la búsqueda de la expresión algebraica equivalente, el paso que se debe de seguir es; tantear una raíz, la cual se puede realizar por observación directa de la ecuación inicial.

Para nuestros ejemplos tenemos que:

a) x = 2; b) x = 2, son valores cercanos a una raíz se denota el valor tanteado por X0

En general para determinar este valor inicial se recomienda bosquejar una gráfica de dicha ecuación, claro está si es posible.

Considerando que g(x) debe ser una función que cumpla con el teorema de punto fijo.

,

.

.

Tercero: Terminado el segundo paso se evalúa la relación encontrada en x0 denotándose el resultado de esta evaluación como x1, esto es

G(x0) = x1

Cuarto: El siguiente paso es comparar x1 con x0, resultando dos alternativas:

1. Primero Alternativa. Que x1 = x0

Esto quiere decir que el valor que se ha elegido como valor inicial es una raíz de f(x) y el problema termina.

2. Segunda Alternativa. Que x1 ≠ x0

Este, es el caso más frecuente e indica que x1  y x0  son diferentes de la raíz, puesto que si x = a no es una raíz entonces f(a) ≠ 0, por otro lado si evaluamos g(a) ≠a. Entonces el resultado se le denota con x2 es decir, x2 = g(x1), esto se repite de manera iterativa obteniendo el siguiente esquema:

Valor inicial x0 f(x0)

Iteración 1 x1 = g(x0) f(x1)

Iteración 2 x2 = g(x1) f(x2)

Iteración 3 x3 = g(x2) f(x3)

:

:

Iteración k xk = g (xk-1) f (xk)

Iteración k+1 xk+1 = g (xk) f (xk+1)

Obs.

Debemos resaltar que la sucesión x0, x1, x2, x3;…xk… se va acercando al valor de la raíz r1, de manera que xk se encuentra más cerca de r que xk-1 o se van alejando de la raíz.

X0 X1 X2 X3  Xn-1 Xn = r

,

esta sucesión debe de tender a cero, en este caso se dice que el proceso converge a r1, y debe de continuar hasta un di menor de un error

Veamos para nuestros ejemplos dados:

 Para el caso (a) el proceso diverge y para el caso (b) converge, es decir.

|  |  |  |
| --- | --- | --- |
| Iteraciones k | Valor de xk | g(xk)=2x2-5 |
| 0 | 2 | 3 |
| 1 | 3 | 13 |
| 2 | 13 | 333 |
| 3 | 333 | 221773 |

Caso: (a)

|  |  |  |
| --- | --- | --- |
| Iteraciones k | Valor de xk |  |
| 0 | 2 | 1.87083 |
| 1 | 1.87083 | 1.85349 |
| 2 | 1.85349 | 1.85115 |
| 3 | 1.85115 | 1.85083 |

Caso: (b)

Observamos que en el caso (a) el valor diverge, pero en el caso (b) converge al valor 1.85078 que es una raíz de la función inicial.

**Para el segundo ejemplo se tiene muy parecido,**



Para el caso (a) el proceso diverge y para el caso (b) converge es decir.

|  |  |  |
| --- | --- | --- |
| Iteraciones k | Valor de xk | G(xk)=3x2- 8 |
| 0 | 2 | 4 |
| 1 | 4 | 40 |
| 2 | 40 | 4792 |
| 3 | 4792 |  |

Caso: (a)

|  |  |  |
| --- | --- | --- |
| Iteraciones k | Valor de xk |  |
| 0 | 2 | 1.1180 |
| 1 | 1.1180 | 1.7434 |
| 2 | 1.7434 | 1.8022 |
| 3 | 1.8022 | 1.8076 |

Caso: (b)

Observamos que en el caso (a) el valor diverge, pero en el caso (b) converge al valor 1.8081 que es una raíz de .

Con la finalidad de saber cuándo la sucesión x0, x1,x2,x3;…xk…converge o diverge de la raíz buscada podemos calcular en f(x0), f(x1), f(x3), …, f(xk) si dicha sucesión tiende a cero entonces el proceso anterior converge a la raíz deseada, el mencionado proceso se continuará hasta que  donde el valor de épsilon uno es un valor pequeño que indica la exactitud o acercamiento de xk a la raíz r en este caso se toma xk como una raíz aproximada.

En caso contrario si f(x0), f(x1), f(x3),…, f(xk) no tiende a cero entonces la sucesión x0, x1,x2,x3;…xk  diverge de la raíz r y el proceso deberá de detenerse y ensayar con otra función g(x).

Veamos para nuestro ejemplo (b) considerando la convergencia de f(x)

Para el caso (a) el proceso diverge y para el caso (b) converge es decir.

|  |  |  |  |
| --- | --- | --- | --- |
| Iteraciones k | Valor de xk | G(xk)=3x2- 8 |  |
| 0 | 2 | 4 | 2 |
| 1 | 4 | 40 | 36 |
| 2 | 40 | 4792 | 4852 |
| 3 | 4792 |  |  |

Caso: (a)

|  |  |  |  |
| --- | --- | --- | --- |
| Iteraciones k | Valor de xk |  |  |
| 0 | 2 | 1.1180 | 2 |
| 1 | 1.1180 | 1.7434 | -5.368 |
| 2 | 1.7434 | 1.8022 | -0.6250 |
| 3 | 1.8022 | 1.8076 | -0.058 |

Caso: (b)

Se observa que en el caso (a) f(x) diverge mientras que en el caso (b) converge a cero.

### 2.2.1. Programación en Matlab

### Introducción

Matlab puede realizar una diversidad de operaciones con matrices y otros, pero sin embargo, se desea tener el control sobre algunos aspectos importantes de manera iterativa para ello Matlab contiene los archivoc scrip.

Por ahora veremos>

1. Estructura de un programa,
2. Estilos del programa
3. Control de flujo
4. Condicionales (if, if-else; elseif; switch, case, otherwise)
5. Ciclos, (for, while, sentencias break, continue, try catch, entd)

**Primero:** Usando los diferentes comandos citado Matlab puede realizar operaciones más explicadas por ejemplo con matrices, la construcción de programas en Matlab es simples, recordemos que en Matlab se trabaja con comandos scripts, para ello Matlab cuenta con in incorporado para ello solo se debe dar clic fai; new, scripts, y luego escribir las instrucciones tal como se pondría en el Windows, y guadar el documento con la extensión punto m.

Por ejemplo pongamos algunas instrucciones básicas, y lo guardamos, como pro, y se observa que la extensión del archivo es punto.

x=20;

y=30;

estando guardado vamos a la ventana Windows o ventana de trabajo y Matlab reconoces de manera automática, En esta ventana escribimos el nombre (miprog) las ordenes almacenadas serán ejecutadas.

A este fenómeno se le denomina flujo, y a los comando sentencias, se pueden repetir secuencialmente pero podemos controlar su flujo y para ello se usara algunas condicionales.

### 1.Estructura de un programa,

La estructura general de un programa básico en Matlab,

**Primero**, van los comentarios, donde se puede colocar el titulo del programa y comentar sobre ello es importante ponerlos porque permite a partir de la ventana de comandos %....

**Segundo**, Datos del programa que se deben de suministrarse, desde el techado o un fichero de datos,

**Tercero,** el algoritmo que es el desarrollo de un procedimiento con la finalidad de determinar algo en función de los datos de entrada.

**Cuarto,** la salida de los datos, obtenidos como solución del algoritmo que se deben de ofrecer al usuario en pantalla o mediante un fichero de datos.

### 2.Estilos del programa.

Un programa estructurado final debe ser un buen programa, debemos considerar quien es mas eficiente y para esto debemos ir probando su ejecución pero con la práctica se descubre esto, la construcción de programas se vuelven una rutina donde el orden facilita las cosas, veamos algunas recomendaciones,

1. El programa debe resultar de fácil lectura, para permitir su facilidad de repaso, las variables deben de tener un auto explicación,

Po ejemplo

|  |  |
| --- | --- |
| Programa 1 | Programa 2 |
| N=10; | Num\_ptos=10; |
| X=rand(N,10); | Prob=rand(1,Num-ptos); |
| Y=zeros(1,N); | Prob-acum zeros(1,Num-ptos); |
| for i=1;N, | For índice=1: Num-ptos, |
| Y(i)=sum x((1:i)); | Prob-acum(indice)=sum(prob(1:indice)); |
| end | end |

1. Introducir comentarios en el programa, los comentarios son líneas que no son ejecutados, mas bien se escriben detalles de los programas, un programa con comentarios es mucho mas fácil de entender.
2. Definir las variables al principio, pues es posible definir al inicio los vectores matrices con sus propiedades, para decir es más rápido definir al inicio el vector cero, que ir añadiendo componentes al momento en que se calcula.
3. Es importante introducir ciclos y condicionales, es costumbre introducir una sangría, entre las líneas contenidas, pues son de significante ayuda en los ciclos y condiciones anidadas. Matlab cuenta con ello para realizarlo de manera automática solo se tiene que sobreas todo el código luego clic derecho, i la opción Smart indent.

### 3.Control de flujo.

Es llamado así al proceso de toma de decisiones, antes de escribir el programa sobre todo cuando es un poco largo es recomendable trazar un esquema al que se le llama diagrama de flujo. cuando el diagrama esta claro solo resta por traducir el diagrama del español al lenguaje de Matlab. Matlab cuenta con cuatro instrucciones para el control de flujo las condicionales: if-else; switch y los ciclos: for , While. Debemos decir que todas estas instrucciones terminan con la palabra clave end.

**2. Condicionales**, Matlab tiene un lenguaje dispone de sentencias para realizar condicionales. Las cuales son los que permiten realizar una u otra operación, según se cumpla o no una determinada condición. A seguir veremos como crear una condicional:

### 4.Condicionales

En los diagramas las condiciones se encierran en un rombo, mientras que las sentencias o bloque de códigos, se encierran en rectángulos,

2.1. if, if-else, else-if

La sentencia IF, en su forma mas simple se escribe de la siguiente manera>\:

condición

False

true

sentencia

If condición

Sentencia

End.

Siendo su diagrama,

Veamos un programa de uso

% este programa hace que la variable j sea igual a -1 si la varaible a es

% menor que b,

a=2;

b=3;

if a<b

j=-1;

end

**If-else**

condición

False

true

Bloque 1

Bloque 2

**If** condición

Bloque 1

**Else**

Bloque 2

End.

Siendo su diagrama,

Veamos un programa de uso

% este programa hace que la variable j sea igual a -1 si la varaible a es

% menor que b,caso contrario que sea igual 1

a=2;

b=3;

if (a<b)

j=-1;

else

j=1;

end

Guardar el programa como ejemplo 2

**BIFURCACION MULTIPLE**

En esta oportunidad se puede concatenar cuantas formas se desee y tiene dos posibles maneras.

Bloque 2

condición

False

true

Bloque 1

Bloque 3

False

true

|  |  |
| --- | --- |
| Forma 1 | Forma 2 |
| IF, condición 1  Bloque 1 | If condición 1  Bloque 1 |
| Elseif condición 2  Bloque 2 | Elseif condición 2  Bloque 2 |
| Elseif condición 3  Bloque 3 | Elseif condición 3  Bloque 3 |
| : | : |
| Elseif condición N  Bloque N | Elseif condición N  Bloque N |
| End, | Else  Bloque N+1 |
|  | End. |

**Ejemplo**

% un programa para hace que la variable j sea igual a -1 si a<b

% 0 si a==b

% 1 si a>b

a=4;

b=3;

if a<b

j=-1;

elseif a==b

j=0;

elseif a>b

j=1;

end

También es posible comprobar múltiples expresiones booleanas,

La condición i f puede ser de tipo matricial, es decir. A==B donde A y B son matriciales del mismo tamaño, de igual manera para matrices diferentes considerando rigurosamente sus características caso contrario no se ejecuta la sentencia if.

Para este caso en particular Matlab contiene el comando isequal(A,B) la rspuesta es 1 o cero,

Se usa el comando strcmp(‘yes’, ‘not’)

**Sentencias** **switch, case, otherwise**

Condición 1

true

False

Bloque 1

Bloque 2

true

False

Bloque 4

False

Bloque 3

Condición 2

Condición 3

true

switch, var

Case opcion1

Bloque 1

Case opcion 2

Bloque 2

Case opcion 3

Bloque 3

Otherwise

Bloque 4

end

Otras posibles alternativas

switch, var

Case opcion1

Bloque 1

Case {opcion 2ª,opción 2b, opción 2c,…}

Bloque 2

…..

Case opcion N

Bloque N

Otherwise (% opcion por defecto)

Bloque N+1

End

Ejemplo

a=1;b=2

operacion=2;

switch operacion

case 1

r=a+b; % si laopracion es 1 sume a y b

case 2

r=a\*b; % si la operacion es 2 multiplicar a y b

otherwise

r=a-b; % de lo contrario reste a y b

end

r % para que muestre el resultado, fi

### 5.Ciclos o Bucles (LOOP)

Controlan un conjunto de instrucciones que deben de repetirse cierto número de veces, mientras se cumple cierta instrucción que debe ser claramente especificada.

**Partes básicas de un ciclo**

**Primero,**

La decisión donde se evalúa la condición en caso de ser cierta, se ejecuta el cuerpo del ciclo.

**Segundo,**

Cuerpo del ciclo son las instrucciones que se ejecutaran y se repetidamente, un numero determinado de veces, siempre que la decision sea verdadera.

**Tercero,**

Salida del ciclo que es la condición que indica cuando se terminan las iteraciones.

En Matlab se usa la sentencia end mientras que en C/ c++ / Java hacen lo mismo pero entre llaves {…}

Control al inicio del ciclo (for, while)

Control al final del ciclo (do.. while)

En Matlab no existe este tipo de condiciones

**1.1. Ciclo FOR**

Esta sentencia repite un conjunto de sentencias un numero predeterminado de veces, esta sentencia no tiene la forma general de java/c++ ,

Su estructura es ,

For<variable>=<expresion>

Sentencias

End.

Ejemplo

For i=1:n o Tambien for i=vectorvalores

Sentencias sentencias

End. End.

Donde vector Valores es un vector con los distintos valores que tomara la variable i.

Ejemplos

%primero un ciclo simple que se ejecutara 4 veces

% cambiando cada vez una variable j en el ciclo

for j=1:4

j

end

Cuando Matlab lee la sentencia for construye el vector [1:4]

Siendo que j toma cada valor dentro del vector en orden.

En cada vuelta el enunciado for actualizara el valor de j y repitiera los enunciados que se encuentren dentro del ciclo .

For

Caso mas general para la variable del ciclo, (valor inicial, incremento: valor final) como observamos el ciclo se ejecuta por primera ves i=3, y luego i se reduce cada 0.2. hasta llegar a menor que 1 luego el ciclo termina.

For i=3:-2:1

Sentencia

end

**ejemplo**

% el siguiente ejemplo i se inicia desde 1 y se va incrementando de 2 en

% dos hasta llegar a 10 y en cada iteraci[on elevara al cuadrado cada

% variable

for i=1:2:10

r=i^2

end

se debe de guardar y después correrlo.

### 6.Lectura y escritura iterativa de variables (input, disp, fprintf)

### Función Input.

Esta función permite escribir un mensaje en la línea de comandos de Matlab, y recuperar como valor de retorno un valor numérico o el resultado de una expresión tecleado por el usuario desde el teclado.

**Ejemplos.1.**

**>> n= input('teclee el número de ecuaciones:')**

**teclee el número de ecuaciones:6**

**n =**

**6**

**Quedará guardado en la variable n.**

Ejemplo. 2.

**>> nombre=input('Como te llamas?','s')**

**Como te llamas? Santiago %se podrá introducir una cadena de caracteres**

**nombre =**

**Santiago**

**Ejemplo.3.**

**ejemplo='cuantos años tienes?'**

**ejemplo =**

**cuantos años tienes?**

**>> edad=input(ejemplo)**

**cuantos años tienes?23**

**edad =**

**23**

**Ejemplo, 4. Ingresar matrices o vectores**

**>> matriz=input('ingrese usted. la matriz')**

**ingrese usted. la matriz magic(4)**

**matriz =**

**16 2 3 13**

**5 11 10 8**

**9 7 6 12**

**4 14 15 1**

### Función Disp

Esta función permite imprimir en pantalla un mensaje o texto de un vector o matriz pero sin imprimir su nombre.

**Ejemplo 1.**

>> disp('el programa a termiado')

el programa a termiado

>> A=rand(4)

A =

0.8147 0.6324 0.9575 0.9572

0.9058 0.0975 0.9649 0.4854

0.1270 0.2785 0.1576 0.8003

0.9134 0.5469 0.9706 0.1419

>> disp(A)

0.8147 0.6324 0.9575 0.9572

0.9058 0.0975 0.9649 0.4854

0.1270 0.2785 0.1576 0.8003

0.9134 0.5469 0.9706 0.1419

**Ejemplo2.**

> x=-28.9

x =

-28.9000

>> disp(x)

-28.9000

>> disp(pi^3)

31.0063

>> disp('alguna cadena')

alguna cadena.

**Ejemplo 3**

Como se imprime una tabla de valores es decir abscisa u ordenada

x=1:5 % vector fila

y=10:15 %vector fila

res(:,1)=x' %vector columna 1 de la matriz

res(:,2)=y' % vector columna 2 de la matriz

%esto es equivalente a poner res=[x',y']

disp('Tabla funcion')

disp('abscisa ordenada')

disp(res)

se corre.

>> ejemplores20

x =

1 2 3 4 5

y =

10 11 12 13 14 15

res =

1 10

2 10

3 10

4 10

5 10

>>

n=input('numero natural que se desea ver si es primo:')% en esta sentencia se ingresa el numero para ver si es primo

i=2;

primo=1;

while i<=sqrt(n)

if rem(n,i)==0 % el resto de dividir n por i

primo=0;

break

end

i=i+1;

end

if primo

disp('el numero dado es primo.')

else

disp('el numero dado no es primo.')

end

**se guarda y se corre en ventana de trabajo**

**>> ejemplodisp21**

**numero natural que se desea ver si es primo:35**

**n =**

**35**

**el numero dado no es primo.**

**>> ejemplodisp21**

**numero natural que se desea ver si es primo:59**

**n =**

**59**

**el numero dado es primo.**

**>>**

### Función fprintf

Esta función muestra un resultado o mensaje permitiendo mezclar textos y valores numéricos de la variable, y también se puede ajustar al formato de los números.

Esta función tiene una diversidad de variables y su sintaxis es basado en el fprintf del lenguaje C, su forma general puede ser:

**fprintf (formato) o fprintf (formato, variable )**

Donde:

formato es una cadena de texto, que controla la apariencia de salida.

Variables es una lista de variables separadas por comas que van a ser mostradas en pantalla de acuerdo con las especificaciones descritas en el formato.

1. **FORMATO**

Esto nos indica los campos de salida especificados en cadena.

Las cadenas pueden considerar operadores adicionales,los cuales aparecen en el siguiente orden,

Incluye espacios para claridad,

% 3 $ 0 – 12 . 5 b u

Carácter de conversión

Sub tipo

Precisión

Ancho

Bandera

Indicador

**El indicador**, nos indica el orden como debe ser procesado las entradas, se usa la sintaxis n$, donde n representa la posición del valor en la lista de entrada. Por ejemplo:

Fprintf(‘%3$s %2$s %1$s %2$s’,’A’,’B’,’C’)

La salida en Matlab> C B A B

>> fprintf('%3$s,%2$s,%1$s,%3$s', 'A','B','C')

C,B,A,C>>

>>

**Banderas.**

**‘-‘,** justifica a la izquierda, ejemplo %-5.2f

**‘+’** Imprime el carácter + para los valores positivos, ejemplo %+5.2f

**‘’** Rellena el ancho de campo con espacios en blanco antes del valor, ejemplo %5.2f

‘**0’** Reellena el ancho de campo con ceros ejemplo %05.2f

**‘#’** Modifica las conversiones numéricas seleccionadas;

Para decir %0, %x; o %X, imprime los prefijos 0, 0x, o 0x respectivamente.

Para decir %f, %e; o %E, imprime puntos decimales e incluso cuando es cero.

Para decir %g, o %G; no elimina los ceros, a la derecha ni el punto decimal.

**Ancho de campo.**

Es el mínimo número de caracteres que se imprimen, puede ser un numero o un asterisco (\*), se puede referir a un argumento en la lista de entrada, por ejemplo la lista de entrada (‘%12d’, intmax) es equivalente a (‘%\*d’,12, intmax).

**La precisión**

**Para %f, %e, %E.** Indica el número de dígitos a la derecha del punto decimal, ejemplo, ‘%6.4f’ imprime pi con ‘3.1416’.

**Para %g, %G.** Indica el numero significativos. Ejemplo ‘%6.4g’ imprime pi con ‘3.141’.

Puede ser un numero o un asterisco (\*) para referirse a un argumento en la lista. Por ejemplo (‘%6.4f’, pi) es equivalente a (‘%\*.\*f’,6.4, pi)

**Caracteres de control, incluidos**,

%%, carácter de porcentaje,

\\ , Backslash

\a , alarma auditiva,

\b , Backspace,

\f, form feed,

]n, nueva linea,

\r , retorna de carro,

\t , tab horizontal,

\v tab vertical

\xN, carácter cuyo código ASCII es el numero hexadecimal, N

\N, carácter cuyo código ASCII es el numero octal , N

1. **Las Variables**

Son arráis numéricos o de caracteres, especificado como un escalar, un vector, una matriz, o un array multidimensional.

n=5;

x=linspace(-pi,pi,n);

y=x/1e6;

z=x\*1e6;

fprintf('\n un simple uso de fprintf\n');

for j=1:length(x)

fprintf('%d %f 5f %f \n',j,x(j),y(j),z(j));

end

guardar como ejemplo22, y corres en la Ventana de trabajo.

>> ejemplo22

un simple uso de fprintf

1 -3.141593 5f -0.000003 -3.141593e+06

2 -1.570796 5f -0.000002 -1.570796e+06

3 0.000000 5f 0.000000 0

4 1.570796 5f 0.000002 1.570796e+06

5 3.141593 5f 0.000003

3.141593e+06 >>

**Ejemplo 23**

n=5;

x=linspace(-pi,pi,n);

y=x/1e6;

z=x\*1e6;

fprintf('\n la misma precision para números de punto fijo\n');

for j=1:length(x)

fprintf('%d %.2f %.2f %.2f\n',j,x(j),y(j),z(j));

end

guardar como ejemplo23 y correr en la ventana de trabajo.

Como observamos la precisión se indica después del punto

**>> ejemplo23**

la misma precisión para números de punto fijo

1 -3.14 -0.00 -3141592.65

2 -1.57 -0.00 -1570796.33

3 0.00 0.00 0.00

4 1.57 0.00 1570796.33

5 3.14 0.00 3141592.65

>>

Veamos algunas modificaciones con la finalidad de que se vea mejor la impresión

Ahora se va especificar el ancho de las columnas

n=5;

x=linspace(-pi,pi,n);

y=x/1e6;

z=x\*1e6;

fprintf('\n ancho de campo y control de la precisión y notación científica\n');

for j=1:length(x)

fprintf('%d %8.2f %13.2e %13.2e\n',j,x(j),y(j),z(j));

end

Guardamos y corremos y observamos la diferencia.

**>> ejemplo24**

**ancho de campo y control de la precision y notacion cientifica**

**1 -3.14 -3.14e-06 -3.14e+06**

**2 -1.57 -1.57e-06 -1.57e+06**

**3 0.00 0.00e+00 0.00e+00**

**4 1.57 1.57e-06 1.57e+06**

**5 3.14 3.14e-06 3.14e+06**

Es posible agregar encabezados para cada columna

n=5;

x=linspace(-pi,pi,n);

y=x/1e6;

z=x\*1e6;

fprintf('\n Retoques finales, encabezados\n');

fprintf('\nj x(j) y(j) z(j)\n');

for j=1:length(x)

fprintf('%d %8.2f %13.2e %13.2e\n',j,x(j),y(j),z(j));

end

Guardamos y corremos

>> ejemplo25

Retoques finales, encabezados

j x(j) y(j) z(j)

1 -3.14 -3.14e-06 -3.14e+06

2 -1.57 -1.57e-06 -1.57e+06

3 0.00 0.00e+00 0.00e+00

4 1.57 1.57e-06 1.57e+06

5 3.14 3.14e-06 3.14e+06

>>

### Función Pause(n)

Esta función hace una pausa en la ejecución del programa por n segundos antes de continuar, donde n puede ser cualquier numero real.

Ejemplo26

secs=0.05; %en segundos

for i=0:1:100

fprintf('%3d \\',i);

pause(secs); fprintf('\b|');

pause(secs); fprintf('\b/');

pause(secs); fprintf('\b-');

pause(secs); fprintf('\x');

end

fprintf('FIN:\n')

## 2.Ficheros de comandos o scripts

Son ficheros con un nombre tal como file1.m que contiene una sucesión de comandos análogos al que se teclearía en el uso iterativo del programa.

Los comandos se ejecutan de manera sucesiva cuando se teclea en nombre del fichero que contiene si la extensión es decir cuando se tecle file1.

Cuando se ejecuta desde la línea de comandos las variables creadas file1, pertenecen al espacio de trabajo base de Matlab. Por el contrario, si se ejecuta desde una función, las variables que crea pertenecen al espacio de trabajo de la función.

Por ejemplo, vamos a crear un script en un archivo llamado triarea.m, que calcule el área de un triángulo.

1. Funciones

Son ficheros dende la primera línea del fichero se le llama **mane.m** que define la función que tiene la siguiente estructura:

**Function** [Lista de valores de retorno]=**name**(lista de argumentos)

Donde NAME: es el nombre de la función, entre corchetes y separados por comas van los valores de retorno, siempre que exista mas de uno, Y entre paréntesis van los argumentos de entra también separados por comas

Veamos el ejemplo visto con anterioridad,

function area = areatrape(a,b,h)

%este programa calculara el area de un trapecio

% donde a,b son las bases y h la altura

area=(0.5)\*(a+b)\*h; %calcula el area

end

function x=puntofijo(g)

%

% MÉTODO DE PUNTO FIJO

%

%Primero: determinar una ecuación equivalente g(x) a la ENL dada

%Segundo: Tantear una raíz que será el valor inicial xo

%Tercero: valorar g(Xo) determinando x1

%Cuarto: Comparar los valores de Xo y X1

% Si son iguales i.e. Xo = X1, terminó el problema

% Si son diferentes Xo , X1

%Quinto: valorar g(x) pero ahora en X1 el proceso se repite N veces

help puntofijo;

x0 = input('ingrese el valor de xo=');

N =input('ingrese el valor máximo de iteraciones N=');

e=input('ingrese el error');

i= 1;

x(1)=x0;

while i<=N

x(i+1)= feval(g,x(i)); % este comando busca a la función g y lo evalua en el punto xi

if abs(x(i+1)-x(i))>=e

i=i+1;

else

disp('termino');

x(i+1)

end

end

(Guardar en la ventana de trabajo como archivo function)

ARCHIVO DE LA FUNCIÓN A DETERMINAR SU RAIZ

function y=g1(x)

y=cos(x)/3;

(Guardar en V T como archivo function g)

ARCHIVO DE LA FUNCIÓN ORIGINAL

Esto con la finalidad de observar como la función converge a cero.

Function y=f1(x)

y=cos(x)-3\*x;

(Guardar en V T como archivo function f)

### 2.2.2. Criterios de convergencias

Con la finalidad de analizar la convergencia de las sucesiones formadas estudiaremos otro criterio para nuestro proceso iterativo del método anterior visto basado en que g(r) = r

No olvidar que r es la raíz donde la sucesión x0, x1, x2, x3;…xk…debe converger esto quiere decir que los valores consecutivos de esta sucesión se van acercando cada vez más a dicha raíz conforme se realice el proceso iterativo esquemáticamente se tendrá:

\* \* \* \* … \* \* … \*……… \*………..\*r

X0; X1; X2; X3; ….. X10; X20 …. …….. XK …………r X

Observemos que un modo práctico de saber si los valores consecutivos arriba escritos se acercan a la raíz r, se debe de ir calculando las distancias entre ellos d1,d2,d3;…dk es decir  y si esta nueva sucesión se acerca a cero se puede pensar entonces que nuestro método en análisis de punto fijo converge a una raíz x = r, y se debe seguir hasta que dk< ε en este caso tomar xk+1 como la raíz buscada.

En el caso que la sucesión d1, d2, d3;…dk no converge para un número grande de iteraciones (al cual podemos llamarlo MAXT), entonces la sucesión x0, x1, x2, x3;…xk diverge de la raíz x = r, se tiene que parar el proceso y luego modificar la función g(x). Como se ha visto en los ejemplos anteriores en el caso (a) se tenía que parar y luego se determinó la nueva relación g(x) para la cual si convergía.

Se debe de tener presente que este tipo de convergencia se usa más en el Análisis

Numérico que el de, f(x0), f(x1), f(x3),…, f(xk)..., siendo así además menos segura.

**CRITERIO DE CONVERGEN DE LA PRIMERA DERIVADA **

Este criterio de la primera derivada tiene como base analítica el teorema del punto medio aplicado a la función g(x) en el intervalo Xk-1, Xk  es decir se tiene:



En donde ck  es un punto cualquiera del intervalo en análisis es decir ck pertenece a (xk, xk-1), si tenemos en consideración el valor que toma g(xk) =xk+1 y g(xk-1) = xk; esto quiere decir que nuestra relación se transforma en:

,

Luego podemos tomar valor absoluto a esta nueva expresión obteniendo lo siguiente

 Si tratamos de ponerlo en su forma más expresiva se tiene:





:

Pero ahora podemos suponer que la derivada de g(x) está acotada en toda la región de la sucesión x0, x1, x2, x3;…xk  en otras palabras se tendrá que:

 Esto para algún valor de M entonces tenemos que









:

Observe que si sustituimos la primera en la segunda y está en la tercera y así sucesivamente se tiene que  y esto podemos generalizar así:



Observemos que el método del punto fijo puede converger por una diversidad de criterios, pero es evidente que si M es menor que uno es decir M<1, dicha metodología convergirá puesto que Mk, convergirá a cero al tender k a un número grande.

Consecuentemente ya tenemos un método muy práctico y consistirá en obtener una función de f(x) y luego determinar el valor absoluto de su derivada y evaluarlo en x0 y si este valor es menor que uno entonces se debe de tomar dicha relación funcional.

**INTERPRETACIÓN GEOMÉTRICA**

x = r x3 x2 x1 xo

Y = g(x)

Y = x

X1 = g(x)

X2 = g(x)

Interpretación Geométrica de , Convergencia monótona

X2 = g(x)

Y = g(x)

Y = x

X1 = g(x)

x3 x=rx2 x1 xo

**Convergencia Oscilatoria**

Veamos un ejemplo:

Calcular una raíz de la función  considerando como valor inicial 1

Primero: Consideremos dos formas para

 , y 

Segundo: determinemos sus derivadas de cada funcional considerada,

 y 

Tercero: Valorando el valor las derivadas en 1 y luego tomando su valor absoluto, tenemos:

 y  entonces en este caso se debe tomar la primera relación puesto que es menor que uno; el segundo caso es mucho mayor de uno, veamos que sucede:

|  |  |  |  |
| --- | --- | --- | --- |
| Nº de iteraciones k | xk |  |  |
| 0 | 1.00000 | 0.00000 | 0.47337 |
| 1 | 1.53846 | 0.53846 | 0.42572 |
| 2 | 1.29502 | 0.24344 | 0.45100 |
| 3 | 1.40183 | 0.10681 | 0.44047 |
| 4 | 1.35421 | 0.04762 | 0.44529 |
| 5 | 1.37009 | 0.02101 | 0.44317 |
| 6 | 1.36593 | 0.00937 | 0.44412 |
| 7 | 1.37009 | 0.00416 | 0.44370 |
| 8 | 1.36824 | 0.00185 | 0.44389 |
| 9 | 1.36960 | 0.00082 | 0.44386 |

Debemos observar que el valor absoluto de la derivada valorado en el punto respectivo siempre se mantiene menor que la unidad.

Una vez que llegamos a las condiciones requeridas del problema, es decir que la diferencia de las distancias entre los puntos de la sucesión x0, x1, x2, x3;…xk  sea menor que 0.0001 se termina el proceso y tomamos como raíz a r = 1.36906.

Si hubiésemos tomado las otras funciones equivalentes no obtendríamos la convergencia de las distancias y más aún el valor de la derivada en cada punto sería mayor que uno en muchos casos.

## 2.3. Método de newton Raphson

El método en análisis es un procedimiento que puede ser aplicado en una diversidad de oportunidades, principalmente cuando se trata de funciones que tengan raíces reales.

Supongamos que estamos interesados en determinar las raíces de f(x) numéricamente siendo r una raíz y considerando que x es una aproximación a este valor, consideremos que f’’ existe y es continua luego tendremos por el Teorema de Taylor lo siguiente:





En donde h = r – x, que ocurre si h se aproxima a cero entonces x se aproxima a la raíz r. Si ignoramos el tercer término podemos determinar el valor para h. es decir tenemos que:

 

Si aproximamos a la raíz r entonces se deberá encontrarse más cerca de r. De esta manera el Método de Newton comienza con una estimación x0  para r y a partir de la cual se define usando inducción la sucesión de aproximación que se representa de la siguiente manera:



### 2.3.1. Algoritmo de seudocódigo

Input x0,M,δ, ε

y←f(x0)

output 0, x0, y

if /y/ < ε

then stop

for k = 1,2,3,….M

x←x-y / f’(x)

y ←f(x1)

Output k, x1, y

if |x1-x0| <δ or |y| <ε,

Then stop

x0←x1

End

Es importante destacar que se debe de tener un programa para Calcular el valor de f(x) y de su derivada

### 2.3.2. Programa en Matlab para el método de newton

function newton(f,df)

% help newton;

x0 = input('ingrese el valor de xo=');

N =input('ingrese el valor máximo de iteraciones N=');

e=input('ingrese el error');

x(1)=x0;

%fprintf('\n iteración i valor de x(i) Valor de g(i) valor absoluto de f(xi) \n')

for i=1:N

x(i+1)=x(i)-feval(f,x(i))/feval(df,x(i));

if abs(x(i+1)-x(i))<=e

disp('la raíz aproximada es ');

x(i+1)

break

end

end

x=x'

LA FUNCION

function y=f2(x)

y=x^3+2\*x^2+10\*x-20;

SU DERIVADA

function y=df2(x)

y=3\*x^2+4\*x+10;

### 2.3.3. Interpretación gráfica del método

Antes de continuar con el análisis del método consideremos la idea de bosquejar una interpretación gráfica. En estas circunstancias podemos decir que el método de Newton consiste en la linealización de la función esto quiere decir que la función f(x) será sustituida por una función lineal y esto ocurre cuando usamos la serie de Taylor del siguiente modo



Luego si linealizamos tenemos

l(x) = f ( c) + f ’ (c) (x-c), en este caso se observa que l(x) es una buena aproximación de la función f(x) en c , de hecho tenemos que la función l(x) tiene el mismo valor que f (c) y la misma pendiente es decir l’ ( c ) = f’ ( c ) esto en el punto c . En otras palabras graficar el método de Newton se debe de considerar la tangente a f(x) en un punto cercano de r

f(x)

r xn+1 x0

f(x0)

Línea tangente a la función f(x)

f(x) =f(xn)+f’(xn)(x-xn)

ESQUEMA DEL MÉTODO DE NEWTON

Esto se puede realizar de manera esquemática:

1.- Representar gráficamente la función f(x) la cual corta al eje x en r que es la raíz de f(x)

2.- Representar x0 como el valor inicial de la sucesión de puntos en el eje de las X.

3.- Trazar la tangente a la función f(x) en x0 y ubicar el punto de corte con el eje X y la tangente y denotarlo con x1  el cual será la nueva aproximación a la raíz r.

4.- El proceso se repite hasta que sea necesario es decir cumpla con las exigencias en otras palabras falta que | f(xk) | < δ y | xk+1-xk | < ε se cumpla una o ambas.

5. Si en el caso de no cumplirse en un número máximo de iteraciones se sugiere reiniciar de nuevo.

Ejemplo

Determinar una raíz de  considerando x0 =1, y el criterio de convergencia y con un error de ε = 10-3

Solución

1. Primero determinamos la derivada del polinomio: 
2. Aplicar la sucesión iterativa 
3. 
4. x1 = 1.41176
5. X2 = 1.36934
6. X3 = 1.36881
7. X4 = 1.36881

Cuadro que representa los diferentes cálculos para determinar la aproximación de una raíz usando el Método de Newton.

|  |  |  |  |
| --- | --- | --- | --- |
| Nº de iteraciones k | xk |  |  |
| 0 | 1.00000 | 0.00000 | 0.24221 |
| 1 | 1.41176 | 0.41176 | 0.02446 |
| 2 | 1.36934 | 0.04243 | 0.00031 |
| 3 | 1.36881 | 0.00053 | 1.09x10-6 |
| 4 | 1.36881 | 0.00000 | 1.2714x10-6 |

Debemos destacar que con este método solo se necesitaban tres iteraciones para alcanzar la aproximación necesaria y además se tiene una mayor aproximación.

## 2.4. Método de la secante

Debemos recordar que una de las debilidades del método de Newton es que utiliza la derivada de la función y se trata de encontrar un cero de esta. Es en este sentido que surge una diversidad de métodos y uno de ellos es el Método de la Secante que analizaremos en adelante.

Supongamos que estamos interesados en solucionar la debilidad de la metodología de Newton, empecemos por reemplazar la derivada f’ (x) en la secuencia que origina el método de Newton por un cociente de diferencias es decir



recordemos que esta relación tiene como fundamento la definición de la derivada de f(x) en términos de un límite, realicemos tal sustitución enunciada y así tendremos.



Observemos que si calculamos xk+1 entonces se requiere conocer xk y xk-1 esto quiere decir que se deben de dar en la problemática estos dos valores.

Así también se observa que para determinar el valor de xk sólo se requiere un cálculo de f(x) .

### 2.4.1. Interpretación gráfica

L a interpretación gráfica es similar que la interpretación grafica del método de Newton solo que en este caso se debe de considerar la línea tangente como una línea secante

f(x)

r Xk+1 Xk Xk-1

f(Xk-1)

Línea secante a la función f(x)

f(Xk)

### 2.4.2. Ejemplos

**Ejemplo**

Usar el método de la secante para encontrar una raíz real de la ecuación polinomial , considere x0 = 0; x1 = 1, usar como criterio de convergencia la secuencia de distancias de aproximación a la raíz.

Solución

a) Aplicamos la secuencia que determina la metodología:



 Entonces x2 = 1.53846

b) X3 = 1.35031

c) X4 = 1.36792

d) X5 = 1.36881

a seguir presentamos el cuadro que se obtiene al realizar dicha metodología en el cual observaremos que se trata de un método rápido en convergencia casi tan igual que el Método de Newton pero mucho más rápido que el Método de Punto Fijo

|  |  |  |
| --- | --- | --- |
| Nº de iteraciones k | xk |  |
| 0 | 0.00000 | 0.00000 |
| 1 | 1.00000 | 1.00000 |
| 2 | 1.53856 | 0.53846 |
| 3 | 1.35031 | 0.18815 |
| 4 | 1.36792 | 0.01761 |
| 5 | 1.36881 | 0.00090 |

Ejemplo

Usar el método de la secante para encontrar una raíz real de la ecuación polinomial , considere x0 = 7; x1 = 8, usar como criterio de convergencia la secuencia de distancias de aproximación a la raíz.

**Solución**

a) Aplicamos la secuencia que determina la metodología:



 Entonces x2 = 7.05895

b) X3 = 7.11764

c) X4 = 7.11289

d) X5 = 7.11306

e) X6 = 7.11306

A seguir presentamos el cuadro que se obtiene al realizar dicha metodología en el cual observaremos que se trata de un método rápido en convergencia casi tan igual que el Método de Newton pero mucho más rápido que el Método de Punto Fijo

|  |  |  |
| --- | --- | --- |
| Nº k | xk |  |
| 0 | 7.00000 |  |
| 1 | 8.00000 | 1.00000 |
| 2 | 7.05895 | 0.94105 |
| 3 | 7.11764 | 0.05859 |
| 4 | 7.11289 | 0.00475 |
| 5 | 7.11306 | 0.00017 |
| 6 | 7.11306 | 0.00000 |

### 2.5. Método de la bisección

Supongamos que se tiene una función continua en el intervalo [a,b] y de tal manera que f(a)\*f(b) < 0 esto quiere decir que f(x) tiene un cero en el intervalo abierto (a,b). Por la razón que el producto del valor de la función en a y b es negativo es decir cambia de signo en el intervalo [a,b], lo que afirma es una consecuencia del teorema del valor medio.

Pues el método en análisis explota el hecho anterior para su fundamento, pues dicho método determina c = (a+b)/2 y averigua si f(a) f(c) <0 si esto resulta siendo cierto entonces f(x) tiene una raíz en el intervalo [a,c]. En seguida tomamos el valor de c como b y realizamos el mismo análisis anterior.

Si ocurriera que f(a) f(c) >0 entonces f(c) f(b) < 0 en este caso redefinimos a c =a En ambos caso a sucedido que se a determinado un nuevo intervalo que contiene una raíz de la función y el proceso puede repetirse.

Si f(a)\* f(c) = 0, o f(c)\* f(b) = 0; entonces f(c) = 0 y con esto se ha determinado una raíz del polinomio, pero vale aclarar que este caso no sucede en general puesto que los redondeos en una computadora difícil es cero. Por esta razón es que para concluir se debe realizar con una tolerancia de 10-3.

Este método también se le conoce con el nombre de método de la bipartición, pero debemos destacar este método es el más sólido y seguro que los otros métodos para encontrar una raíz en un intervalo.

**2.5.1 INTERPRETACIÓN GRÁFICA**

**caso a) f(a)\*f(c)<0¸ Caso b) f\*(c)f(b)<0**

a  c b

f(a)

f(c) f(b)

a  c b

f(a)

f(b)

### 2.5.2. Programa en Matlab

**PROGRAMA DE LA BISECCION**

function biseccion(f)

% help newton;

a = input('ingrese el valor de a=');

b = input('ingrese el valor de b=');

N =input('ingrese el valor máximo de iteraciones N=');

e=input('ingresde el error');

fa=feval(f,a);

fb=feval(f,b);

if fa\*fb<0

for i=1:N

c=(a+b)/2;

fc=feval(f,c);

if fa\*fc<0

b=c;

else

a=c;

end

end

if abs(b-a)< e

disp('mi raiz aproximada es');

c;

end

else

disp('en el intervalo ingresado no existe raiz');

end

c

FUNCIÓN

function y=f2(x)

y=x^3+2\*x^2+10\*x-20;

### 2.5.3. Ejemplos

Determinar una raíz real del polinomio , considerando un error de 10-3

**Solución**

1. Determinamos los valores de a y b evaluando la función en algunos puntos donde sea relativamente fácil de evaluar por ejemplo:

f(0)= -20; f(1) = -7 ; f(-1) = -29 ; f(2) = 16.

Considerando el razonamiento del método observamos que en el intervalo [1,2] existe una raíz de la funcional. En este caso nuestros valores de a = 1, b = 2

b) Determinamos el número de iteraciones M considerando la siguiente relación



En donde a es la longitud del intervalo, en nuestro caso

c) Realizamos la primera iteración determinando

c = (2+1) / 2 = 1.5; 

Observamos que f(c) = f(1.5) = 2.88 >0 distinto signo que f(a) = f(1) = -7 luego reemplazamos el valor de b por el valor de c es decir se tiene el siguiente intervalo (1, 1.5), entonces

a = 1; f(1) = -7

b = 1.5; f(1.5) = 2.88

d) Segunda Iteración:



Observamos que f(c) = f(1.25) = -2.42 < 0 igual signo que f(a) = f(1) = -7 luego reemplazamos el valor de a por el valor de c es decir se tiene el siguiente intervalo (1.25, 1.5), entonces

a = 1.25; f(1.25) = -2.42

b = 1.5; f(1.5) = 2.88

A continuación seguir presentamos el siguiente cuadro que contiene las 7 iteraciones deseadas en donde denotaremos a = Xa: b = Xb y C = XM

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| k | a | B | C |  |  |
| 0 | 1.00000 | 2.00000 |  |  |  |
| 1 | 1.00000 | 2.00000 | 1.50000 | 0.25000 | 2.87500 |
| 2 | 1.00000 | 1.50000 | 1.25000 | 0.12500 | 2.42188 |
| 3 | 1.25000 | 1.50000 | 1.25000 | 0.06250 | 2.42188 |
| 4 | 1.25000 | 1.37500 | 1.37500 | 0.03125 | 0.13086 |
| 5 | 1,31250 | 1.37500 | 1,31250 | 0.01563 | 0.52481 |
| 6 | 1.34375 | 1.37500 | 1.34375 | 0.00781 | 0.19846 |
| 7 | 1.35938 | 1.37500 | 1.35938 | 0.00395 | 0.03417 |
| 8 | 1.36719 | 1.37500 | 1.36719 | 0.00195 | 0.04825 |
| 9 | 1.36719 | 1.37109 | 1.37109 | 0.00098 | 0.00702 |
| 10 | 1.36719 | 1.36914 | 1.36914 | 0.00049 | 0.01358 |
| 11 | 1.36826 | 1.36914 | 1.36865 | 0.00025 | 0.00329 |
| 12 | 1.36865 | 1.36914 | 1.36890 | 0.00013 | 0.00186 |
| 13 | 1.36865 | 1.36890 | 1.36877 | 0.00006 | 0.00071 |

## 2.6. Practica nº 4

**I.- Resolver las siguientes ecuaciones usando los diferentes métodos analizados**

**1.**

**2) **

**3) **

**4) **

**5) **

**6) **

**7) **

**8) **

**9) **

**10) **

**11) **

**12) **

**13) **

**14) **

**15) ;**

**16) **

**17) ,**

**18) **

**19) ;**

**20) **

**21) ;**

**22) **

**23) ;**

**24) **

**25);**

**26)**

**27) ;**

**28.)**

**29) **

**30) **