1 RAICES DE ECUACIONES

DEFINICIÓN:

La raíz de una ecuación es aquel valor de la variable independiente que hace que el resultado de la ecuación sea cero o por lo menos se acerque a cero con una cierto grado de aproximación deseado (error máximo permitido).

OBJETIVOS:

Al terminar esta unidad de estudio, el estudiante estará en capacidad de:

- ✓ Aprender el concepto de raíz de una ecuación
- ✓ Comprender las distintas técnicas usadas para hallar raíces de ecuaciones
- ✓ Evaluar la confiabilidad de cada método
- ✓ Estar en capacidad de elegir el mejor método para aplicarlo en la solución de problemas de ingeniería, relacionados con la búsqueda de raíces de una ecuación.

1.1 MÉTODOS BASADOS EN INTERVALOS

Su característica fundamental es que se elige un intervalo [a, b] dentro del cual se encuentre la raíz buscada. No hay una regla a seguir para la selección de este intervalo, sin embargo, se debe cumplir que en los extremos del intervalo la función cambie de signo lo cual que equivale a que: ***f(a)*f(b) < 0.*** Una primera aproximación a la solución se logra al elaborar un modelo gráfico de la ecuación y a partir de él, por simple inspección, seleccionar el intervalo más adecuado.

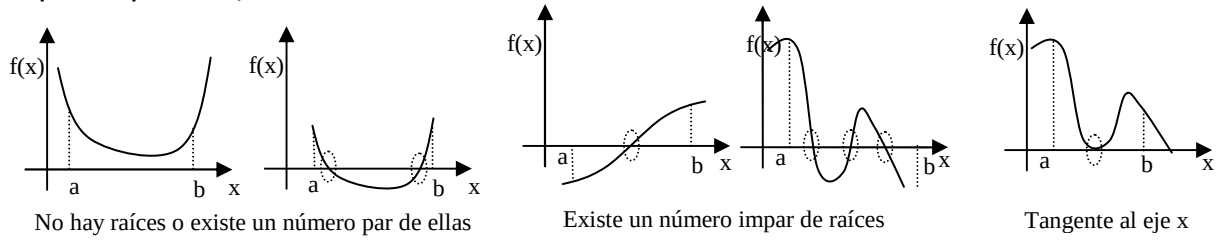


Figura 1. Raíces en diferentes intervalos.

Si al evaluar la función, en los extremo del intervalo elegido, presenta igual signo puede no existir raíces o existir un número impar de ellas. Si la función cambia de signo esto nos indica que al menos hay una raíz en dicho intervalo. Los casos que requieren análisis especial se presentan cuando la función tiene puntos tangentes al eje x o cuando tiene discontinuidades.

1.1.1 **Criterio de convergencia.** Se considera que se ha encontrado una respuesta satisfactoria cuando el valor hallado para la variable faltante (*x*) cumple con alguno de los siguientes criterios:

$$|f(x)| < Tol \; ; \; \textbf{ó} \; \left| \frac{|X_r - X_{r-1}|}{X_r} \right| \leq Tol$$

Siendo X_r, y X_{r-1} los valores de las dos últimas iteraciones y ***Tol*** es el nivel máximo de error aceptado que se puede definir ya sea con base en el número de cifras significativas (NCS) $\boldsymbol{Tol = 5x10^{-(NCS+1)}}$ o en el número de cifras decimales (ND) que se desea obtener, $\boldsymbol{Tol = 5x10^{-(ND+1)}}$. Otro criterio para terminar el proceso es que se exceda el número máximo de iteraciones propuesto, en cuyo caso lo más probable es que la solución no esté convergiendo hacia un valor determinado (cada vez se aleja más del valor estimado), por lo tanto se debe probar con otra estrategia de solución o revisar muy bien lo cálculos matemáticos realizados para ver si no se están cometiendo errores en el proceso.

1.1.2 **Método gráfico.** Lo esencial en este método es poder construir un modelo gráfico de la ecuación y luego por inspección estimar una aproximación a la raíz.

El mayor inconveniente de éste método es su poca precisión y exactitud. Sin embargo, hoy día se cuenta con excelentes herramientas de software para realizar rápidamente gráficas con un alto grado de realismo. El primer problema a resolver es ¿que intervalo usar para construir la gráfica? No hay regla que nos diga como hacerlo, por eso lo mejor es probar con varios intervalos hasta encontrar el más adecuado, no obstante es importante considerar las características particulares del problema que vamos a resolver, ya que eso nos dará una idea del rango de posibles soluciones, por ejemplo si queremos hallar una magnitud física como velocidad, distancia, masa, etc., sabemos que no tiene sentido probar con valores negativos, por lo tanto podemos graficar rangos a partir de cero.

Como ejemplo consideremos las funciones: $f(x) = seno(3x) + Cos(x)$ y $f(x) = e^x - x - 2$. El conjunto de posibles valores que puede tomar ***x*** están en el intervalo (-∞, ∞), pero sabemos además que la primera función tiene período 2π, lo cual nos ayuda a limitar e intervalo a graficar.

Usando MatLab podemos graficar la primera función el intervalo [0, π] y la segunda en el intervalo [-4, 4], para obtener un modelo gráfico como el de la figura 2.

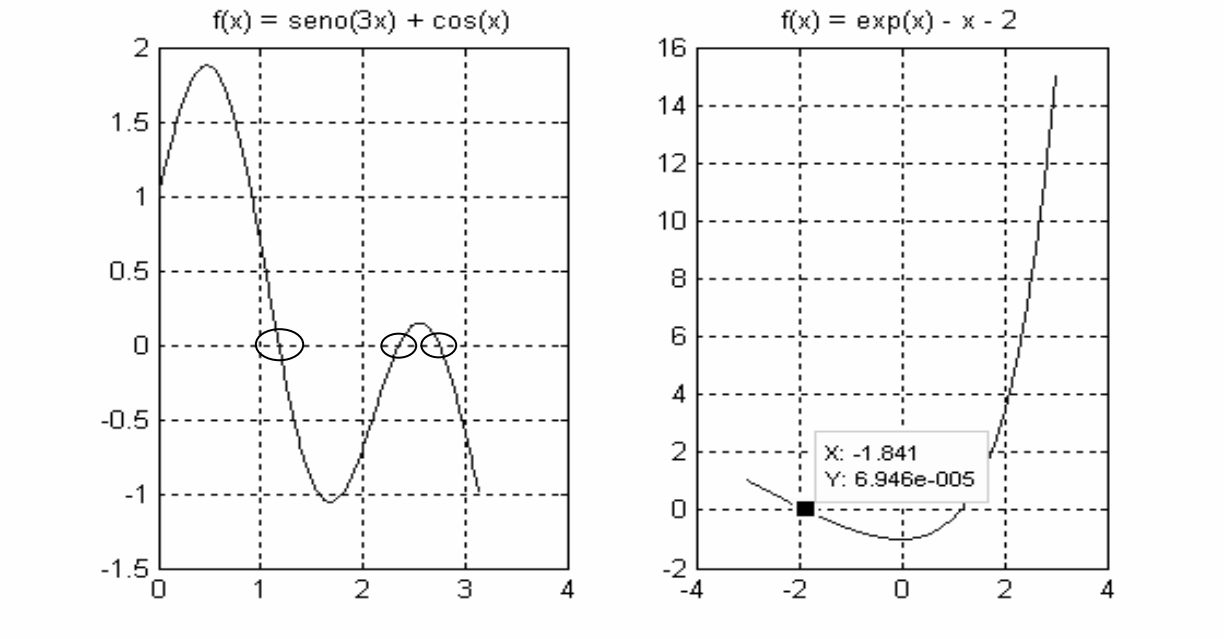


Figura 2. Método gráfico para hallar raíces.

La gráfica ha sido elaborada usando MatLab, en ella se puede apreciar que para la primera función, hay una raíz en el intervalo [1, 2] y posiblemente 2 raíces en el intervalo [2, 3]. En la segunda gráfica se ve que las raíces se encuentran en el intervalo [-2, 2], y se muestra el valor aproximado de una de ellas, con cuatro cifras significativas. El siguiente archivo escrito en MatLab permite elaborar la gráfica anterior:

```
Cls; clear all;
x=0:0.01:pi; %Creamos un vector con valores entre 0 y pi.
y=sin(3.*x)+cos(x);
subplot(1,2,1) % Dividimos la pantalla en dos secciones y graficamos en la primera sección
plot(x,y)
title('f(x) = seno(3x) + cos(x)')
grid
x=-3:0.01:3; % Definimos el rango de valores para graficar la segunda función
y=exp(x) - x - 2;
subplot(1,2,2)
plot(x,y)
title ('f(x) = exp(x) - x - 2')
grid %Trazamos la cuadrícula
```

Aunque la aproximación a la raíz no es muy exacta, el método es útil para estimar el intervalo inicial para los métodos numéricos como bisección y falsa posición.

Ejemplo 1: un objeto de *60 Kg* de masa, después de *10 s* en caída libre adquiere una velocidad de *40 m/s*. ¿Cual será el coeficiente de rozamiento del aire en ese momento?

Por leyes de la física se sabe que para este tipo de movimiento la velocidad está dada por: $v(t) = \frac{gm}{c}(1 - e^{-\frac{c}{m}t})$ donde ***m*** es la masa del objeto que cae, ***c*** es el coeficiente de rozamiento, ***t*** es el tiempo y ***g*** es la fuerza de gravedad (9.81 m/s² aprox.).

Solución: Si reemplazamos los valores conocidos, el único valor que queda por averiguar es ***c***, no es sencillo despejarlo de la ecuación, por lo tanto recurrimos a los métodos numéricos para hallar una aproximación satisfactoria. Reemplazando los valores conocidos obtenemos:

$$40 = \frac{9.81 * 60}{c}(1 - e^{-\frac{c}{60}10})$$

Esta expresión es una ecuación matemática en función de ***c*** que podemos expresar como:

$$f(c) = \frac{9.81 * 60}{c}(1 - e^{-\frac{c}{60}10}) - 40 .$$

Podríamos resolver por ensayo y error dando valores a ***c*** e ir haciendo las respectivas operaciones hasta que se logre obtener un valor cercano a cero en la ecuación y esa sería la solución buscada, método que probablemente nos tome mucho tiempo y cálculos matemáticos. Si elaboramos la gráfica con MatLab, obtenemos una respuesta aproximada de ***13.02*** con un error cercano al 5%. (Ver figura 3).

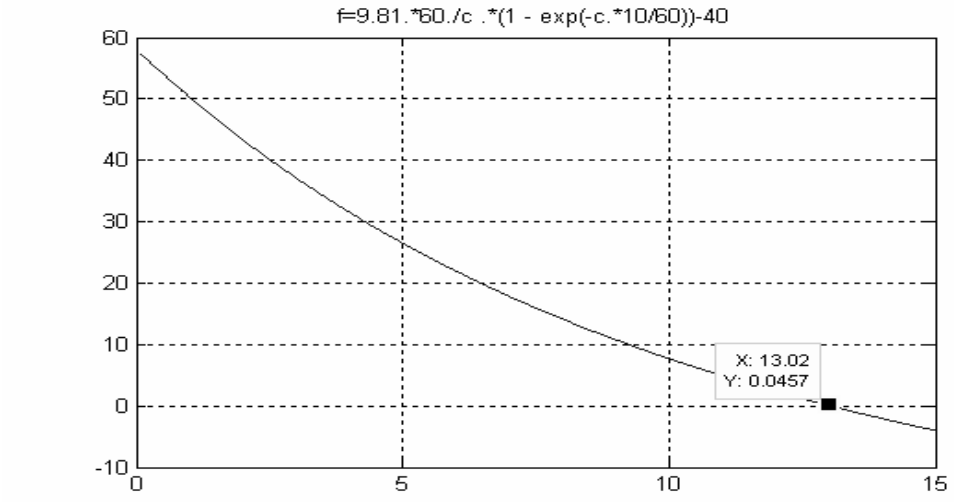


Figura 3. Raíz aproximada de f(c) usando el método gráfico.

El archivo .m de MatLab que permite obtener esta gráfica es:

```
clc
clear all
c=0.1:.01:15;
```

```
f=9.81.*60./c .*(1 - exp(-c.*10/60))-40;
plot(c,f)
grid
title ('f=9.81.*60./c .*(1 - exp(-c.*10/60))-40')
```

No siempre la gráfica realizada con el MatLab, es la representación real de la función, esto depende le incremento que usemos para representar los valores de la variable independiente, de la resolución de la pantalla que usemos para visualizar la gráfica, del intervalo a considerar, etc. En MatLab, podemos delimitar la región que va a ser usada para trazar la gráfica y esto nos permite visualizar mejor la imagen. Veamos el siguiente ejemplo:

y = cos(x)*cosh(x)+1 en el intervalo [0,20].

El siguiente guión escrito en MatLab, permite obtener la anterior figura:

```
x=0:0.01:20;
y = cos(x).*cosh(x)+1;
plot(x,y)
grid
subplot(1,2,1)
plot(x,y)
grid
subplot(1,2,2)
plot(x,y)
AXIS([0,20,-20,20])
GRID
subplot(1,2,1)
TITLE('Y = cos(x)*cosh(x)+1')
subplot(1,2,2)
TITLE('Y = cos(x)*cosh(x)+1, usando el comando AXIS([0,20,-20,20])')
```

La gráfica de la izquierda difiere de la de la derecha, esto se debe a que el comando AXIS permite ajustar los valores mínimo y máximo de las coordenadas x e y para dibujar la figura. El comando es: ***axis([xmin, xmax, ymin, ymax])***. Si un valor se sale de esos límites, se cortan, lo cual de una imagen mas cercana a la realidad. Si se desea una figura de forma cuadrada se pude usar el comando *axis('square')*.

El comando *AXIS* también puede usarse para omitir los ejes de coordenadas y la escala, tecleando el comando *axis(`off')*. Para activar nuevamente los ejes se utiliza el comando *axis('on').*

El comando ***axis*** se usa después de la orden ***plot*** y se puede modificar el área de visualización tantas veces como se desee invocando nuevamente dicha orden, sin necesidad de invocar la orden plot nuevamente.

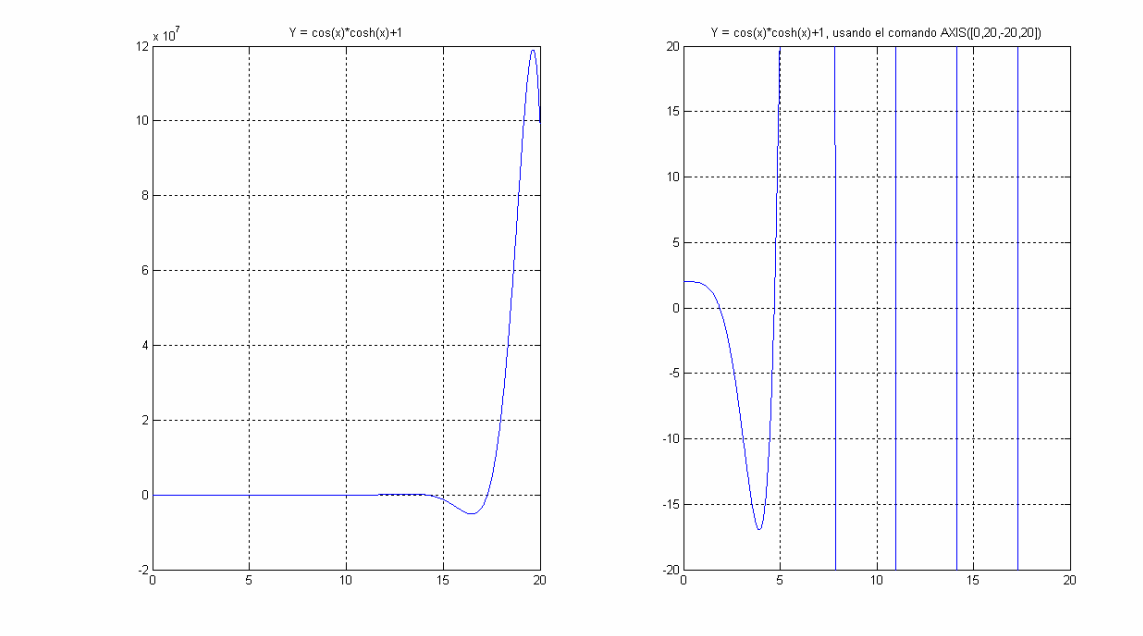


Figura 4. Uso del comando AXIS

1.1.3 **Método de bisección (Corte binario o Bolzano).** En este algoritmo se escoge un intervalo [a, b] en el cual la función cambie de signo en cuyo caso se cumple que: *f(a)*f(b) < 0* y se toma como aproximación a la raíz el punto medio de dicho intervalo. Este valor se puede calcular por medio de la expresión: $Xr = a + \frac{b-a}{2}$. O su equivalente: $Xr = \frac{a+b}{2}$. El proceso se termina cuando se cumpla el criterio de convergencia antes mencionado en el numeral 1.1.1.

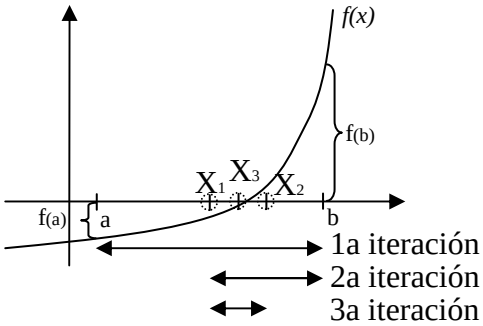


Figura 4. Método de bisección.

Algoritmo:

1. Defina el error máximo aceptado (*Tol*) y el número máximo de iteraciones (*MaxIter*), la aproximación anterior X_{r-1}=0 y el contador de iteraciones (*r*)

2. *r = r + 1*.
3. Elija un intervalo [a, b] donde la función cambie de signo (*f(a)*f(b)) < 0*).
4. La aproximación a la raíz se determina como: ***x_r = a + (b - a)/2***.
5. Verifique en que mitad del intervalo queda la nueva aproximación, evaluando:
   a. Si f(a)*f(x_r) < 0 la raíz esta en este primer subintervalo. Haga ***b = x_r***. Vaya al paso 3.
   b. Si f(a)*f(x_r) > 0 la raíz está en el segundo subintervalo derecho. Haga ***a = x_r***. (Se cambia la cota inferior). Vaya al paso 3.
6. Verificar si se cumple el criterio de convergencia:

   Si $|f(x)| < Tol$ ó $\left|\frac{|X_r - X_{r-1}|}{X_r}\right| \le Tol$. Se ha halló la raíz y su valor es ***x_r*** ; el proceso se detiene.

   Si ***r > maxIter***: se excedió el número de iteraciones, por lo tanto la solución no está convergiendo. Se debe parar el proceso.
7. $X_{r-1} = X_r$. Guardamos la anterior aproximación a la raíz.
8. Volver al paso 3.

El siguiente programa codificado en MatLab, permite calcular raíces de ecuaciones por el método de bisección y corresponde al diagrama de flujo de la figura 5. En la parte final del programa se cambia la ecuación que se desea procesar. El programa se ejecuta directamente desde la línea de comandos de MatLab tecleando ***biseccion*** o abriendo el archivo y ejecutando el comando ***run***, en el menú ***debug***, o presionando la tecla ***F5*** o haciendo clic en el icono ***Run*** en la barra de herramientas.

```
% Programa para calcular raices de ecuaciones basado en el metodo de biseccion
%autor: Ing. Edgar Romero Rodríguez
%Docente Fundación Universitaria de San Gil, UISANGIL
%San Gil, Santander – Colombia.  Enero de 2004

function biseccion
clc;
clear;
a=input('Valor de la cota inferior: ');
b=input('Valor de la cota superior: ');
TOL=input('Tolerancia de error  : ');
max_iter=input('Numero de iteracciones: ');
iter = 0;
fprintf('\nMETODO DE BISECCION...\nRESULTADOS\n')
fprintf('Iter      a          Xr         b         f(a)      f(xr)     Error\n');
while 1 %Inicio un cliclo infinito del cual salgo al hallar la raíz o exceder max_iter
  xr=(b+a)/2; %Aproximo el valor de la posible raíz como el punto medio del intervalo.
  fa = funcion(a); %Evaluamos la función en la cota inferior
  fb = funcion(b); %evaluamos la función en la cota superior
  fxr= funcion(xr); %Evaluamos la función en la posible raíz
  error = abs((b - a)/a);
```

```
  % Impresion de resultados en cada iteración
  fprintf('%4.0f,%10.6f,%10.6f,%10.6f,%10.6f,%10.6f,%12.6f\n',iter,a,xr,b,fa,fxr,error)
  if error <= TOL %Si supero el error aceptado detengo el proceso
      fprintf('Proceso concluido exitosamente con el nivel de error <= %10.6f\n\n',TOL)
      break;
  end
  if abs(fxr)<=TOL %Se hallo la aproximación a la raíz y salgo del ciclo infinito
     fprintf('\nProceso concluido exitosamente con el nivel de error <= %12.6e\n\n',TOL)
     break;
  end
  if fa*fxr < 0
      b=xr;  %La raiz esta en el primer intervalo
  else
      a=xr; %La raiz esta en el segundo intervalo
  end
  if (iter > max_iter)  %Verifico si se excede el numero de iteracciones
      fprintf('\nNumero de iteracciones excedido...\n\n')
      break;
  end
  iter=iter+1; %Incremento el numero de iteracciones
end
fprintf('Raiz aproximada: %12.6f',xr);
fprintf('  Iteraciones  : %5.0f\n',iter);
% Modelo gráfico
x=CotaInf:0.01:CotaSup;
f=funcion(x);
plot(x,f)
gris

function y=funcion(x)  %A continuación debo colocar la función a evaluar
y=cos(x);
% fin del programa
```

Ejemplo 2: Si evaluamos *f(x) = cos(x)* en el intervalo *[1, 2],* con cuatro cifras significativas obtenemos:

```
Valor de la cota inferior : 1
Valor de la cota superior: 2
Tolerancia de error       : 5e-5
Numero de iteraciones   : 25

METODO DE BISECCION...
RESULTADOS
 Iter     a            Xr           b          f(a)         f(xr)        Error
   0,  1.000000,  1.500000,  2.000000,  0.540302,  0.070737,   0.500000
   1,  1.500000,  1.750000,  2.000000,  0.070737, -0.178246,   0.250000
   2,  1.500000,  1.625000,  1.750000,  0.070737, -0.054177,   0.125000
```

```
  3,  1.500000,  1.562500,  1.625000,  0.070737,  0.008296,    0.062500
  4,  1.562500,  1.593750,  1.625000,  0.008296, -0.022952,    0.031250
  5,  1.562500,  1.578125,  1.593750,  0.008296, -0.007329,    0.015625
  6,  1.562500,  1.570313,  1.578125,  0.008296,  0.000484,    0.007813
  7,  1.570313,  1.574219,  1.578125,  0.000484, -0.003422,    0.003906
  8,  1.570313,  1.572266,  1.574219,  0.000484, -0.001469,    0.001953
  9,  1.570313,  1.571289,  1.572266,  0.000484, -0.000493,    0.000977
 10,  1.570313,  1.570801,  1.571289,  0.000484, -0.000004,    0.000488

Proceso concluido exitosamente con el nivel de error <= 5.000000e-005
Raíz aproximada:  1.570801  Iteraciones:  10
```

Ejemplo 3: Si evaluamos la función planteada en el ejemplo 1 tenemos:

```
Valor de la cota inferior: 10
Valor de la cota superior: 15
Tolerancia de error  : 5e-3
Numero de iteracciones: 25

METODO DE BISECCION...
RESULTADOS
Iter     a          Xr         b         f(a)      f(xr)     Error
  0, 10.000000, 12.500000, 15.000000,  7.742782,  1.224863,    2.500000
  1, 12.500000, 13.750000, 15.000000,  1.224863, -1.520449,    1.250000
  2, 12.500000, 13.125000, 13.750000,  1.224863, -0.185835,    0.625000
  3, 12.500000, 12.812500, 13.125000,  1.224863,  0.509676,    0.312500
  4, 12.812500, 12.968750, 13.125000,  0.509676,  0.159503,    0.156250
  5, 12.968750, 13.046875, 13.125000,  0.159503, -0.013765,    0.078125
  6, 12.968750, 13.007813, 13.046875,  0.159503,  0.072718,    0.039063
  7, 13.007813, 13.027344, 13.046875,  0.072718,  0.029439,    0.019531
  8, 13.027344, 13.037109, 13.046875,  0.029439,  0.007827,    0.009766
  9, 13.037109, 13.041992, 13.046875,  0.007827, -0.002971,    0.004883
Proceso concluido exitosamente con el nivel de error <=   0.005000
Raiz aproximada:  13.041992  Iteraciones: 9
```

El error relativo entre el método gráfico y el de bisección sería:

Er = (13.041992-13.02)/13.041992 = 0.00168624547538, aproximadamente el 0.16%.

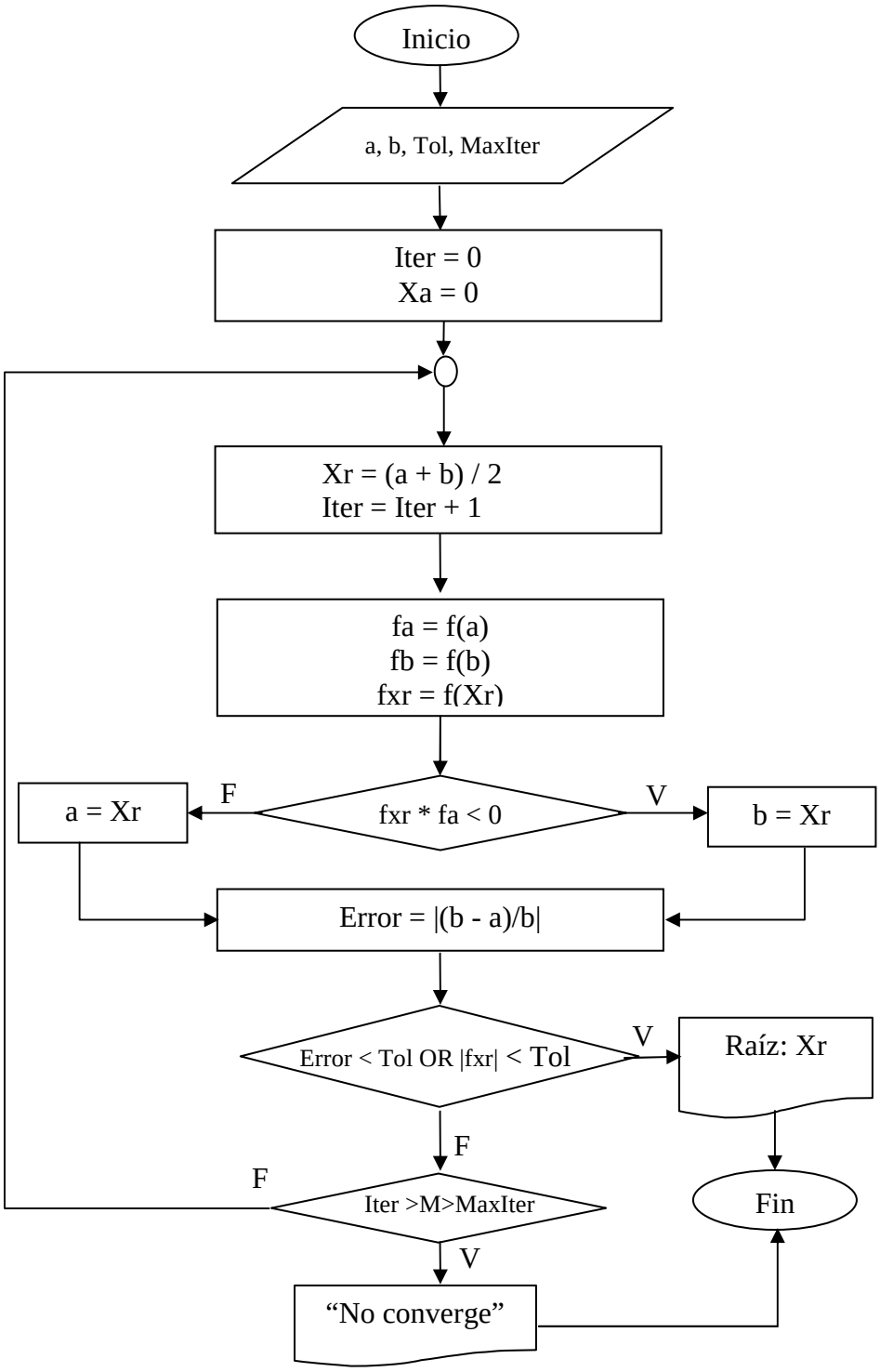


Figura 5. Diagrama de flujo para el algoritmo del método de Bisección.

En este método en cada iteración el intervalo se divide en dos, por lo tanto en **n** iteraciones se ha dividido el intervalo en $\mathbf{2^n}$ partes. Si deseamos obtener un determinado ***error***, este hecho podemos usarlo para hallar de antemano cuantas

iteraciones realizar. $E_a^n = \frac{|b-a|}{2^n}$ donde E_a^n es el error aproximado en la iteración n. Aplicando logaritmos podemos despejar n y obtenemos: $n = \frac{Log\left(\frac{|b-a|}{E_a^n}\right)}{Log(2)}$ que será el número de iteraciones necesarias para obtener la respuesta aproximada con el nivel de error deseado.

1.1.4 Falsa posición (Regula falsi)

La aproximación a la raíz se toma como la intersección de la línea que une los puntos (a,f(a)) y (b,f(b)) con el eje X. Por semejanza de triángulos se puede expresar: $\frac{-f(a)}{Xr-a} = \frac{f(b)}{b-a}$ de donde podemos derivar la fórmula: $Xr = b - \frac{f(b)(a-b)}{f(a)-f(b)}$ [1].

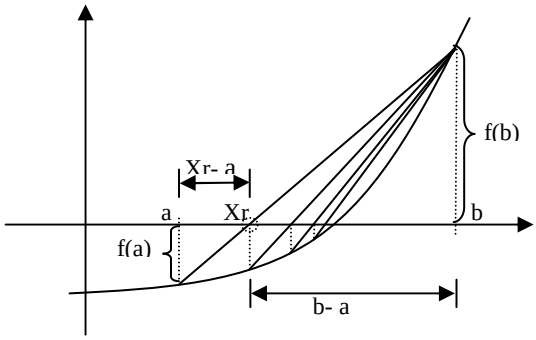


Figura 6. Modelo gráfico.

El algoritmo para hallar raíces es entonces en esencia el mismo planteado para el método de bisección, pero reemplazando la fórmula para calcular la aproximación a la raíz *Xr* (Numeral 4). El programa codificado en matlab es el siguiente:

```
%Programa para calcular raíces de ecuaciones, basado en el algoritmo de Falsa
%Posición
%Autor: Ing. Edgar Romero Rodríguez
%Docente fundación Universiraria UNISANGIL
%San Gil,Santander - Colombia. Enero de 2004

function FalsaPosicion
clc;
clear all;
a=input('Valor de la cota inferior: ');
b=input('Valor de la cota superior: ');
TOL=input('Tolerancia de error  : ');
max_iter=input('Numero de iteracciones: ');
%Modelo gráfico
x=a:0.01:b;
f=funcion(x);
plot(x,x,x,f);
grid;
title ('Grafica de la función en el intervalo dado');
```

[1] Ver demostración completa en el libro de métodos numéricos de Chapra y P. Canale, página 142.

```
%Termina la creación del modelo gráfico
iter = 0; %Inicio el contador de iteracciones
% Se imprimen los titulos generales
fprintf('\nMETODO DE FALSA POSICION...\n\n')
fprintf('Iter a Xr b f(a) f(b) f(xr) Error\n');
% El criterio de convergencia se cumple cuando la función evaluada en la
% raiz sea menor que el error máximo aceptado.
while 1 %  Iniciamos el proceso
  fa = funcion(a); %Evaluamos la funcion en la cota inferior
  fb = funcion(b); %evaluamos la funcion en la coa superior
  xr = b - (fb*(a-b)) / (fa - fb); %Calculo el valor de la aproximacion
  fxr= funcion(xr); %Evaluamos la funcion en la posible raiz
  error = abs(fxr);
  % Impresion de resultados en cada iteraccion
fprintf('%4.0f,%10.6f,%10.6f,%10.6f,%16.6f,%16.6f,%16.6f,%12.4e\n',iter,a,xr,b,fa,fb,fxr,err
or)
  if error <= TOL %Si supero el error aceptado detengo el proceso
    fprintf('Proceso concluido exitosamente con el nivel de error <= %12.2e\n\n',TOL)
    break;
  end
  if fa*fxr < 0
      b=xr;  %La raiz esta en el primer intervalo
  else
      a=xr; %La raiz esta en el segundo intervalo
  end
  if (iter>max_iter)  %Verifico si se excede el numero de iteracciones
      fprintf('\nNumero de iteracciones excedido...\n\n')
      break;
  end
  iter=iter+1; %Incremento el numero de iteracciones
end
fprintf('\nRaiz aproximada: %12.6f',xr);
fprintf('  Iteraciones: %5.0f\n',iter);

function y=funcion(x) %Aca es donde dobo colocar la funcion a evaluar
y=x.^3+2.*x.^2+10.*x-20;
%Fin del programa
```

Ejemplo 4: esta ecuación fue analizada por Leonardo de Piza, en el año 1.225:

$$f(x) = X^3 + 2X^2 + 10X - 20.$$

Quien la resolvió con 10 cifras decimales, usando el método de la falsa posición. Veamos el proceso seguido. Gráficamente se observa que la raíz está en el intervalo [1, 2] y definimos $Tol = 5*10^{-11}$, como el error máximo aceptado.

Primera iteración:

- Evaluamos la función en los extremos del intervalo:

  f(1) = -7 y f(2) = 16 (la función cambia de signo en el intervalo)

- Calculamos la posible raíz:

  Xr = 2 – 16*(1 – 2)/(-7 – 16) = 1,30434782609

- Evaluamos:

  f(Xr)=1,30434782609³ + 2*1,30434782609² + 10*1,30434782609 - 20 = -1,33475795184

En este punto no tenemos aun datos para evaluar el error relativo pero podemos evaluar */f(Xr)/ < Tol* y vemos que no cumple la condición, por lo tanto pasamos a la siguiente iteración.

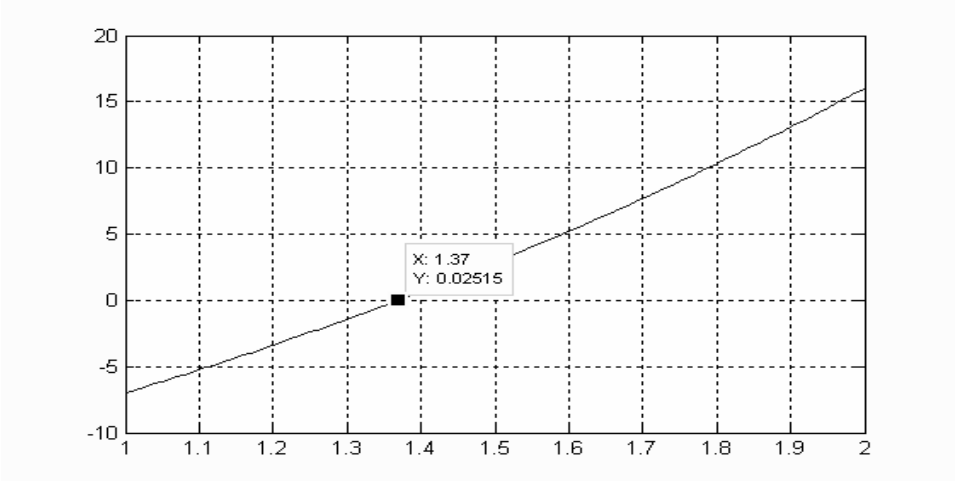


Figura 7. Modelo gráfico de la ecuación de Leonardo de Piza.

Segunda iteración:

- Verificamos en que subintervalo queda la nueva aproximación de la raíz. Para ello evaluamos el producto ***f(Xl)*f(Xu)*** y vemos que es mayor que cero, por lo cual variamos la cota inferior pasándola a **Xr**, el nuevo intervalo a procesar es **[1,30434782609, 2].**
- Evaluamos f(1,30434782609) = -1,33475795, f(2) = 16
- Calculamos la posible raíz:

  Xr = 2 – 16*(-1,33475795 – 2)/(-1,33475795 - 16) = 1,35791230466
- Evaluamos la función en la nueva raíz: f(1,35791230466) = -0,22913572959
- Calculamos el error como la diferencia entre las dos últimas iteraciones:

  Error = Abs(-1,33475795184 - -0,22913572959) = 0,05356447857

El valor es mayor que el error máximo aceptado, *Tol*, por lo tanto realizamos una nueva iteración.

Los cálculos de las siguientes se realizaron con ayuda de la hoja electrónica y sus resultados se muestran a continuación:

Iter	a	b	f(a)	f(b)	Xr	f(xr)	Error
1	1	2	-7	16	1,30434782609	-1,33475795184	
2	1,30434783	2	-1,33475795	16	1,35791230466	-0,22913572959	0,05356447857
3	1,3579123	2	-0,22913573	16	1,36697780482	-0,03859187678	0,00906550016
4	1,3669778	2	-0,03859188	16	1,36850097560	-0,00647872815	0,00152317078
5	1,36850098	2	-0,00647873	16	1,36875657901	-0,00108704283	0,00025560341
6	1,36875658	2	-0,00108704	16	1,36879946288	-0,00018237436	0,00004288388
7	1,36879946	2	-0,00018237	16	1,36880665748	-0,00003059668	0,00000719459
8	1,36880666	2	-3,0597E-05	16	1,36880786450	-0,00000513315	0,00000120702
9	1,36880786	2	-5,1331E-06	16	1,36880806700	-0,00000086118	0,00000020250
10	1,36880807	2	-8,6118E-07	16	1,36880810097	-0,00000014448	0,00000003397
11	1,3688081	2	-1,4448E-07	16	1,36880810667	-0,00000002424	0,00000000570
12	1,36880811	2	-2,4239E-08	16	1,36880810763	-0,00000000407	0,00000000096
13	1,36880811	2	-4,0665E-09	16	1,36880810779	-0,00000000068	0,00000000016
14	1,36880811	2	-6,8223E-10	16	**1,36880810782**	-0,00000000011	0,00000000003

En la iteración 14 vemos como se alcanza la tolerancia de error deseada y por lo tanto se finaliza el proceso encontrándose una raíz aproximada. ***Xr = 1,36880810782***.

Usando Matlab y el algoritmo de ***FalsaPosicion*** mostrado antes obtenemos los siguientes resultados:

```
Valor de la cota inferior: 1
Valor de la cota superior: 2
Tolerancia de error  : 5e-11
Numero de iteracciones: 50

METODO DE FALSA POSICION...

 Iter      a          Xr          b          f(a)          f(xr)          f(b)          Error
   0,  1.000000,  1.304348,  2.000000,  -7.000000,  -1.334758,  16.000000, 1.3348e+000
   1,  1.304348,  1.357912,  2.000000,  -1.334758,  -0.229136,  16.000000, 2.2914e-001
   2,  1.357912,  1.366978,  2.000000,  -0.229136,  -0.038592,  16.000000, 3.8592e-002
   3,  1.366978,  1.368501,  2.000000,  -0.038592,  -0.006479,  16.000000, 6.4787e-003
   4,  1.368501,  1.368757,  2.000000,  -0.006479,  -0.001087,  16.000000, 1.0870e-003
   5,  1.368757,  1.368799,  2.000000,  -0.001087,  -0.000182,  16.000000, 1.8237e-004
   6,  1.368799,  1.368807,  2.000000,  -0.000182,  -0.000031,  16.000000, 3.0597e-005
   7,  1.368807,  1.368808,  2.000000,  -0.000031,  -0.000005,  16.000000, 5.1331e-006
   8,  1.368808,  1.368808,  2.000000,  -0.000005,  -0.000001,  16.000000, 8.6118e-007
   9,  1.368808,  1.368808,  2.000000,  -0.000001,  -0.000000,  16.000000, 1.4448e-007
  10,  1.368808,  1.368808,  2.000000,  -0.000000,  -0.000000,  16.000000, 2.4239e-008
  11,  1.368808,  1.368808,  2.000000,  -0.000000,  -0.000000,  16.000000, 4.0665e-009
  12,  1.368808,  1.368808,  2.000000,  -0.000000,  -0.000000,  16.000000, 6.8223e-010
```

```
 13,  1.368808,  1.368808,  2.000000,      -0.000000,      -0.000000,      16.000000, 1.1445e-010
 14,  1.368808,  1.368808,  2.000000,      -0.000000,      -0.000000,      16.000000, 1.9202e-011
Proceso concluido exitosamente con el nivel de error <=    5.00e-011

Raíz aproximada:   1.368808  Iteraciones:   14
```

Los resultados son prácticamente los mismos que los obtenidos con la hoja electrónica. La diferencia en cifras se debe al número de decimales que imprime el algoritmo programado en MatLab.

La convergencia del método de falsa Posición, en la mayoría de los casos es más rápida que la del método de bisección. Si evaluamos la ecuación de Leonardo de Piza, por el método de bisección se reqieren 34 iteraciones, mientras que el método de falsa Posición solo realizó 14, esto se debe a que el método de bisección no toma en cuenta el hecho de que la raíz puede quedar más cerca de uno de los extremos del intervalo, situación aprovechada por el método de falsa posición.

1.2 MÉTODOS ABIERTOS

Se caracterizan porque no se requiere especificar de antemano un intervalo en el cual se halle la raíz. Sólo se necesita una aproximación inicial a la misma (cualquier valor puede servir), más sin embargo un modelo gráfico ayuda a definir esta aproximación. El inconveniente de estos métodos está en que algunas veces la convergencia no se da por lo tanto es necesario implementar controles para evitar que el algoritmo se detenga cuando se esté alejando de la raíz verdadera, por ejemplo controlando el número de iteraciones a realizar.

1.2.1 Iteración de punto fijo (Sustituciones sucesivas)

En este método e necesario reacomodar la ecuación de modo que podamos obtener una ecuación de la forma: **x = g(x)**, ya sea despejando una **x** de la ecuación original o sumando **x** a ambos lados de la misma.

Gráficamente podemos usar el método de las dos curvas. Graficar $f(x) = x$ y graficar $g(x)$, el punto en donde se corten se proyecta sobre el eje x y se toma como raíz aproximada. En este método la convergencia se da siempre y cuando g'(x) < 1.

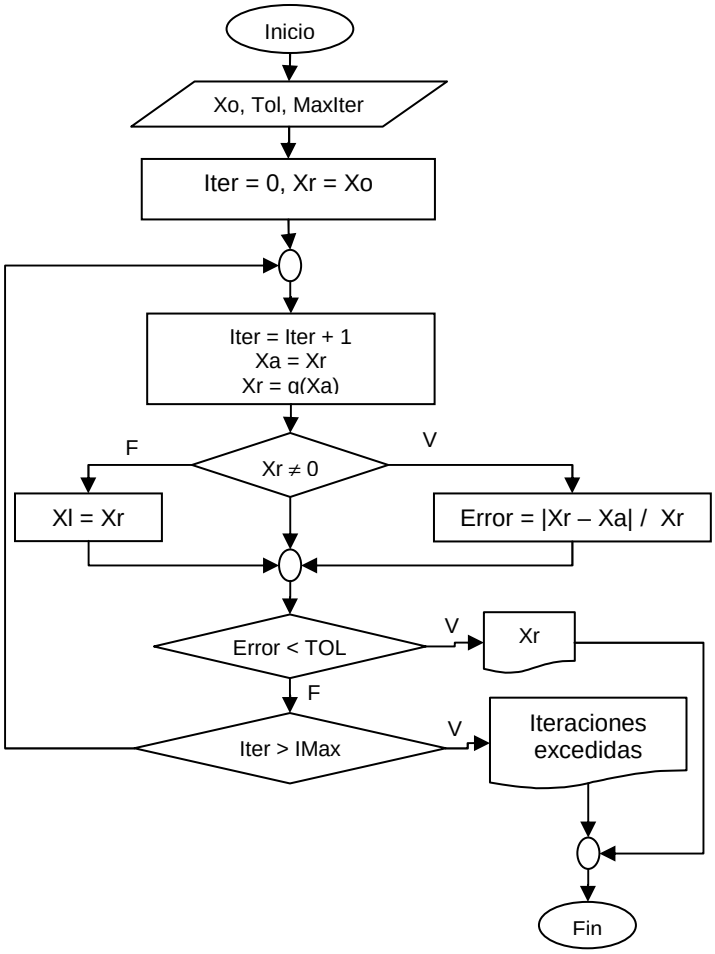
Algoritmo:

1. Definir el error máximo a aceptar (*Tol*).
2. Definir una aproximación inicial cualquiera (Xo)
3. Definir el máximo número de iteraciones a realizar (MaxIter)
4. Reacomodar f(x) para obtener la ecuación: $x = g(x)$
5. Establecer a cero el contador de iteraciones (Iter)
6. Tomar Xo como primera aproximación de la raiz (Xr = Xo)

7. **Repetir estos pasos:**

a. Guardar el valor de la aproximación anterior (Xa = Xr).
b. Calcular una nueva aproximación: Xr = g(Xa)
c. Incrementar el número de iteraciones (Iter = Iter + 1)
d. Si Xr ≠ 0 entonces calcular: Error = |Xr – Xa| / Xr.
e. Si Error < *Tol* entonces Imprimir Xr como aproximación de la raíz y parar.
f. Si Iter > MaxIter entonces Imprimir mensaje de error de que el proceso fallo y parar.
g. Volver al paso 7.

DIAGRAMA DE FLUJO PARA EL METODO DE PUNTO FIJO

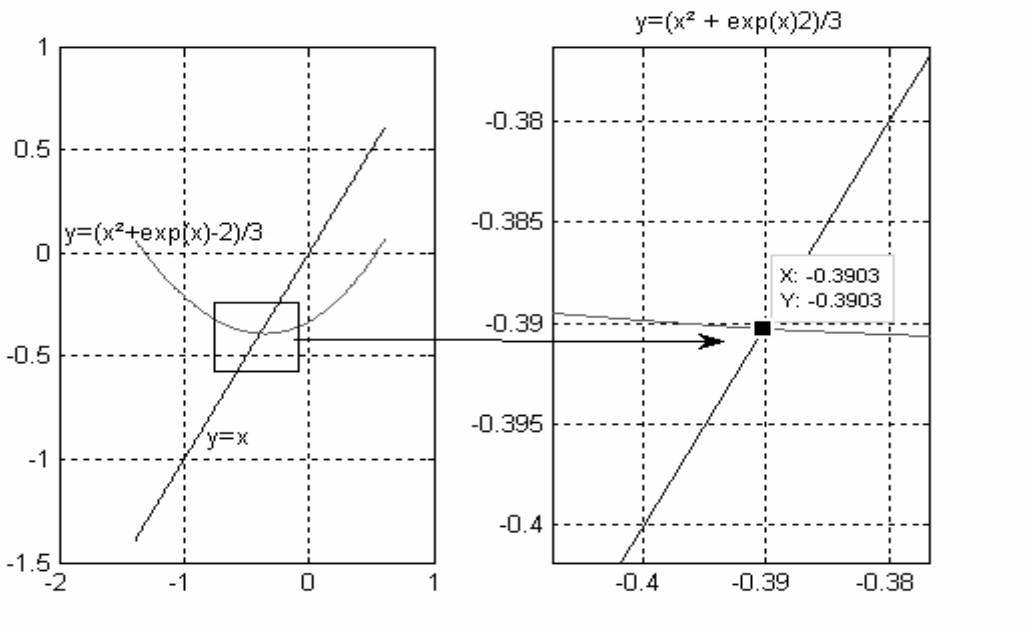


Ejemplo:

Resolver la ecuación: $f(x) = X^2 - 3X + e^X - 2$.

Se despejamos ***x*** de la ecuación podemos obtener:
x = g(x), donde g(x) = $\boldsymbol{(X^2 + e^X - 2)/3.}$

Usando MatLab podemos ver la solución al elaborar la gráfica de X y de g(x) respectivamente así:



Definimos:

- Error máximo aceptado: *Tol = 5*10^{-11}*.
- Número máximo de iteraciones a realizar: *MaxIter = 25*
- Aproximación inicial: *Xo = 0.*

Primera iteración:

Xr = g(Xo) = (0² + exp(0) -2)/3 = -0,33333333

Evaluamos el error como el error relativo entre las dos aproximaciones:

Er = Abs((-0.3333333333 - 0)/-0.333333) = 1,

El valor es mayor que el error máximo aceptado, por lo tanto continuamos el proceso.

Segunda iteración:

Calculamos la nueva aproximación como el valor de g(x) en la aproximación anterior:

Xr = g(-0,33333333) = -0,39078586

Evaluamos el error como el error relativo entre las dos aproximaciones:

Er = Abs((-0,39078586 -0.3333333333)/ -0,39078586) = 0,17235757832.

Es mayor que el error máximo aceptado, por lo tanto continuamos el proceso.

Tercera iteración:

Calculamos la nueva aproximación como el valor de g(x) en la aproximación anterior:

Xr = g(-0,39078586) = -0,3902538

Evaluamos el error como el error relativo entre las dos aproximaciones:

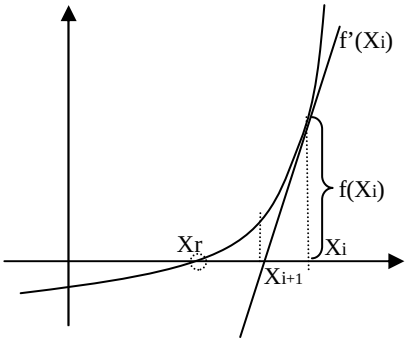
Er = Abs((-0,39078586 --0,3902538)/ -0,3902538) = 0,00136151142.

Es mayor que el error máximo aceptado, por lo tanto continuamos el proceso hasta lograr un error menor a Tol. Los resultados se muestran en la siguiente tabla y han sido hallados con la ayuda de la hoja electrónica.

Iter	Xr	f(Xr)	Error	Cumple CC
1	0	-0,33333333	1,000000000000	
2	-0,33333333	-0,39078586	0,172357578315	No
3	-0,39078586	-0,3902538	0,001361511416	No
4	-0,3902538	-0,3902723	0,000047415491	No
5	-0,3902723	-0,39027166	0,000001638085	No
6	-0,39027166	-0,39027169	0,000000056608	No
7	-0,39027169	-0,39027169	0,000000001956	No
8	-0,39027169	-0,39027169	0,000000000068	No
9		-0,39027169	0,000000000002	Si

En la novena iteración se logra superar el margen de error y se termina el proceso encontrándose una raíz aproximada en: **x = -0,39027169.**

1.2.2 **Método de Newton Raphson**



En este método, la aproximación a la raíz se toma como la proyección sobre el eje x de la tangente en el punto correspondiente a (X_i, $f(X_i)$). Siendo X_r la aproximación a la raíz, como se ve en la figura.

La recta que pasa por los puntos (X_{i+1},0) y (X_i, $f(X_i)$) está dada por: $f'(X_i) = \frac{f(X_i) - 0}{X_i - X_{i+1})}$ de donde podemos despejar X_{i+1}, obteniendo así la fórmula para calcular la siguiente iteración así: $X_{i+1} = X_i - \frac{f(X_i)}{f'(X_i)}$. El algoritmo para calcular el valor de la raíz es el siguiente:

Algoritmo:

1. Definir el error máximo a aceptar (Tol).
2. Definir una aproximación inicial cualquiera (Xo)
3. Definir el máximo número de iteraciones a realizar (MaxIter)
4. Establecer a cero el contador de iteraciones (Iter)
5. Tomar Xo como primera aproximación (Xr = Xo)
6. **Repetir estos pasos:**
   h. Guardar el valor de la aproximación anterior (Xa = Xr).
   i. Calcular una nueva aproximación: Xr = Xa – f(Xa) / f'(Xa)
   j. Incrementar el número de iteraciones (Iter = Iter + 1)
   k. Si Xr ≠ 0 entonces calcular: Error = |Xr – Xa| / Xr.
   l. Si Error < Tol entonces Imprimir Xr como aproximación de la raíz y parar.
   m. Si Iter > MaxIter entonces Imprimir mensaje de error de que el proceso fallo y parar.
   n. Volver al paso 6.

1.2.3 **Método de la secante**

Surge como una variación del método de Newton. Usando el concepto de diferencia dividida para evaluar la derivada de la función con base en los dos valores anteriores de modo que la nueva aproximación puede calcularse como:

$$X_i = X_{i-1} - \frac{X_{i-1} - X_{i-2}}{f(X_{i-1}) - f(X_{i-2})} \text{ para i=2,3,...}$$

En este caso para iniciar el proceso se requiere de dos aproximaciones iniciales pero no necesariamente deben ser un intervalo que incluya la raíz, incluso la segunda aproximación se pude calcular como $X_{i-1}+\Delta x$ siendo Δx un valor pequeño como 0.01.

Ejemplo: Halla la primera raíz positiva de la ecuación de Leonardo de Piza:

$f(x) = X^3 + 2X^2 + 10X - 20$.

Aproximación inicial: X_0=1 y X_1 = 1.01. Tol = 5E-11.

Primera iteración:
Evalúo la función en la aproximación inicial: f(1) = -7 y f(1.01) = -6.82949.
Calculo la aproximación a la raíz:

X_2=-7 – (1.01 - 1)/(-7 -6.82949) = 1.41055478

Evalúo el error: Ea=|1.41055478 – 1.01|/1.41055478 = 0.283969.

Como el valor del error es mayor que Tol, se continúa el proceso.

Segunda iteración:
Evalúo la función en la nueva aproximación: f(1.4105549)=0.89140848
Calculo la aproximación a la raíz:

X_3=1.41055478 – (1.41055478 - 1.01)/(-6.82949 - 0.89140848) = 1,35867554

Evaluación del error: Ea=|1,35867554 - 1.41055478|/1,35867554 = 0.03818.

Como el valor del error es mayor que Tol, se continúa el proceso.

Tercera iteración:
Evalúo la función en la nueva aproximación: f(1,35867554)=-0.213132
Calculo la aproximación a la raíz:

X_4=1.41055478 – (1.41055478 - 1.01)/(-6.82949 - 0.89140848) = 1,368686

Evaluación del error: Ea=|1,368686 - 1,35867554|/1,368686= 0.007314.

Como el valor del error es mayor que Tol, se continúa el proceso.

De esa misma manera se continúa el proceso hasta lograr el error tolerado. Los resultados se muestran en la siguiente tabla.

Iter	Xi-2	Xi-1	Xi	F(Xi-1)	f(Xi-2)	F(Xi)	Error
1	1	1,01	1,41055478	-6,829499	-7	0,89140848	0,283969671644
2	1,01	1,41055478	1,35867554	0,89140848	-6,829499	0,30566022	0,038183682435
3	1,41055478	1,35867554	1,36868616	-0,21313213	0,89140848	-0,10267875	0,007314033313
4	1,35867554	1,36868616	1,36880847	-0,00257259	-0,21313213	-0,00121553	0,000089353921
5	1,36868616	1,36880847	1,36880811	7,5552E-06	-0,00257259	3,5812E-06	0,000000261648
6	1,36880847	1,36880811	1,36880811	-2,667E-10	7,5552E-06	-1,2642E-10	0,000000000009

En la sexta iteración se alcanza el nivel de error deseado por lo tanto se detiene el proceso y se toma como valor de la raíz aproximada de 1.3688011.

1.1. RAICES MÚLTIPLES

1.2. SISTEMAS DE ECUACIONES NO LINEALES