

**Instituto Politécnico Nacional**

**Unidad Profesional Interdisciplinaria de Biotecnología**

**Métodos Numéricos (taller)**

**Grupo 4LM1**

**Equipo 6**

**Primer Departamental**

**Tarea #3: *Raíces: métodos cerrados y abiertos.***

**Profesores:**

**Flores Núñez José Ignacio**

**Bueno Hernández Diana**

**Alumnos:**

**Carrillo Cadena Sofia. *Ingeniería Biotecnológica***

**García Maldonado Francisco Javier. *Ingeniería Biotecnológica***

**Lagunas Pérez Alan Ernesto. *Ingeniería Farmacéutica***

**Nava Díaz Daniela Macarena. *Ingeniería en Alimentos***

**Pérez Pacheco Josseline. *Ingeniería Biotecnológica***

**Fecha de entrega: 20 de Febrero del 2019**

***BUSQUEDA DE RAÍCES***

**INTRODUCCIÓN**

Las raíces de una ecuación se definen como los valores de ***x***que hacen ***f*(*x*) = 0**. Debido a esto, algunas veces a las raíces se les conoce como *ceros* de la ecuación. Aunque las raíces de ecuaciones aparecen en el contexto de diversos problemas, son frecuentes en el área de diseño en ingeniería y pueden ser obtenidas a través de los métodos numéricos.

**Métodos cerrados**

Los métodos cerrados aprovechan el hecho de que una función cambia de signo en la vecindad de una raíz. Dichos métodos utilizan intervalos para encontrar raíces. Empiezan con intervalos que encierran o contienen a la raíz, y después reducen sistemáticamente el tamaño del intervalo. Uno de estos métodos, empleado en el presente trabajo, es el *método de bisección*.

Como preámbulo de esta y otras técnicas se analizarán los *métodos gráficos* para representar tanto las funciones como sus raíces. Estos sirven para dar una comprensión visual de las técnicas y para determinar valores iniciales y finales del intervalo a trabajar.

Se dice que dice los métodos cerrados son *convergentes* porque se acercan progresivamente a la raíz a medida que se avanza en el cálculo.

* Método gráfico

Un método simple para obtener una aproximación a la raíz de la ecuación ***f (x) = 0*** consiste en graficar la función y observar dónde cruza el eje x. Este punto, que representa el valor de **x** para el cual f(***x) = 0***, ofrece una aproximación inicial de la raíz.

Las interpretaciones gráficas, además de proporcionar estimaciones de la raíz, son herramientas importantes en la comprensión de las propiedades de las funciones y en la prevención de las fallas de los métodos numéricos.

* Método de bisección

Los *métodos de búsqueda incremental* aprovechan la característica de que ***f(x)*** cambia de signo a ambos lados de una raíz, localizando un intervalo en el que la función cambie de positiva a negativa. Entonces, la localización del cambio de signo (y, en consecuencia, de la raíz) se logra con más exactitud al dividir el intervalo.

El *método de bisección*, conocido también como de corte binario, de partición de intervalos o de Bolzano, es un tipo de búsqueda incremental en el que el intervalo se divide siempre a la mitad. Si la función cambia de signo sobre un intervalo, se evalúa el valor de la función en el punto medio. La posición de la raíz se determina situándola en el punto medio del subintervalo, dentro del cual ocurre un cambio de signo. El proceso se repite hasta obtener una mejor aproximación.

**Métodos abiertos**

Estos métodos también emplean iteraciones sistemáticas de prueba y error; pero no requieren que el intervalo inicial encierre a la raíz. En general, son más eficientes computacionalmente que los métodos cerrados, aunque no siempre funcionan. Se analizan los métodos de *iteración de un punto fijo* y de *Newton-Raphson.* Los *métodos* *gráficos* sirven para dar una idea geométrica en los casos donde los métodos abiertos no funcionan.

Los tipos de *métodos abiertos* mencionados se basan en fórmulas que requieren únicamente de un solo valor de inicio ***x***o que empiecen con un par de ellos, pero que no necesariamente contienen encerrada a la raíz.

Éstos, algunas veces *divergen* o se alejan de la raíz verdadera a medida que se avanza en el cálculo Sin embargo, cuando los métodos abiertos *convergen*, en general lo hacen mucho más rápido que los métodos cerrados.

* Iteración simple de punto fijo

La fórmula que emplean los métodos abiertos para predecir la raíz puede desarrollarse como una iteración simple de punto fijo (también llamada *iteración de un punto* o *sustitución sucesiva* o *método de punto fijo*), al arreglar la ecuación ***f(x) = 0*** de tal modo que ***x*** esté del lado izquierdo de la ecuación: ***x = g(x).***

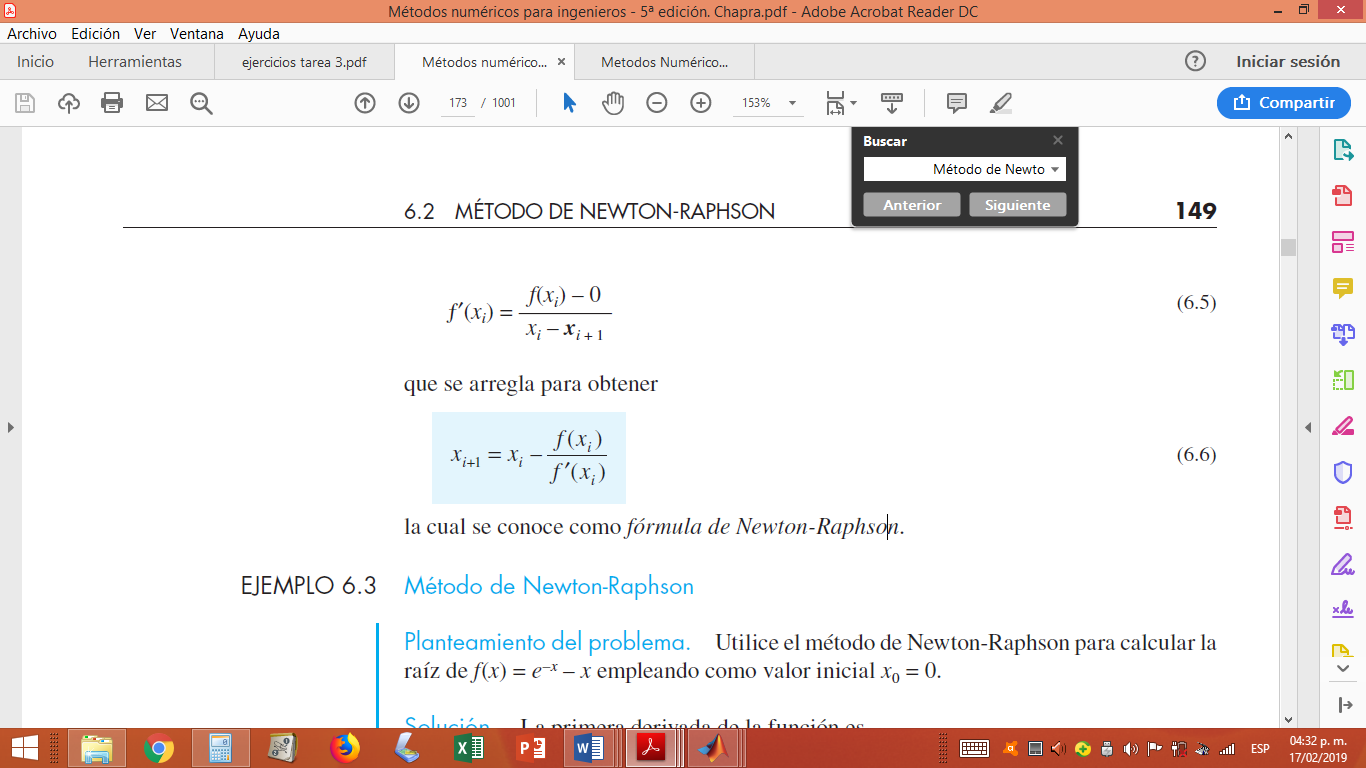
Esta transformación se realiza mediante operaciones algebraicas o simplemente sumando *x* a cada lado de la ecuación original. Por ejemplo, ***x2 – 2x + 3 = 0*** se arregla para obtener ***x* =** mientras que ***sen x = 0*** puede transformarse sumando ***x*** a ambos lados para obtener ***x= sen x + x***

La utilidad de la ecuación ***x = g(x)***  es que proporciona una fórmula para predecir un nuevo valor de ***x*** en función del valor anterior de ***x****.* De esta manera, dado un valor inicial para la raíz ***xi***, la ecuación se utiliza para obtener una nueva aproximación ***xi+*1**, expresada por la fórmula iterativa ***xi*+1 = *g*(*xi).***

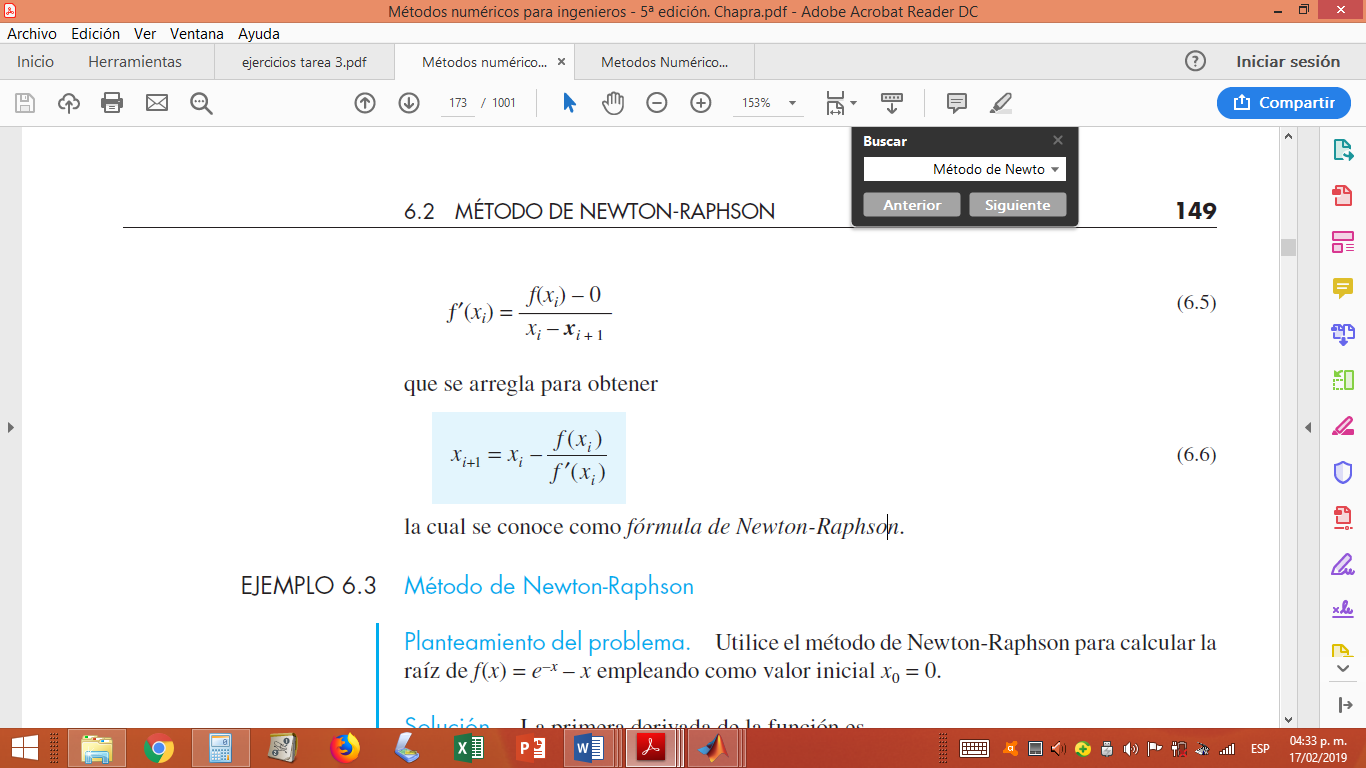
* Método de Newton-Raphson

Tal vez, de las fórmulas para localizar raíces, la fórmula de Newton-Raphson sea la más ampliamente utilizada. Si el valor inicial para la raíz es ***xi***, entonces se puede trazar una tangente desde el punto **[*xi, f*(*xi*)]** de la curva. Por lo común, el punto donde esta tangente cruza al eje *x* representa una aproximación mejorada de la raíz.

El método de Newton-Raphson se deduce a partir de esta interpretación geométrica. Se tiene que la primera derivada en ***x***es equivalente a la pendiente:



que se arregla para obtener:



la cual se conoce como *fórmula de Newton-Raphson.*

No hay un criterio general de convergencia para el método de Newton-Raphson. Su convergencia depende de la naturaleza de la función y de la exactitud del valor inicial. La única solución para esto es tener un valor inicial que sea “suficientemente” cercano a la raíz (y para algunas funciones ningún valor inicial funcionará).

Los buenos valores iniciales por lo común se predicen con un conocimiento del problema físico o mediante el uso de recursos alternativos, tales como las *gráficas*, que proporcionan mayor claridad en el comportamiento de la solución.

**OBJETIVOS**

El alumno empleará métodos cerrados y métodos abiertos para la solución de ecuaciones de una variable.

**DESARROLLO DE LA PRACTICA (REPORTE)**

**-Métodos cerrados**

1. **f(x)= x3-9x2-x+5**

Código:

clear all

clc

P3= [1 -9 -1 5]

roots(P3)

9.0494

-0.7685

0.7190

f= inline (‘x^3-9\*x^2-x+5’)

ezplot(f)

grid on

ezplot(f [-2,10])

grid on

1. **f(x)= sen(x)-2cos(x)=2-x2**

Código:

f= inline(‘sin(x)-2\*cos(x)-2+x^2’)

ezplot(f)

grid on

1. **(‘sen(x/2) (cos()) - 1/5’)**

Código:

ezplot(f,[0,pi]

grid on

fzero(f,0.5)

fzero(f,1.69)

1. **Hallar un valor aproximado de la raíz de ex – 2 = 0, en el intervalo de 0 a 2 con una tolerancia de 0.01.**

Código:

clear all

clc

f= inline(‘exp(x) – 2’)

disp (‘Pimer interación’)

a= 0

b= 2

r= (b+a)/2

f(a)

f(r)

f(b)

error= (b-a)/2

disp (‘Segunda iteración’)

a2= 0

b2= 1

r2= (b2+a2)/2

f(a2)

f(r2)

f(b2)

error2= (b2-a2)/2

disp (‘Tercera iteración’)

a3= 0.5

b3= 1

r3= (b3+a3)/2

f(a3)

f(r3)

f(b3)

error3= (b3-a3)/2

disp (‘Cuarta iteración’)

a4= 0.5

b4= 0.75

r4= (b4+a4)/2

f(a4)

f(r4)

f(b4)

error4 = (b4-a4)/2

disp (‘Quinta iteración’)

a5= 0.625

b5= 0.75

r5= (b5+a5)/2

f(a5)

f(r5)

f(b5)

error5= (b5-a5)/2

disp (‘Sexta iteracion’)

a6= 0.6875

b6= 0.75

r6= (b6+a6)/2

f(a6)

f(r6)

f(b6)

error6= (b6-a6)/2

disp (‘Septima iteracion’)

a7= 0.6875

b7= 0.7188

r7= (b7+a7)/2

f(a7)

f(r7)

f(b7)

error7= (b7-a7)/2

disp (‘Octava iteracion’)

a8= 0.6875

b8= 0.7031

r8= (b8+a8)/2

f(a8)

f(r8)

f(b8)

error8= (b8-a8)/2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| n | a | r(n) | b | f(a) | f(r(n)) | f(b) | e(n) |
| 1 | 0 | 1 | 2 | - | + | + | 1 |
| 2 | 0 | ½ | 1 | - | - | + | 0.5 |
| 3 | ½ | 0.75 | 1 | - | + | + | 0.25 |
| 4 | 0.5 | 0.625 | 0.75 | - | - | + | 0.125 |
| 5 | 0.625 | 0.6875 | 0.75 | - | - | + | 0.0625 |
| 6 | 0.6875 | 0.7188 | 0.75 | - | + | + | 0.0313 |
| 7 | 0.6875 | 0.7031 | 0.7188 | - | + | + | 0.0156 |
| 8 | 0.6875 | 0.6953 | 0.7031 | - | + | + | 0.0078 |

**-Métodos abiertos**

Problema 1. Código

clear all

clc

f=inline('x^2-2\*x-3')

ezplot(f)

grid on

syms x

Xo=input('Punto inicial, Xo: ');

gx=sqrt(3+2\*x)

dgx=diff(gx,1)

edgx=vpa(subs(dgx,Xo),5)

format long

gx1=sqrt(3+2\*Xo)

gx2=sqrt(3+2\*gx1)

gx3=sqrt(3+2\*gx2)

gx4=sqrt(3+2\*gx3)

gx5=sqrt(3+2\*gx4)

gx6=sqrt(3+2\*gx5)

gx7=sqrt(3+2\*gx6)

gx8=sqrt(3+2\*gx7)

fgx7=(gx7)^2-2\*(gx7)-3

fgx8=vpa((gx8)^2-2\*(gx8)-3 , 5)

Problema 2. Código

%Newton-Raphson para exp(-x)-x

clear all

clc

f=inline('exp(-x)-x')

ezplot(f)

grid on

Xo=input('Punto inicial, Xo= ');

syms x

fx=exp(-x)-x

dfx=diff(fx,1)

%Primer iteración Xi

Xi=vpa(Xo-(subs(fx,Xo)/subs(dfx,Xo)),4)

er=abs((Xi-Xo)/Xi)

%Segunda iteración Xi

X2=vpa(Xi-(subs(fx,Xi)/subs(dfx,Xi)),4)

er=abs((X2-Xi)/X2)

%Tercera iteración Xi

X3=vpa(X2-(subs(fx,X2)/subs(dfx,X2)),4)

er=abs((X3-X2)/X3)

%Quinta iteración Xi

X4=vpa(X3-(subs(fx,X3)/subs(dfx,X3)),4)

er=abs((X4-X3)/X4)

**TAREA**

***Utilice los métodos de a) punto fijo y b) Newton-Raphson para resolver los siguientes ejercicios***

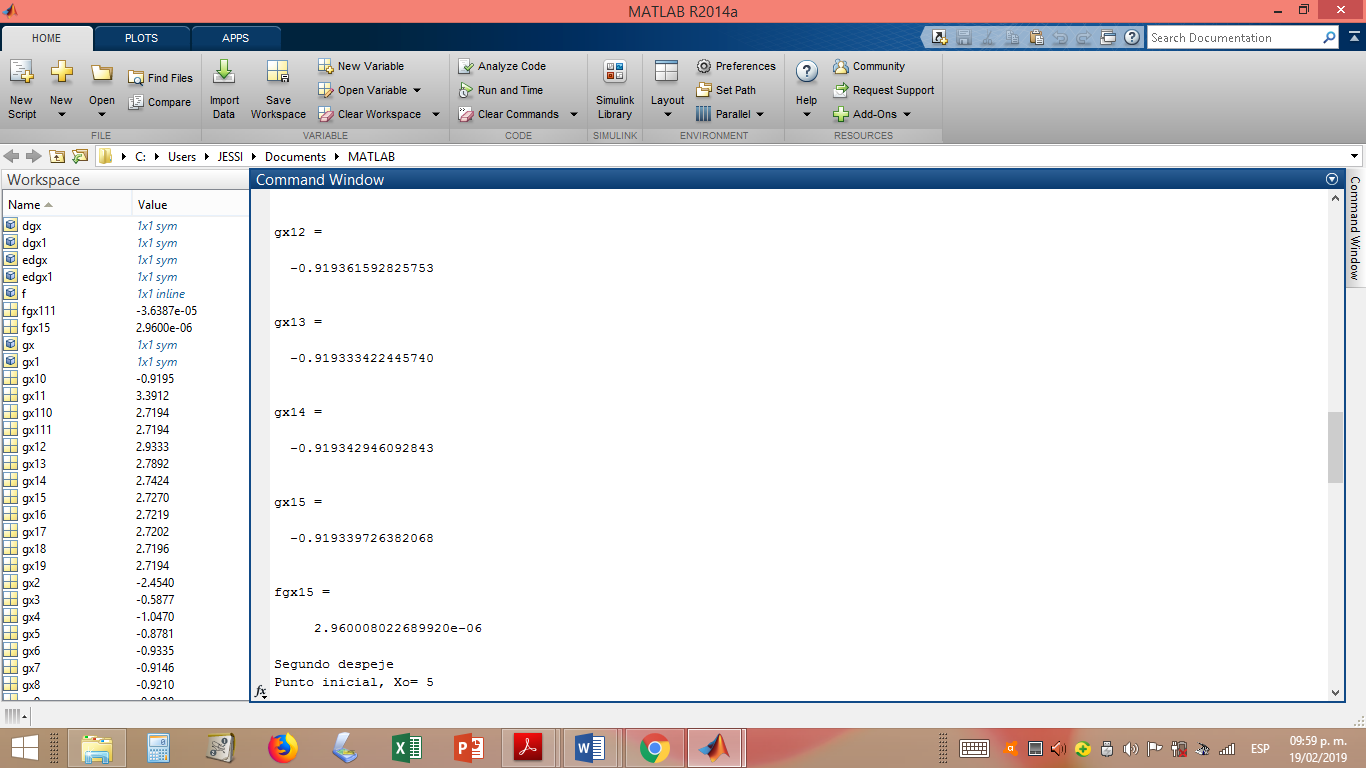
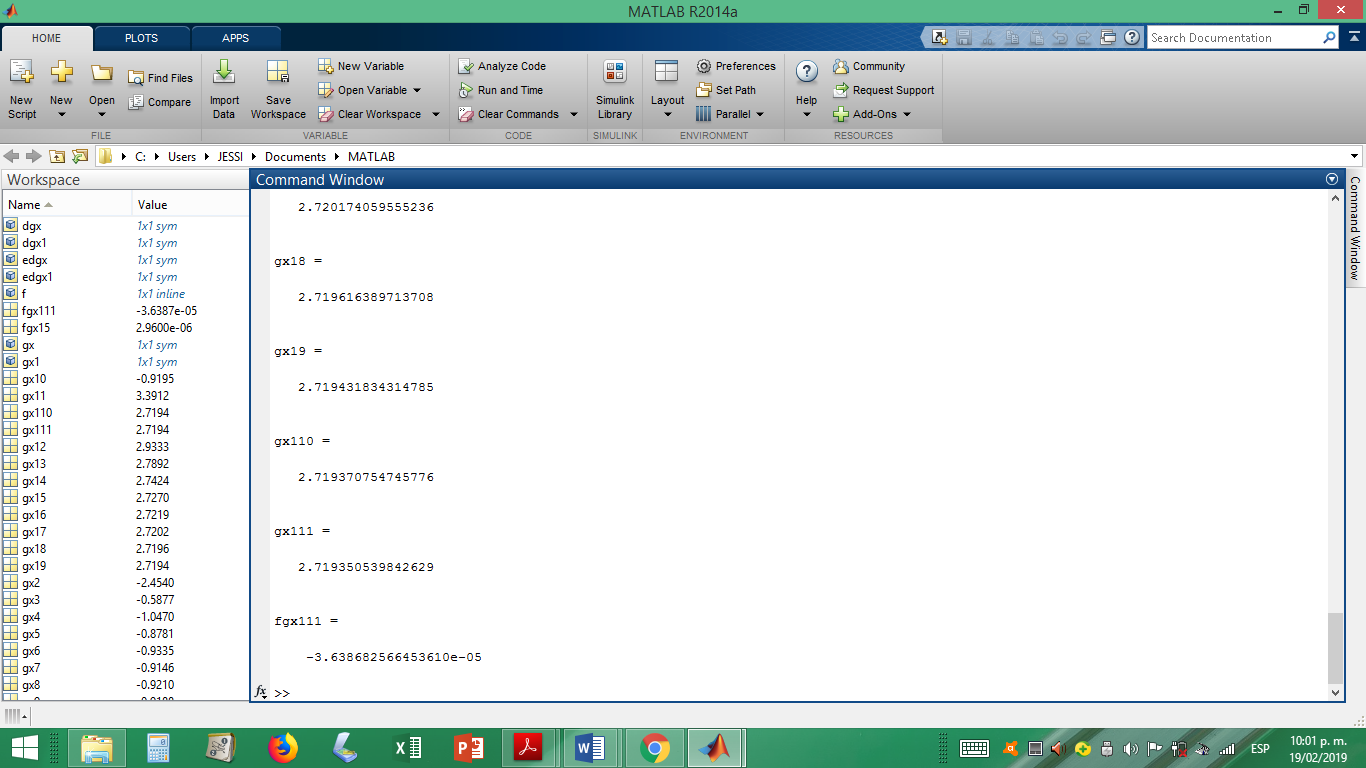
1.- Determinar la raíz de *f*(*x*) = –*x*2 + 1.8*x* + 2.5 con el uso de *x*0 = 5. Con una tolerancia de 0.05

**a) Punto fijo**

Respuesta

**Primera raíz**: -0.9193 **Segunda raíz:** 2.7193

**Error**: 2.96000802 x 10-6 **Error**: -3.6386 x 10-5

Código

clear all

clc

f=inline('-x^2+1.8\*x+2.5')

ezplot(f)

grid on

syms x

%Primer despje utilizado

disp('Primer despeje')

Xo=input('Punto inicial, Xo= ')

gx=(-2.5)/(-1\*x+1.8)

dgx=diff(gx,1)

edgx=vpa(subs(dgx, Xo),5)

format long

gx1=(-2.5)/((-Xo)+ 1.8)

gx2=(-2.5)/((-gx1)+ 1.8)

gx3=(-2.5)/((-gx2)+ 1.8)

gx4=(-2.5)/((-gx3)+ 1.8)

gx5=(-2.5)/((-gx4)+ 1.8)

gx6=(-2.5)/((-gx5)+ 1.8)

gx7=(-2.5)/((-gx6)+ 1.8)

gx8=(-2.5)/((-gx7)+ 1.8)

gx9=(-2.5)/((-gx8)+ 1.8)

gx10=(-2.5)/((-gx9)+ 1.8)

gx11=(-2.5)/((-gx10)+ 1.8)

gx12=(-2.5)/((-gx11)+ 1.8)

gx13=(-2.5)/((-gx12)+ 1.8)

gx14=(-2.5)/((-gx13)+ 1.8)

gx15=(-2.5)/((-gx14)+ 1.8)

fgx15=-gx15^2+1.8\*gx15+2.5

%Segundo despeje utilizado

disp('Segundo despeje')

Xo=input('Punto inicial, Xo= ')

gx1=sqrt(1.8\*x+2.5)

dgx1=diff(gx1,1)

edgx1=vpa(subs(dgx1, Xo),5)

format long

gx11=sqrt(1.8\*Xo+2.5)

gx12=sqrt(1.8\*gx11+2.5)

gx13=sqrt(1.8\*gx12+2.5)

gx14=sqrt(1.8\*gx13+2.5)

gx15=sqrt(1.8\*gx14+2.5)

gx16=sqrt(1.8\*gx15+2.5)

gx17=sqrt(1.8\*gx16+2.5)

gx18=sqrt(1.8\*gx17+2.5)

gx19=sqrt(1.8\*gx18+2.5)

gx110=sqrt(1.8\*gx19+2.5)

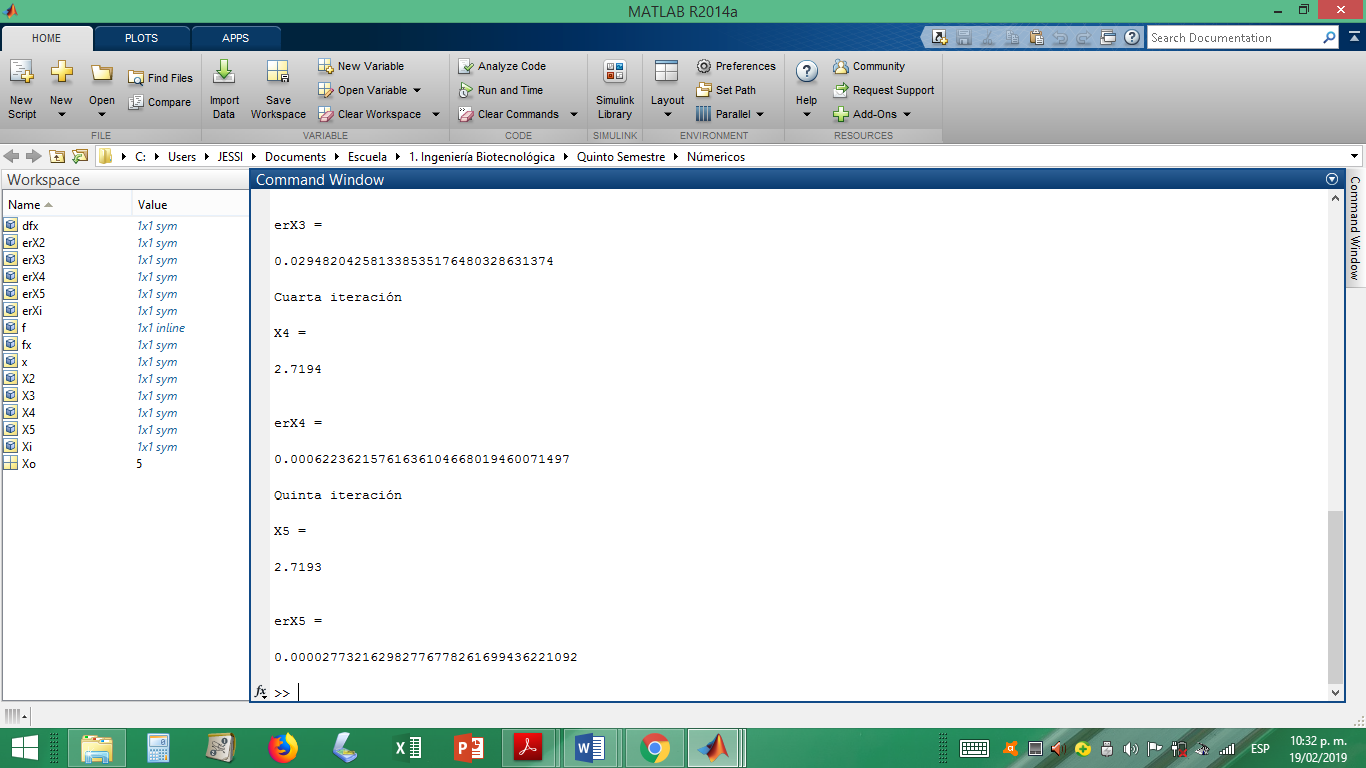
gx111=sqrt(1.8\*gx110+2.5)

fgx111=-gx111^2+1.8\*gx111+2.5

**b) Por Newton Raphson**

Respuesta

Para una Xo=5 y una tolerancia del 0.05, se obtuvo una de las raíces de la ecuación, igual a= **2.7194 (error de 0.0006).**



|  |  |  |
| --- | --- | --- |
| **Iteración** | **Xi** | **Error** |
| **0** | 0 |  |
| **1** | 3.3537 | 0.49090 |
| **2** | 2.8013 | 0.1971 |
| **3** | 2.7211 | 0.0294 |
| **4** | 2.7194 | 0.0006 |

Código

clear all

clc

f=inline('-x^2+1.8\*x+2.5')

ezplot(f)

grid on

Xo=input('Punto inicial Xo= ')

syms x

fx=-x^2+1.8\*x+2.5

dfx=diff(fx,1)

disp('Primera iteración')

Xi=vpa(Xo-(subs(fx,Xo)/subs(dfx,Xo)),5)

erXi=abs((Xi-Xo)/Xi)

disp('Segunda iteración')

X2=vpa(Xi-(subs(fx,Xi)/subs(dfx,Xi)),5)

erX2=abs((X2-Xi)/X2)

disp('Tercera iteración')

X3=vpa(X2-(subs(fx,X2)/subs(dfx,X2)),5)

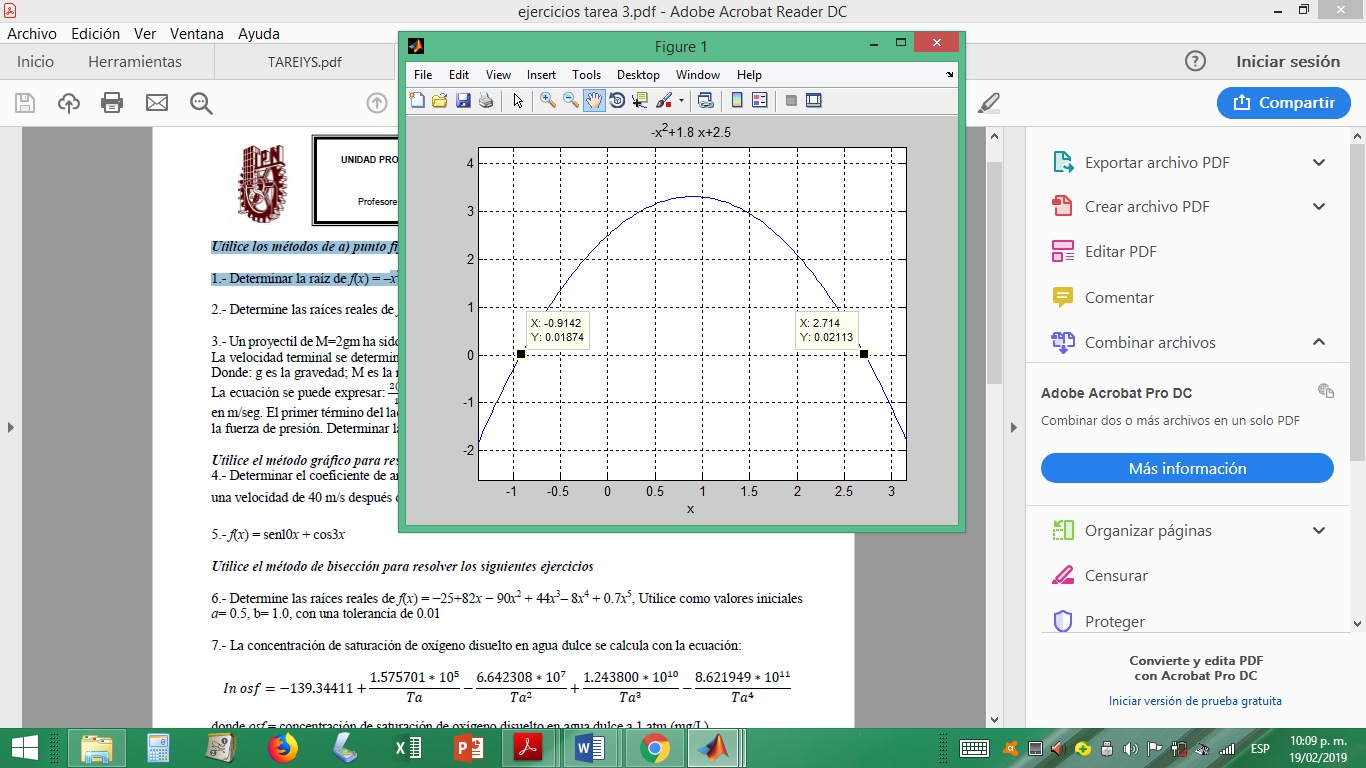
erX3=abs((X3-X2)/X3)

disp('Cuarta iteración')

X4=vpa(X3-(subs(fx,X3)/subs(dfx,X2)),5)

erX4=abs((X4-X3)/X4)

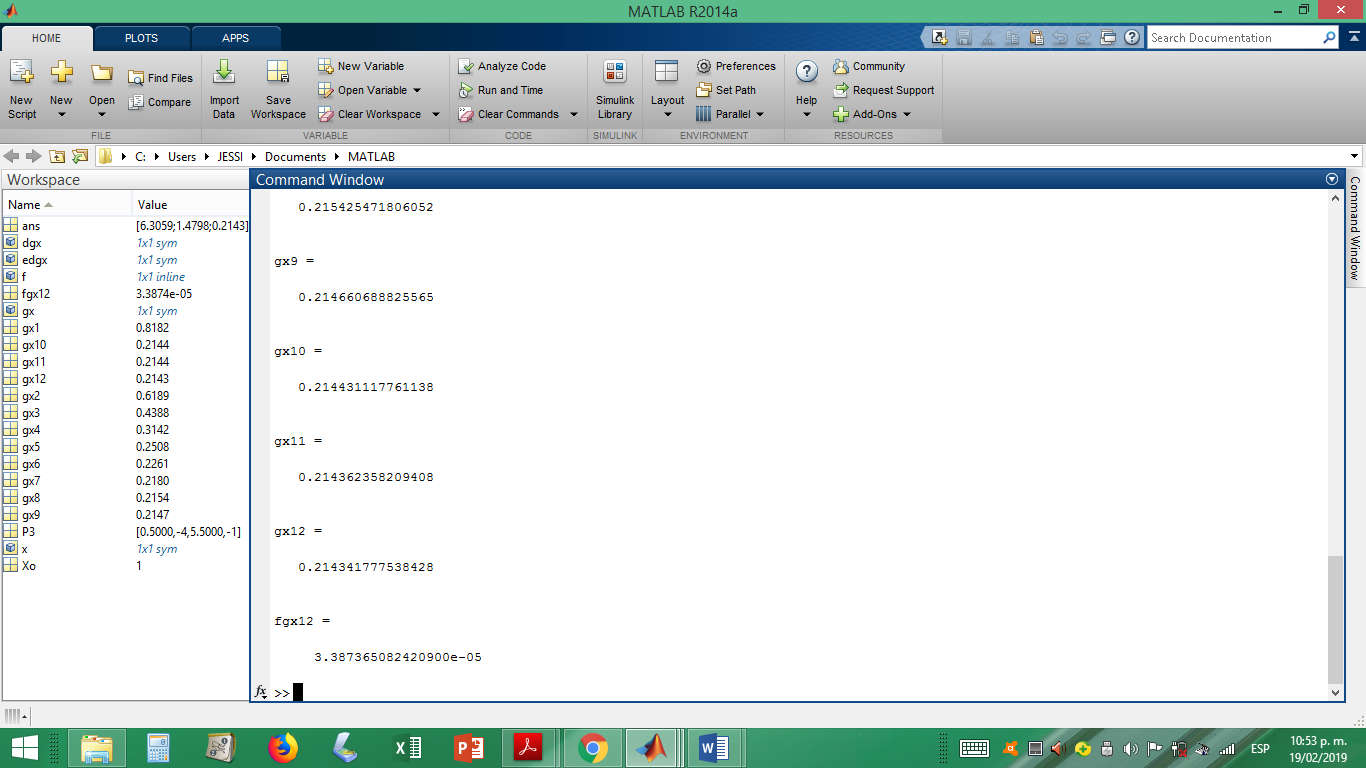
Gráfica



2.- Determine las raíces reales de *f*(*x*) = –1 + 5.5*x* – 4*x*2 + 0.5*x*3. Con una tolerancia de 0.01

**a) Punto fijo**

Respuesta

Una de las 3 raíces de la ecuación, encontrada considerando Xo=1 y 0.01 de tolerancia, es: **0.2143 (error de 3.3873 x10-5)**

Código

clear all

clc

f=inline('-1+5.5\*x-4\*x^2+0.5\*x^3')

ezplot(f)

grid on

syms x

%Primer despeje utilizado

disp('Primer despeje')

Xo=input('Punto inicial, Xo= ')

gx=(4\*x^2+1-0.5\*x^3)/5.5

dgx=diff(gx,1)

edgx=vpa(subs(dgx, Xo),5)

format long

gx1=(4\*Xo^2+1-0.5\*Xo^3)/5.5

gx2=(4\*gx1^2+1-0.5\*gx1^3)/5.5

gx3=(4\*gx2^2+1-0.5\*gx2^3)/5.5

gx4=(4\*gx3^2+1-0.5\*gx3^3)/5.5

gx5=(4\*gx4^2+1-0.5\*gx4^3)/5.5

gx6=(4\*gx5^2+1-0.5\*gx5^3)/5.5

gx7=(4\*gx6^2+1-0.5\*gx6^3)/5.5

gx8=(4\*gx7^2+1-0.5\*gx7^3)/5.5

gx9=(4\*gx8^2+1-0.5\*gx8^3)/5.5

gx10=(4\*gx9^2+1-0.5\*gx9^3)/5.5

gx11=(4\*gx10^2+1-0.5\*gx10^3)/5.5

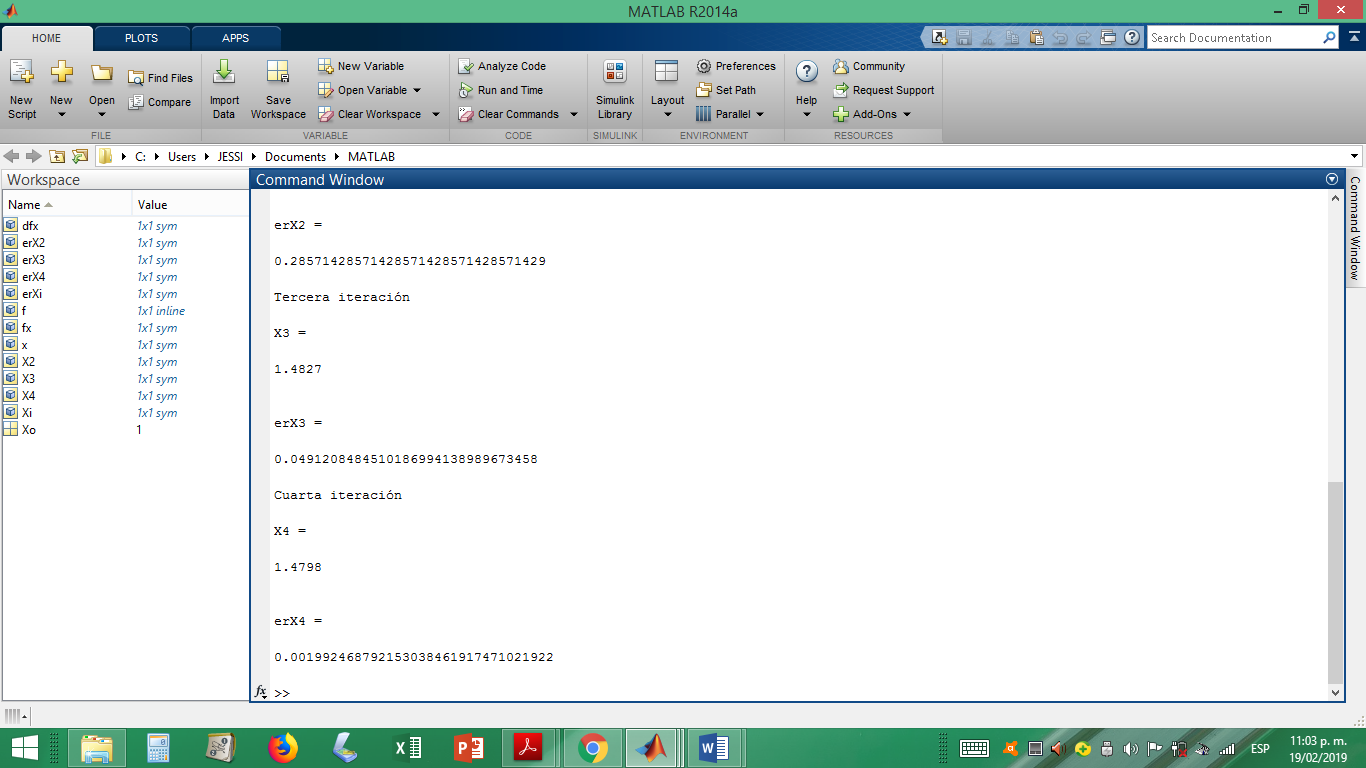
gx12=(4\*gx11^2+1-0.5\*gx11^3)/5.5

fgx12=-1+5.5\*gx12-4\*gx12^2+0.5\*gx12^3

**b) Por Newton Raphson**

Respuesta

Considerando Xo=1 y una tolerancia de 0.01, se obtuvo una raíz (la 2da raíz de la ecuación), igual a= **1.4798 (error de 0.0019).**



Código

clear all

clc

f=inline('-1+5.5\*x-4\*x^2+0.5\*x^3')

ezplot(f)

grid on

Xo=input('Punto inicial Xo= ')

syms x

fx=-1+5.5\*x-4\*x^2+0.5\*x^3

dfx=diff(fx,1)

disp('Primera iteración')

Xi=vpa(Xo-(subs(fx,Xo)/subs(dfx,Xo)),5)

erXi=abs((Xi-Xo)/Xi)

disp('Segunda iteración')

X2=vpa(Xi-(subs(fx,Xi)/subs(dfx,Xi)),5)

erX2=abs((X2-Xi)/X2)

disp('Tercera iteración')

X3=vpa(X2-(subs(fx,X2)/subs(dfx,X2)),5)

erX3=abs((X3-X2)/X3)

disp('Cuarta iteración')

X4=vpa(X3-(subs(fx,X3)/subs(dfx,X3)),5)

erX4=abs((X4-X3)/X4)

* Utilizando el comando “roots” para un polinomio, se encontraron las tres raíces (todas reales) de la ecuación *f*(*x*) = –1 + 5.5*x* – 4*x*2 + 0.5*x*3 , las cuales son:

Código

clear all

clc

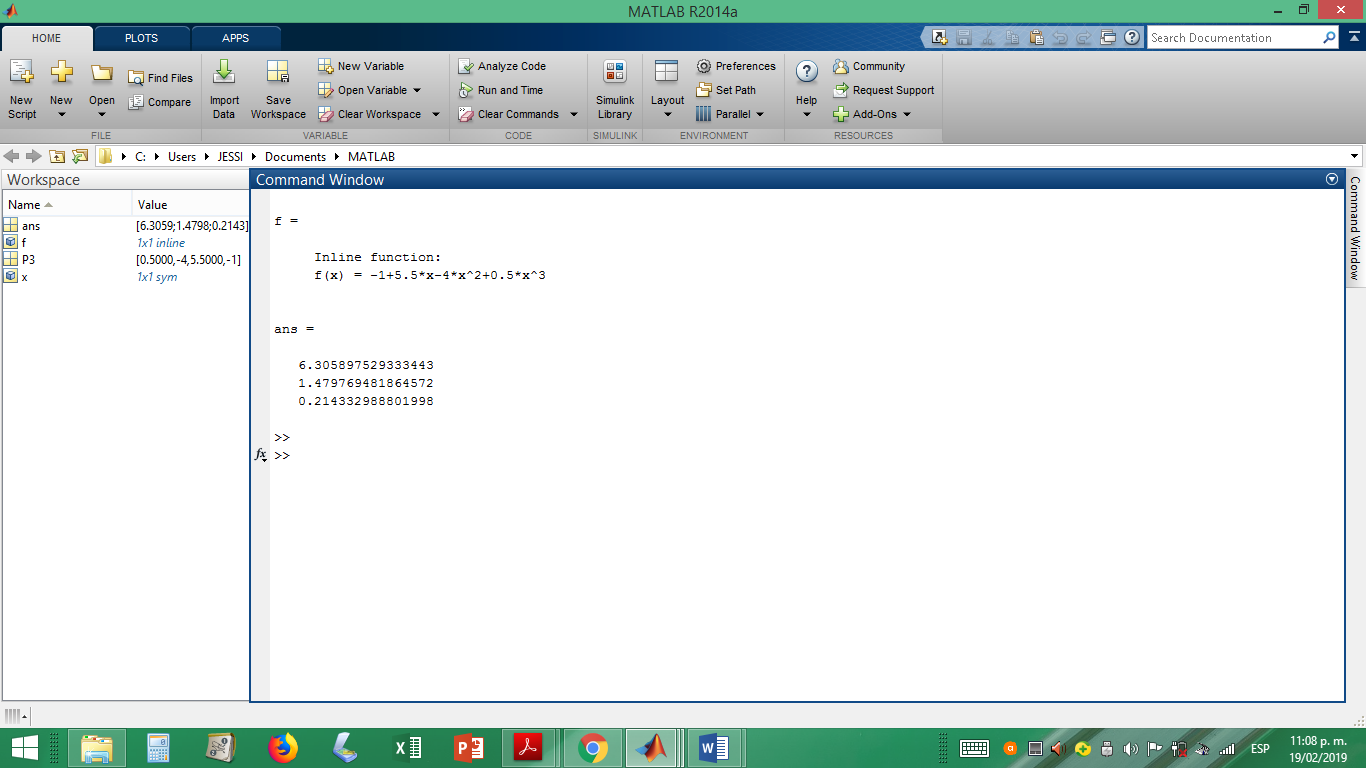
f=inline('-1+5.5\*x-4\*x^2+0.5\*x^3')

ezplot(f)

grid on

P3=[0.5 -4 5.5 -1];

roots(P3)



Por lo tanto conociendo que la tercer raíz es igual a 6.3, podemos encontrarla haciendo uso de igual manera del Método de Newton Raphson, pero con un código diferente, en el que establezcamos a partir de qué valor en adelante, queremos encontrar la raíz. Por ejemplo, estableceremos que busque la raíz a partir de x=5 (r1=5).

Por lo tanto, aplicando un código distinto para el Método de Newton Raphson:

Respuesta: la tercer raíz es **6.3058 (error de 6.2497 x10-5).**

Código

clear all

clc

syms x

f= @(x) -1+5.5\*x-4\*x^2+0.5\*x^3

fplot(f, [-8,8])

grid on

disp('Primera iteración')

%Primera iteracion

r1=5

e1=inf;

disp('Segunda iteración')

%Segunda iteracion

r2=r1-f(r1)/double(subs(diff(f(x)),r1))

e2=abs(r2-r1)

disp('Tercera iteración')

%Tercera iteracion

r3=r2-f(r2)/double(subs(diff(f(x)),r2))

e3=abs(r3-r2)

disp('Cuarta iteración')

%Cuarta iteracion

r4=r3-f(r3)/double(subs(diff(f(x)),r3))

e4=abs(r4-r3)

disp('Quinta iteración')

%Quinta iteracion

r5=r4-f(r4)/double(subs(diff(f(x)),r4))

e5=abs(r5-r4)

disp('Sexta iteración')

%Sexta iteracion

r6=r5-f(r5)/double(subs(diff(f(x)),r5))

e6=abs(r6-r5)

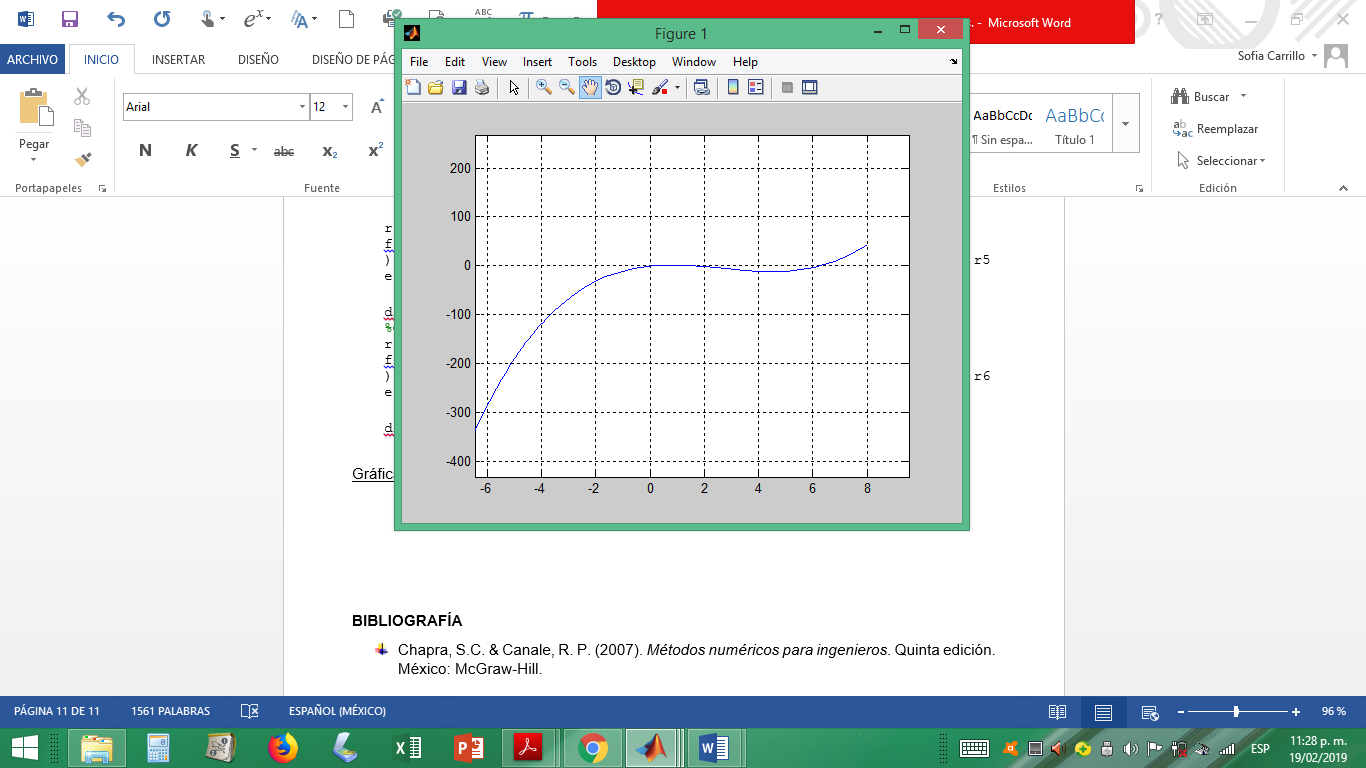
disp('Séptima iteración')

%Sexta iteracion

r7=r6-f(r6)/double(subs(diff(f(x)),r6))

e7=abs(r7-r6)

Gráfica



3. Un proyectil de M=2gm ha sido alcanzado verticalmente al aire y está defendiendo a su velocidad terminal. La velocidad terminal se determina mediante gM=Fdrag.

Donde: g es la gravedad; M es la masa

L a ecuación se puede expresar donde v es la velocidad terminal en m/seg. El primer término del lado derecho representa la fuerza de fricción y el segundo tiempo representa la fuerza de presión. Determina la velocidad terminal. Donde x0 es 30 y la tolerancia es de 0.001.

clear all

clc

syms x

f= @(x) 1.4\*10^-5\*x^1.5+1.15\*10^-5\*x^2-(2\*9.81)/1000

fplot(f,[35,40])

grid on

%Primera iteracion

r1=37

e1=inf;

%Segunda iteracion

r2=r1-f(r1)/double(subs(diff(f(x)),r1))

e2=abs(r2-r1)

%tercera iteracion

r3=r2-f(r2)/double(subs(diff(f(x)),r2))

e3=abs(r3-r2)

%cuarta iteracion

r4=r3-f(r3)/double(subs(diff(f(x)),r3))

e4=abs(r4-r3)

f =

@(x)1.4\*10^-5\*x^1.5+1.15\*10^-5\*x^2-(2\*9.81)/1000

r1 =

37

r2 =

37.7414

e2 =

0.7414

r3 =

37.7346

e3 =

0.0068

r4 =

37.7346

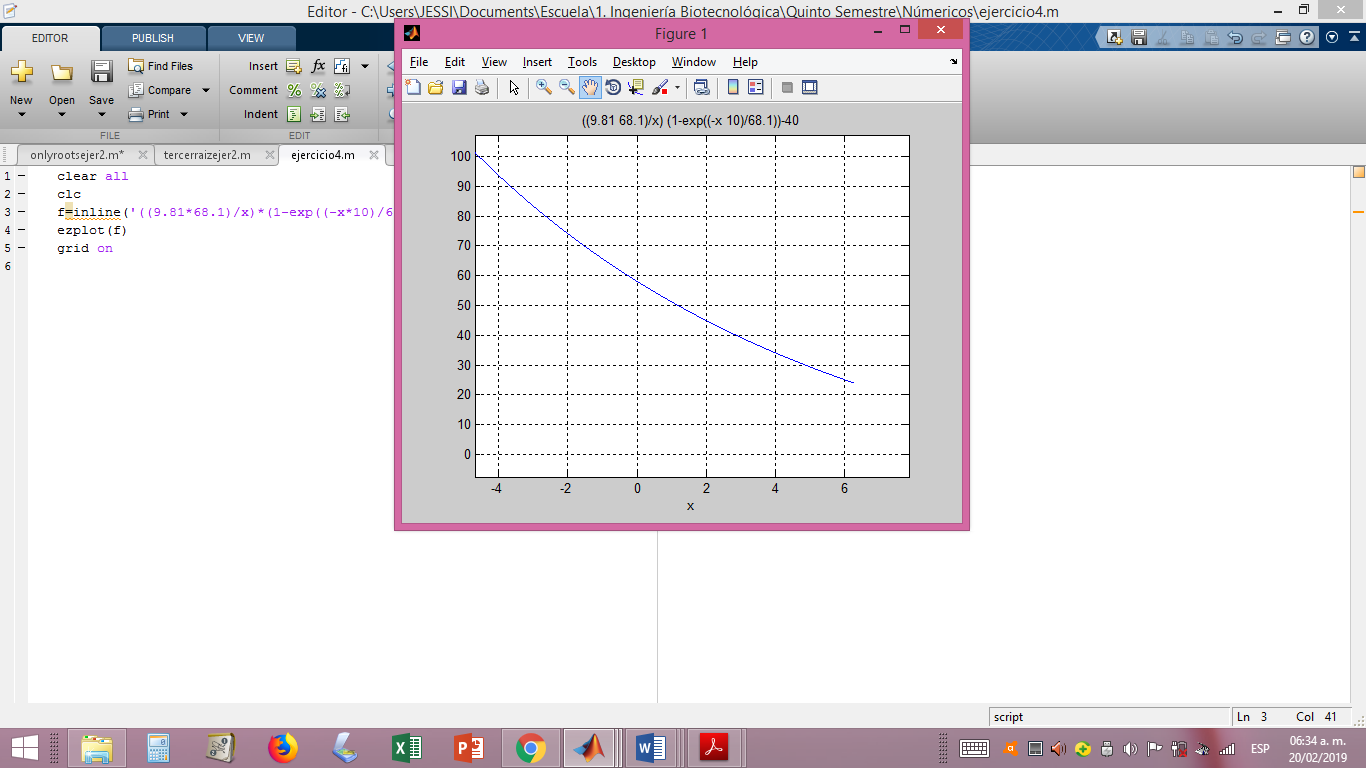
e4 =

5.7541e-07

**Utilice el método grafico para resolver los siguientes ejercicios**

4.-Determinar el coeficiente de arrastre c necesario para que un paracaidista de masa m=68.1Kg tenga una velocidad de 40m/s después de una caída libre de t=10s. Con la ecuación

Respuesta

Ningún coeficiente de arrastre le permitirá al paracaidista caer a dichas condiciones de velocidad, mas y tiempo (ecuación sin raíces).

5.-

f=inline('sin(10\*x)+cos(3\*x)')

ezplot(f,[1,2])

grid on

fzero(f,1.5)

fzero(f,2)

f =

Inline function:

f(x) = sin(10\*x)+cos(3\*x)

f =

Inline function:

f(x) = sin(10\*x)+cos(3\*x)

ans =

0.8458

ans =

1.8125

f =

Inline function:

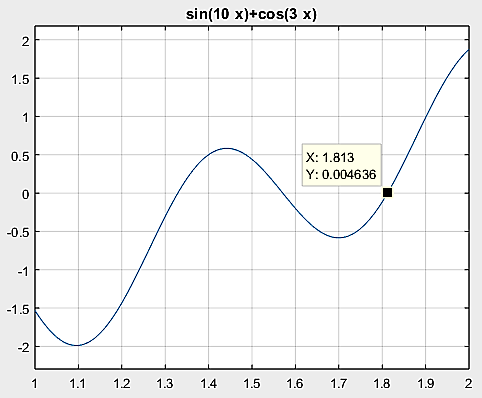
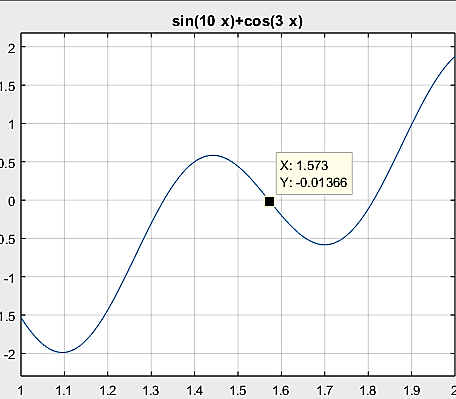
f(x) = sin(10\*x)+cos(3\*x)

ans =

1.5708

ans =

1.8125



***Utilice el método de bisección para resolver los siguientes ejercicios***

6.-Determine las raíces reales de f(x)=-25+82x-90x^2+44x^3-8x^4+0.7x^5, utilice como variables iniciales a=0.5, b=1.0, con una tolerancia de 0.01

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Interacción | a |  | b | f(a) | f(r) | f(b) | error |
| 1 | 0.5 | 0.7500 | 1.0 | -1.4781 | 2.0724 | 3.7000 | 0.2500 |
| 2 | 0.5 | 0.6250 | 0.7500 | -1.4781 | 0.6820 | 2.0724 | 0.1250 |
| 3 | 0.5 | 0.5625 | 0.6250 | -1.4781 | -0.2820 | 0.6820 | 0.0625 |
| 4 | 0.5625 | 0.5938 | 0.6250 | -0.2820 | 0.2265 | 0.6820 | 0.0313 |
| 5 | 0.5625 | 0.5781 | 0.5938 | -0.2820 | -0.0208 | 0.2265 | 0.0156 |
| 6 | 0.5781 | 0.5829 | 0.5938 | -0.0208 | 0.1045 | 0.2265 | 0.0078 |

%Determine las raices reales de f(x)=-25+82x-90x^2+44x^3-8x^4+0.7x^5,

%utilice como variables iniciales a=0.5, b=1.0, con una tolerancia de 0.01

clear all

clc

f=inline('-25+82\*x-90\*x^2+44\*x^3-8\*x^4+0.7\*x^5')

disp('Primera iteracion')

a=0.5

b=1.0

r=(a+b)/2

f(a)

f(r)

f(b)

error=(b-a)/2

disp('Segunda iteracion')

a

b=r

r=(a+b)/2

f(a)

f(r)

f(b)

error=(b-a)/2

disp('Tercera iteracion')

a

b=r

r=(a+b)/2

f(a)

f(r)

f(b)

error=(b-a)/2

disp('Cuarta iteracion')

a=r

b

r=(a+b)/2

f(a)

f(r)

f(b)

error=(b-a)/2

disp('Quinta iteracion')

a

b=r

r=(a+b)/2

f(a)

f(r)

f(b)

error=(b-a)/2

disp('Sexta iteracion')

a=r

b

r=(a+b)/2

f(a)

f(r)

f(b)

error=(b-a)/2

ni=[log(1.0-0.5)-log(0.01)/(2)]

ni=ceil(ni)

eit=(1.0-0.5)/2^6

f =

Inline function:

f(x) = -25+82\*x-90\*x^2+44\*x^3-8\*x^4+0.7\*x^5

Primera iteración

a =

0.5000

b =

1

r =

0.7500

ans =

-1.4781

ans =

2.0724

ans =

3.7000

error =

0.2500

Segunda iteración

a =

0.5000

b =

0.7500

r =

0.6250

ans =

-1.4781

ans =

0.6820

ans =

2.0724

error =

0.1250

Tercera iteración

a =

0.5000

b =

0.6250

r =

0.5625

ans =

-1.4781

ans =

-0.2820

ans =

0.6820

error =

0.0625

Cuarta iteración

a =

0.5625

b =

0.6250

r=

0.5938

ans =

-0.2820

ans =

0.2265

ans =

0.6820

error =

0.0313

Quinta iteración

a =

0.5625

b =

0.5938

r =

0.5781

ans =

-0.2820

ans =

-0.0208

ans =

0.2265

error =

0.0156

Sexta iteración

a =

0.5781

b =

0.5938

r =

0.5859

ans =

-0.0208

ans =

0.1045

ans =

0.2265

error =

0.0078

ni =

1.60

ni =

2

eit =

0.0078

7.-La concentración de saturación de oxígeno disuelto en agua dulce se calcula con la ecuación:

In osf = −139.34411+1.575701\*10^5/Ta−6.642308∗10^7/Ta2 +1.243800∗10^10/Ta3−8.621949∗ 10^11/Ta4

Donde osf = concentración de saturación de oxígeno disuelto en agua dulce a 1 atm (mg/L)

Ta = temperatura absoluta (K)

Ta = T + 273.15, donde T = temperatura (oC)

La saturación disminuye con el incremento de la temperatura. Para aguas naturales comunes en climas templados, la ecuación se usa para determinar que la concentración de oxígeno varía de 14.621 mg/L a 0oC a 6.413 mg/L a 40oC.

a) Si los valores iniciales son de 0 y 40oC, ¿cuántas iteraciones se requerirían para determinar la temperatura con un error absoluto de 0.05oC.

b) Considere osf = 8, 10 y 12 mg/L para determinar la temperatura

f=inline('-139.34411+1.575701e5/x-6.642308e7/x^2+1.243800e10/x^3-8.621949e11/x^4-log(8)')

fplot(f,[260,310])

grid on

pause

a=input('teclee limite inferior');

b=input('teclee limite superior');

tol=input('teclee la toleerancia');

%

%

% %raiz y error

%

r=(a+b)/2;

e=(b-a)/2;

%

% %se hace el ciclo

while e>tol

if f(r)~=0

if f(a)\*f(r)<0

b=r;

else

a=r;

end

r=(a+b)/2;

e=(b-a)/2;

else

e=0;

end

end

disp(r)

disp(e)

f =

Inline function:

f(x) = -139.34411+1.575701e5/x-6.642308e7/x^2+1.243800e10/x^3-8.621949e11/x^4-log(8)

f =

Inline function:

f(x) = -139.34411+1.575701e5/x-6.642308e7/x^2+1.243800e10/x^3-8.621949e11/x^4-log(8)

f =

Inline function:

f(x) = -139.34411+1.575701e5/x-6.642308e7/x^2+1.243800e10/x^3-8.621949e11/x^4-log(8)

f =

Inline function:

f(x) = -139.34411+1.575701e5/x-6.642308e7/x^2+1.243800e10/x^3-8.621949e11/x^4-log(8)

f =

Inline function:

f(x) = -139.34411+1.575701e5/x-6.642308e7/x^2+1.243800e10/x^3-8.621949e11/x^4-log(8)

f =

Inline function:

f(x) = -139.34411+1.575701e5/x-6.642308e7/x^2+1.243800e10/x^3-8.621949e11/x^4-log(8)

f =

Inline function:

f(x) = -139.34411+1.575701e5/x-6.642308e7/x^2+1.243800e10/x^3-8.621949e11/x^4-log(8)

f =

Inline function:

f(x) = -139.34411+1.575701e5/x-6.642308e7/x^2+1.243800e10/x^3-8.621949e11/x^4-log(8)

Se tecleo limite inferior0+273.15

Se tecleo limite superior40+273.15

Se tecleo la toleerancia0.05

299.9078

0.0391

f =

Inline function:

f(x) = -139.34411+1.575701e5/x-6.642308e7/x^2+1.243800e10/x^3-8.621949e11/x^4-log(8)

Se tecleo limite inferior0+273.15

Se tecleo limite superior40+273.15

Se tecleo la toleerancia0.05

299.9078

0.0391

f=inline('-139.34411+1.575701e5/x-6.642308e7/x^2+1.243800e10/x^3-8.621949e11/x^4-log(10)')

fplot(f,[260,310])

grid on

pause

a=input('teclee limite inferior');

b=input('teclee limite superior');

tol=input('teclee la toleerancia');

%

%

% %raiz y error

%

r=(a+b)/2;

e=(b-a)/2;

%

% %se hace el ciclo

while e>tol

if f(r)~=0

if f(a)\*f(r)<0

b=r;

else

a=r;

end

r=(a+b)/2;

e=(b-a)/2;

else

e=0;

end

end

disp(r)

disp(e)

f =

Inline function:

f(x) = -139.34411+1.575701e5/x-6.642308e7/x^2+1.243800e10/x^3-8.621949e11/x^4-log(10)

teclee limite inferior273.15

teclee limite superior40+273.15

teclee la toleerancia0.05

288.5016

0.0391

f=inline('-139.34411+1.575701e5/x-6.642308e7/x^2+1.243800e10/x^3-8.621949e11/x^4-log(12)')

fplot(f,[260,310])

grid on

pause

a=input('teclee limite inferior');

b=input('teclee limite superior');

tol=input('teclee la toleerancia');

%

%

% %raiz y error

%

r=(a+b)/2;

e=(b-a)/2;

%

% %se hace el ciclo

while e>tol

if f(r)~=0

if f(a)\*f(r)<0

b=r;

else

a=r;

end

r=(a+b)/2;

e=(b-a)/2;

else

e=0;

end

end

disp(r)

disp(e)

f =

Inline function:

f(x) = -139.34411+1.575701e5/x-6.642308e7/x^2+1.243800e10/x^3-8.621949e11/x^4-log(12)

teclee limite inferior0+273.15

teclee limite superior40+273.15

teclee la toleerancia0.05

280.6109

0.0391

**Aplicación a la cinemática**. Una partícula se mueve con velocidad (m/s) dada en función del tiempo por medio de la función:

Utilizando el método de Newton Raphson aproxima el tiempo en el que la partícula alcanza una velocidad de 1m/s a partir del reposo

1. Plantea la ecuación a resolver y la fórmula de Newton Raphson para este caso
2. Calcula hasta la quinta iteración con iteración inicial igual a 3
3. ¿Se puede considerar el valor inicial de iteración igual a 0?

No, porque cuando se evalúa saldría igual a 0 ya que en ese tiempo no hay movimiento alguno

1. Con la quinta y cuarta iteración calcula el error relativo de la aproximación para la ultima iteración, indicando el número de cifras significativas correctas según el error relativo.

%Newton raphson4

clear all

clc

f=inline('x^3-2\*x^2-1')

ezplot(f)

grid on

xo=input('Punto inicial, xo:');

syms x

fx=x^3-2\*x^2-1

dfx=diff(fx,1)

%Primer iteración

x1=xo-(subs(fx,xo)/subs(dfx,xo))

er=abs((x1-xo)/x1)

%Segunda iteracion

x2=x1-(subs(fx,x1)/subs(dfx,x1))

er=vpa(abs((x2-x1)/x2),4)

%Tercer iteracion

x3=x2-(subs(fx,x2)/subs(dfx,x2))

er=vpa(abs((x3-x2)/x3),4)

%Cuarta iteracion

x4=x3-(subs(fx,x3)/subs(dfx,x3))

er=vpa(abs((x4-x3)/x4),4)

%Quinta iteracion

x5=x4-(subs(fx,x4)/subs(dfx,x4))

er=vpa(abs((x5-x4)/x5),4)

f =

Inline function:

f(x) = x^3-2\*x^2-1

Punto inicial, xo:3

fx =

x^3 - 2\*x^2 - 1

dfx =

3\*x^2 - 4\*x

x1 =

37/15

er =

8/37

x2 =

63611/28305

er =

0.09759

x3 =

308398726847077/139742932569615

er =

0.01832

x4 =

34810541371451247388527730195058075538539771/15783007581753187653924166373114041594098705

er =

0.0006028

x5 =

50045733210820400774041815030104996649260978859465366710116057296831799158594241849499728950059846490796910508201656085934339396837/22690617906185990383316447952168338228806212048490499574002366995563844385058416102433217261297843093405911516989799340211526921615

er =

6.415e-7

**CONCLUSIONES.**

El método de bisección es un método de búsqueda de raíces el cual nos permitió, con ayuda de un método gráfico, la identificación de los ceros de una ecuación haciendo que el intervalo donde se encontrara fuese cada vez más pequeño partiendo a la mitad.

Resultó medianamente laborioso, puesto que se debe llevar un registro de los valores y de los signos de las evaluaciones para bisectar correctamente, de no ser así, el nuevo valor guardado en el vector arruinaría todo el proceso consecuente.

Resultó de ayuda el método grafico para reducir el intervalo y por tanto reducir el número de iteraciones, las cuales también pueden ser calculadas con ayuda de la fórmula correspondiente.

Sin embargo, consideramos que los métodos abiertos nos permiten obtener las raíces con una mayor precisión. Aunque no siempre es fácil encontrar la convergencia, cuando sí se logra, es mucho más fácilmente que en los métodos cerrados.

**BIBLIOGRAFÍA**

* Chapra, S.C. & Canale, R. P. (2007). *Métodos numéricos para ingenieros*. Quinta edición. México: McGraw-Hill.