Informe laboratorio análisis numérico
Práctica No. 2

Brayan Barajas
Escuela de ingeniería de sistemas e informática
Universidad Industrial de Santander
13 de octubre de 2019

1 Introducción

En esta práctica de laboratorio se pretende estudiar de forma práctica el método de bisección mediante el uso de MATLAB, en el cual se programará el código respectivo para ejecutar y comprobar las iteraciones del método de bisección aplicado a ciertas funciones en determinados intervalos, evidenciando su convergencia y las condiciones para que esta se de.
Además, se comparará el error teórico con el práctico por medio de una gráfica y se procederá a resolver un problema aplicado haciendo uso de este método.

2 Desarrollo

IMPLEMENTING

2.1. Encontrar el intervalo de la función en el que se puede aplicar el método de bisección.

Para hallar el intervalo, se creó la función my_finding_interval_brayan_barajas.m la cual tomaba dos puntos, uno partiendo en cero (debido a que era requerido) y el otro aleatorio para evaluar la función y comprobar si tenían signos diferentes, de no ser así, cambiaba los número aleatoriamente hasta encontrar dos valores que cumplieran esta condición. Al encontrarlos, ya es posible afirmar que se ha encontrado el intervalo para aplicar el método de bisección a la función dada.
Con esto, se devuelven los valores obtenidos como intervalo.
Esta función se probó y ejecutó con el código run_2a_brayan_barajas.m

2.2. Implementar el método de bisección.

Para implementar el método de bisección, se creó la función my_bisection_function_brayan_barajas.m la cual tenía como parámetros la función respectiva, los puntos del intervalo a evaluar, el número de iteraciones y el error solicitado. Esta función se basaba en un bucle que obtenía el punto medio entre cada extremo y evaluaba con qué extremo anterior las imágenes respectivas tenían diferente signo para que

fuera el nuevo intervalo; este proceso se repite ya sea hasta acabar el número de repeticiones o llegar al error deseado (que se va calculando en cada iteración).
Al final, la función retorna el resultado de la raíz en el eje x tal que f(x)=0.
Esta función se probó y ejecutó con el código run_2b_brayan_barajas.m en el que se asignó una función y se halló el intervalo con la función respectiva creada anteriormente.

2.3. Utilizar la función anterior para encontrar la raíz de $f(x) = (x-8) * (x-3)^2$. Además, comparar el número de iteraciones teóricas y prácticas en una gráfica de acuerdo al error dado.

Para hacer esto, se creó el archivo run_2c_brayan_barajas.m en el que se definió la función dada, se halló el intervalo con la primera función, se guardaron los errores en un vector para posteriormente recorrerlo e ir ejecutando la función del método de bisección, a la cual se le hizo una pequeña alteración para contar la cantidad de iteraciones realizadas por medio de la variable cont que al final se devuelve como iterf, la cual representa las iteraciones prácticas. Este dato se almacenó en otro vector con el nombre de iterp, así como se calculó el número de iteraciones teóricas de acuerdo al error correspondiente y se guardó en itert.

Estos vectores son utilizados posteriormente para realizar la gráfica solicitada de número de iteraciones, tanto teóricas como prácticas, de acuerdo al error mediante la función plot.

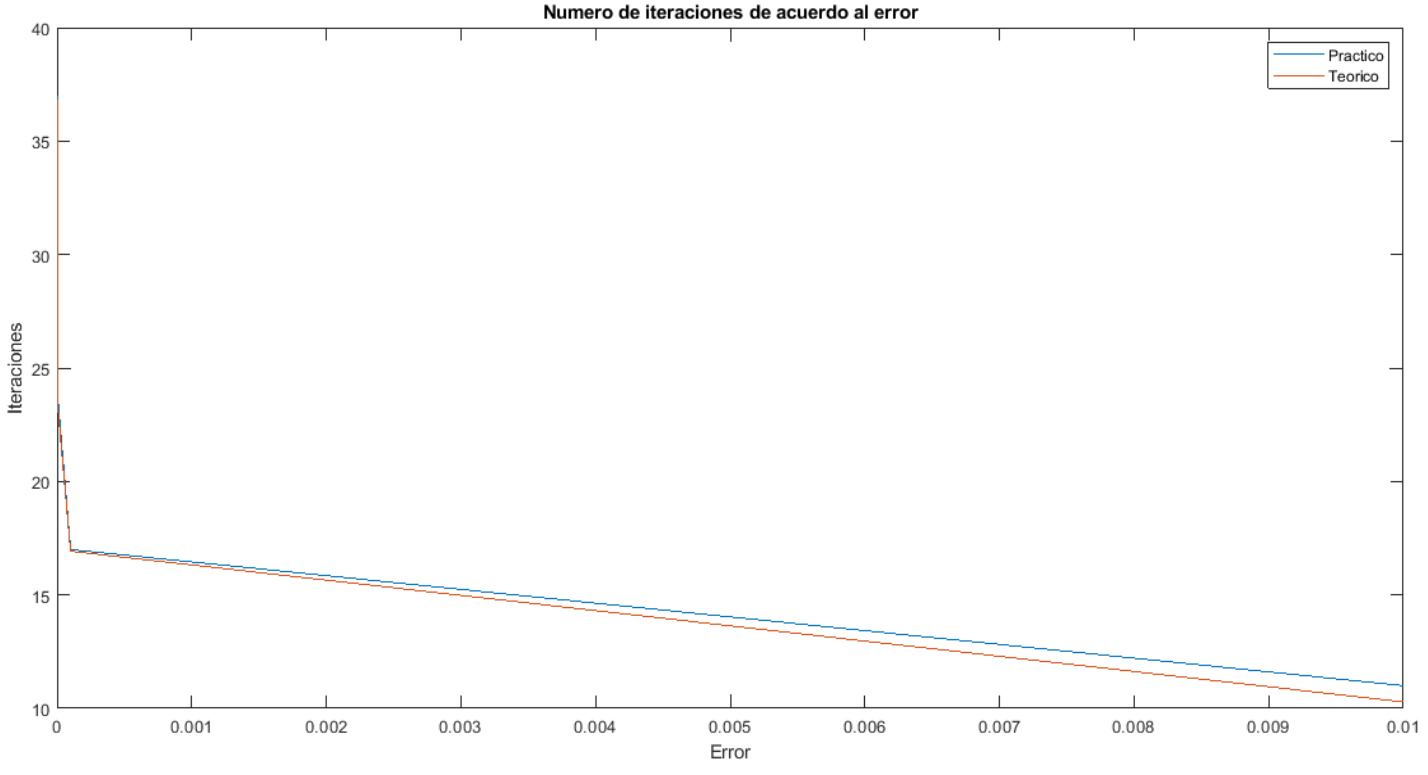

Figura 1. Gráfica iteraciones vs error

De la anterior gráfica es posible evidenciar que el valor obtenido de las iteraciones teóricas normalmente es algo menor al práctico; pero es de importancia resaltar que a menor error, el valor teórico se acerca más al valor práctico; mientras que al incrementar el error, estos valores empiezan a alejarsen, perdiendo precisión.

2.4. Crear una función que permita visualizar de forma gráfica el funcionamiento del método de bisección.

A partir de la función que ya se tenía anteriormente, se creó una nueva nombrada my_visual_bisection_function_brayan_barajas.m la cual varía poco con respecto a la anterior. La gran diferencia es que ahora se grafica la función a la que se le aplicará el método, así como, a cada iteración, se va marcando el punto obtenido hasta llegar al último que sería la raíz obtenida por medio del método de bisección.
Esta función fue probada en el archivo run_2d_brayan_barajas.m en el cual se le asignó un valor, el intervalo, las iteraciones y el error para poder ejecutar la función y poder visualizar el resultado, que se muestra a continuación. Para este ejemplo se utilizó $f(x) = x^3 - 7$

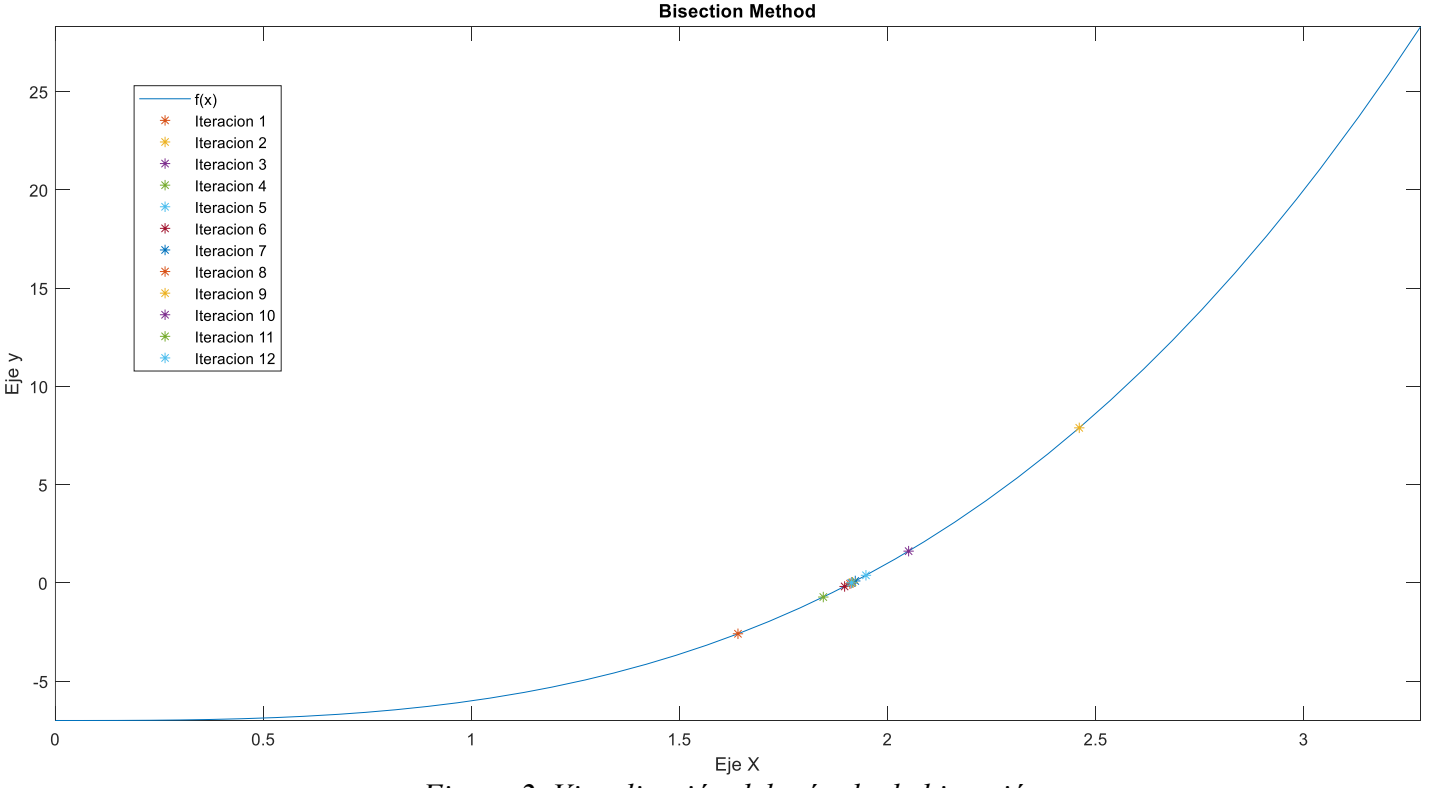

Figura 2. Visualización del método de bisección.

Con la gráfica anterior, se puede observar el comportamiento del método, que va diviendo los intervalos por mitad hasta llegar a la raíz; con esto, se evidencia gráficamente la convergencia del método desde que se cumpla la condición del intervalo.

INTERPRETING

2.5. A partir del problema enunciado, plantear una función de la forma f(x)=0 para obetener la tasa de crecimiento.

Teniendo en cuenta los datos, se tiene que

$$N_0 = 1500, \qquad \varphi = 475$$

$$N(1) = 2264 = 1500 * e^{\lambda} + 475 * \frac{e^{\lambda} - 1}{\lambda}$$

Al despejar lo anterior, se puede hallar f(λ):

$$f(\lambda) = 1500 * e^{\lambda} + 475 * \frac{e^{\lambda} - 1}{\lambda} - 2264 = 0$$

2.6. Hacer un script que resuelva la ecuación anterior utilizando la función 2. Además, graficar.

En el script run_2e_brayan_barajas.m se ejecutó la función de my_bisection_function_brayan_barajas para resolver la función anterior, dando como resultado 0.1544, que es la raíz en el eje y, por lo que esta sería la tasa de crecimiento para el problema planteado. Esto se puede verificar en la gráfica generada:

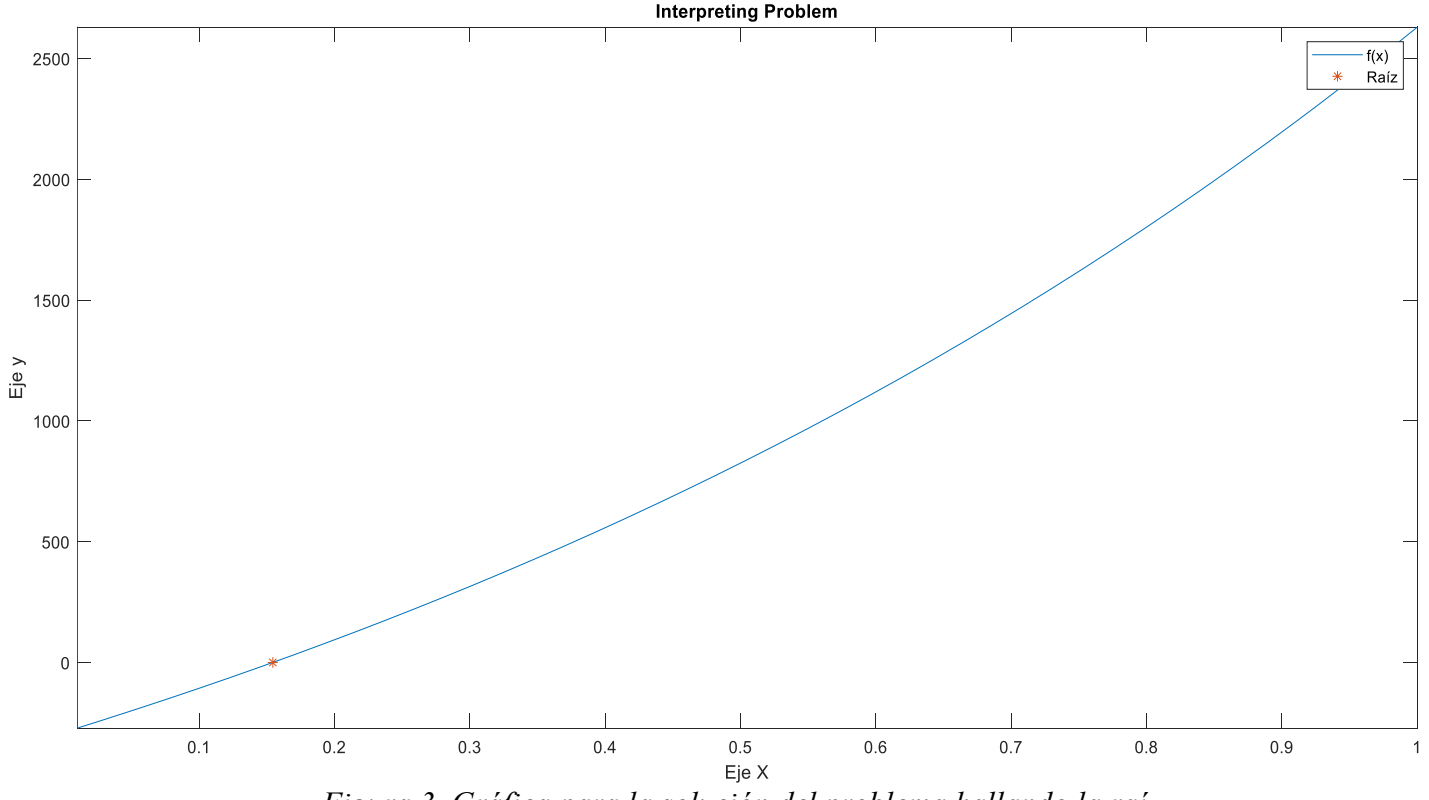

Figura 3. Gráfica para la solución del problema hallando la raíz.

3 Anexos

run_2a_brayan_barajas.m

```
f=@(x) x.^3-7;
[a,b]=my_finding_interval_brayan_barajas(f);
```

my_finding_interval_brayan_barajas.m

```
function [a,b]=my_finding_interval_brayan_barajas(f)
    a=0;
    b=rand(1)*30-10;
    while sign(f(a))==sign(f(b))
        a=rand(1)*30-10;
        b=rand(1)*30-10;
    end
    p=[a b];
    p=sort(p);
    disp('El intervalo es: ')
    disp(p)
end
disp('Errores relativos:')
disp(errorrel)
```

run_2b_brayan_barajas.m

```
f=@(x) x.^3-7;
[a,b]=my_finding_interval_brayan_barajas(f);
my_bisection_function_brayan_barajas(f,a,b,10,0.001);
```

my_bisection_function_brayan_barajas.m

```
function
[root,iterf]=my_bisection_function_brayan_barajas(f,a,b,iter,error)
c=(a+b)/2;

    if sign(f(a))==sign(f(c))
        a=c;
    else
        b=c;
    end
err=abs(b-a);
cont=2;
while cont<=iter || err>error
    cn=(a+b)/2;
    err=abs(cn-c);
    c=cn;
    if sign(f(a))==sign(f(c))
        a=c;
    else
        b=c;
    end
    iterf=cont;
    cont=cont+1;
end
root=c;

disp('La raíz es: ')
disp(root)
end
```

run_2c_brayan_barajas.m

```
f=@(x) (x-8)*(x-3).^2;
[a,b]=my_finding_interval_brayan_barajas(f);
error=[1e-2,1e-4,1e-6,1e-8,1e-10];
iterp=[];
itert=[];
for i = error
disp('Con error de: ')
disp(i)
[root,iterf]=my_bisection_function_brayan_barajas(f,a,b,0,i);
iterp=[iterp, iterf];
itert=[itert, log2(abs((b-a)/i))];
end
figure,plot(error, iterp),title("Numero de iteraciones de acuerdo al
error"),xlabel('Error'),ylabel('Iteraciones')
hold on
plot(error,itert)
legend('Practico','Teorico')
```

my_visual_bisection_function_brayan_barajas.m

```
function
[root,iterf]=my_visual_bisection_function_brayan_barajas(f,a,b,iter,error)
fplot(f,sort([a,b])),title("Bisection Method"),legend('f(x)'),xlabel('Eje
X'),ylabel('Eje y')
c=(a+b)/2;
hold on
plot(c,f(c), '*','DisplayName','Iteracion 1')
    if sign(f(a))==sign(f(c))
        a=c;
    else
        b=c;
    end
err=abs(b-a);
cont=2;
while cont<=iter || err>error
    cn=(a+b)/2;
    err=abs(cn-c);
    c=cn;
    plot(c,f(c), '*','DisplayName',['Iteracion ' num2str(cont)])
    if f(c)==0, break, end
    if sign(f(a))==sign(f(c))
        a=c;
    else
        b=c;
    end
    iterf=cont;
    cont=cont+1;
end
root=c;
disp('La raíz es: ')
disp(root)
end
```

run_2d_brayan_barajas.m

```
f=@(x) x.^3-7;
[a,b]=my_finding_interval_brayan_barajas(f);
my_visual_bisection_function_brayan_barajas(f,a,b,10,0.001)
```

run_2e_brayan_barajas.m

```
f=@(x) 1500*exp(x)+475*((exp(x)-1)/x)-2264;
[root,iterf]=my_bisection_function_brayan_barajas(f,0.01,1,0,1e-10);
figure,fplot(f,[0.01,1]),title("Interpreting
Problem"),legend('f(x)'),xlabel('Eje X'),ylabel('Eje y')
hold on
plot(root,f(root),"*",'DisplayName','Raíz')
```