Informe laboratorio de análisis numérico
Práctica No. 2

Kevin Javier Lozano Galvis
Escuela de ingeniería de sistemas e informática
Universidad Industrial de Santander

23 de mayo de 2019

1. Introducción

En esta práctica de laboratorio estudiaremos el método de bisección en el sistema de cómputo numérico MATLAB; en el cual haremos el código de programación correspondiente para obtener, verificar y graficar diferentes funciones que cumplen con el método de bisección en determinados intervalos.

Podremos observar y analizar las raíces, iteraciones teóricas y prácticas e intervalos para ciertas funciones. Para dar éxito a la práctica de laboratorio contamos con la documentación adecuada de MATLAB y ayuda del docente de laboratorio en el aula de clase.

2. Desarrollo

Para el desarrollo de esta práctica de laboratorio vamos a dividir los pasos realizados, en los cuales se desarrolló el código.

2.1. Búsqueda del intervalo

Para la búsqueda de un intervalo se definió la función `f(x)= x^2+x-1;` que pasará cómo parámetro en la función `my_finding_interval_kevin_lozano(f,a,b),` a y b son los intervalos, los cuales se verificarán para ver si evaluados en la función y multiplicando estos resultados, da un número menor a cero, de lo contrario está definido un while el cual buscará valores aleatorios de intervalos (entre 10 y -10) que cumplan el criterio de ser evaluados en la función y multiplicar dichos resultados para obtener el valor negativo.

Luego de cumplir estos pasos, la función imprimirá los resultados que fueron digitados o encontrados aleatoriamente.

2.2. Búsqueda de la raíz de la función

En la búsqueda de la raíz se trabajó con la función definida anteriormente y se pasó como parámetro a la función my_bisection_interval_kevin_lozano(f,a,b,iter), en la cual a y b siguen siendo los intervalos, con los mismos criterios, y el parámetro 'iter' es la variable que guardará el criterio de parada (donde se detendrá).

Se comenzó verificando que se cumplan los intervalos para dicha función según el método de bisección, seguido a esto se define la variable 'c' en la cual se guardará el valor de la suma y división en dos del intervalo encontrado, así mismo se define la variable 'ni' que nos guardará el número de las veces de iteración. Luego, un while me comprobará que mientras el valor de 'c' evaluado en la función y con el respectivo valor absoluto, sea mayor o igual a la variable 'iter' definida en los parámetros, se seguirán dividiendo los intervalos hasta llegar a encontrar un valor exacto de la raíz. El programa me imprime la raíz encontrada y el número de iteraciones que necesitó para hallarla.

2.3. Búsqueda de la raíz y gráfica con diferente función

En esta búsqueda utilizaremos la función `f(x)=(x-8) *(x-3) ^2;` y se pasó como parámetro a la función `my_plot_iter_kevin_lozano(f,a,b)`, en la cual a y b son los intervalos y se hará el mismo procedimiento descrito anteriormente, la diferencia es que en esta parte hallamos las iteraciones teóricas y graficamos cada épsilon con su respectivo valor de iteración teórico e iteración práctica.

Podemos observar que en las épsilon de menor valor como es el caso de 1e-10, la raíz da más puntual que en los otros, esto se debe a que el valor en el que se debe detener es mucho menor, teniendo en cuenta que épsilon a la menos 10 es 0.00004539, se va a detener cuando haga más divisiones entre los intervalos, por lo tanto, va a encontrar la raíz exacta, que para este caso la raíz de la función es 8.

También podemos ver que a una menor épsilon el número de iteraciones es mayor, esto se debe a la misma explicación daba anteriormente, de que tiene que dividir muchas más veces los resultados de los intervalos para llegar al valor esperado de la raíz.

La gráfica mostrada a continuación nos representa con línea punteada de color rojo la gráfica de épsilon respecto a las iteraciones teóricas y los puntos de color azul nos representa la gráfica de épsilon respecto a las iteraciones prácticas, todos estos valores de la gráfica son respecto a la función `f(x)=(x-8) *(x-3) ^2` en los intervalos de [-10,10]

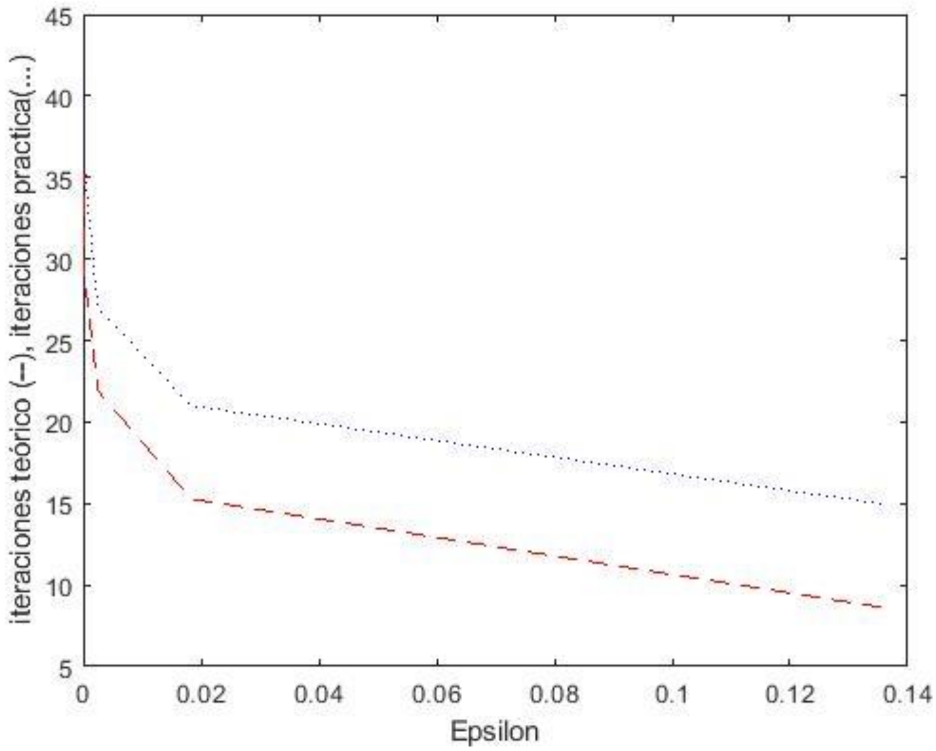

Figura 1. Gráfica de épsilon respecto a las iteraciones

2.4. Gráfica de la función para el método de bisección

En esta sección se gráfica la función que nos dan `f(x)=(x-8) *(x-3) ^2` y observamos que la raíz efectivamente es 8. Esto lo hacemos con el código que ya teníamos para hallar la raíz de una función.

Agregamos la línea de código `fplot(f,[-11,11])` para graficar la función en el intervalo en x de -11 a 11 y ver que en 8 está la raíz, y la línea de código `plot(root,f(root),'*')` para graficar el punto de coordenadas de la raíz que en este caso es (8,0).

Podemos observar la siguiente gráfica que describe lo dicho anteriormente:

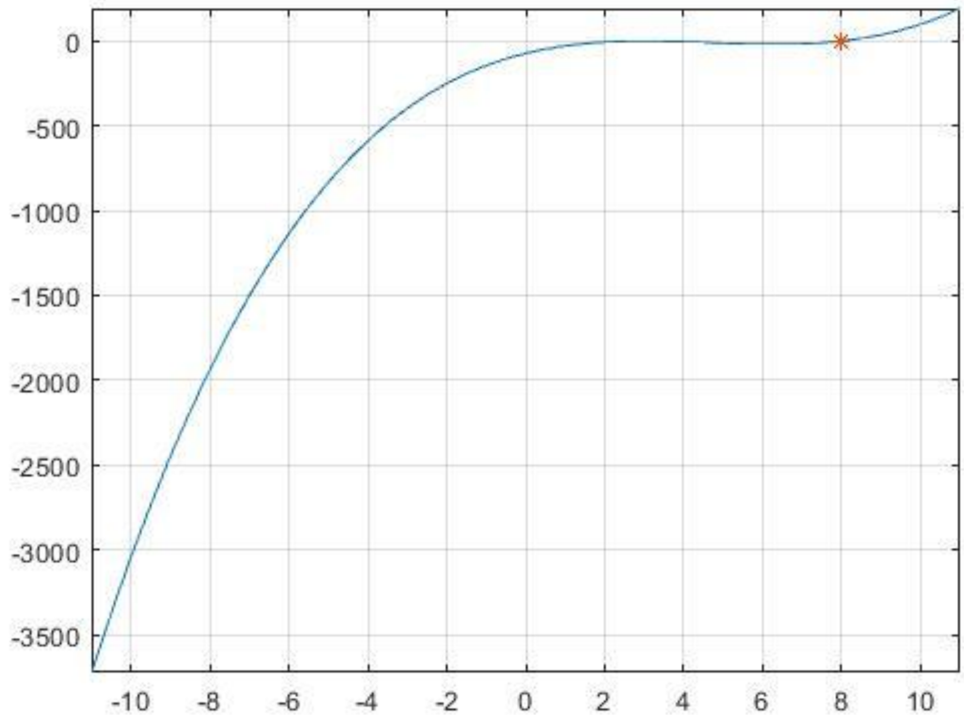

Figura 2. Gráfica del método de bisección mostrando la raíz

3. Anexos

A continuación, se mostrarán los editores en los cuales se implementó el laboratorio.

run_2a_KevinLozano.m

```
%Definimos la función de la cual hallaremos el intervalo.
f=@(x) x.^2+x-1;

%Ingresamos la función definida con el intervalo [0,0], que es un intervalo
%que no cumple con la regla. Entonces el programa busca el intervalo que al
%evaluarlo en la función me cumpla que la multiplicación es negativa.
my_finding_interval_kevin_lozano(f,0,0)
```

my_finding_interval_kevin_lozano.m

```
function [a,b]=my_finding_interval_kevin_lozano(f,a,b)

%Si se establecen los intervalos de (a) y (b) adecuadamente se cumple,
%por ejemplo si se coloca a=0 y b=2, no hay necesidad de entrar al while
%porque se cumpliría que la multiplicación de los intervalos evaluados en
%la función es menor a 0.
%Si no se cumple lo anterior, se entra al while,donde se van seleccionando
%números aleatorios en el rango de [-10,10] hasta que se encuentren los
%intervalos que cumplan la regla de que debe dar menor a 0.

    while sign(f(a))*sign(f(b))>0
    a=rand(1)*20-10;
    b=rand(1)*20-10;
    end
    disp('Se encontraron los intervalos: ');
    int = [a b];
    disp(int)
end
```

run_2b_KevinLozano.m

```
f=@(x) x.^2+x-1;
a=0;
b=3;
iter=0.0025;
my_bisection_interval_kevin_lozano(f,a,b,iter)
```

my_bisection_interval_kevin_lozano.m

```
function [root]=my_bisection_interval_kevin_lozano(f,a,b,iter)

%Primero volvemos a verificar que se cumplan los intervalos ingresados, de
%lo contrario, hallamos nuevos intervalos.
 while sign(f(a))*sign(f(b))>0
    a=rand(1)*20-10;
    b=rand(1)*20-10;
 end
    disp('Se encontraron los intervalos: ');
    int = [a b];
    disp(int)

% Hallamos la raíz
    c=(a+b)/2;
    ni=0;
    %iter es la variable que me va a guardar el criterio donde se
    %detiene.
    while (abs(f(c))>=iter)
        c = (a+b)/2;
        fc = f(c);
        ni=ni+1;
        if fc == 0
            break
        end
        if sign(f(a))*sign(f(c))<0
            b=c;
        else
            a=c;
        end
    end
    root=c;
    disp('la raiz es:')
    disp(root)
    disp('El numero de iteraciones es:')
    disp(ni)
end
```

run_2c_KevinLozano.m

```
f=@(x) (x-8)*(x-3).^2;
a=10;
b=-10;
%Si los valores del intervalo no sirven, el programa de manera aleatoria
%los busca.
my_plot_iter_kevin_lozano(f,a,b)
```

my_plot_iter_kevin_lozano.m

```
function [root]=my_plot_iter_kevin_lozano(f,a,b)
%Primero volvemos a verificar que se cumplan los intervalos ingresados, de
%lo contrario, hallamos nuevos intervalos.
 while sign(f(a))*sign(f(b))>0
    a=rand(1)*20-10;
    b=rand(1)*20-10;
 end
    disp('Se encontraron los intervalos: ');
    int = [a b];
    disp(int)
% Hallamos la raíz
    %iter es la variable que me va a guardar el criterio donde se
    %detiene.
    ni=0;
    e=1e-10;
    nt = ((log(abs(b-a)))-log(abs(e))/log(2))-1;
    c=(a+b)/2;

    while (abs(f(c))>=e)
        c = (a+b)/2;
        fc = f(c);
        ni=ni+1;
        if fc == 0
            break
        end
        if sign(f(a))*sign(f(c))<0
            b=c;
        else
            a=c;
        end
    end
    root=c;
    disp('la raiz es:')
    disp(root)
    disp('El numero de iteraciones es:')
    disp(ni)
    disp('El numero de iteraciones teórico es:')
    disp(nt)

   %para el intervalo 10 y -10
   Vtol=[0.135335 0.018315 0.002478 0.0003354 0.00004539];
   Nt= [8.6396 15.2834 21.9273 28.5712 35.2150];
   Vne=[15 21 27 35 41];
   plot(Vtol,Nt,'--r')
   xlabel('Epsilon')
   hold on
   plot (Vtol, Vne,':b')
   ylabel('iteraciones teórico (--), iteraciones practica(…)')

end
```

run2d_KevinLozano.m

```
f=@(x) (x-8)*(x-3).^2;
a=-10;
b=10;
iter=0.001;
my_visual_bisection_function_Kevin_Lozano(f,a,b,iter)
```

my_visual_bisection_function_Kevin_Lozano.m

```
function [P]=my_visual_bisection_function_Kevin_Lozano(f,a,b,iter)

%Primero volvemos a verificar que se cumplan los intervalos ingresados, de
%lo contrario, hallamos nuevos intervalos.
 while sign(f(a))*sign(f(b))>0
    a=rand(1)*20-10;
    b=rand(1)*20-10;
 end
    disp('Se encontraron los intervalos: ');
    int = [a b];
    disp(int)

% Hallamos la raiz
    c=(a+b)/2;
    ni=0;
    %iter es la variable que me va a guardar el criterio donde se
    %detiene.
    while (abs(f(c))>=iter)
        c = (a+b)/2;
        fc = f(c);
        ni=ni+1;
        if fc == 0
            break
        end
        if sign(f(a))*sign(f(c))<0
            b=c;
        else
            a=c;
        end
    end
    root=c;
    disp('la raiz es:')
    disp(root)
    disp('El numero de iteraciones es:')
    disp(ni)
  %graficamos la funcion
  fplot(f,[-11,11])
  hold on
  %graficamos el punto de la raiz
  plot(root,f(root),'*')
  grid on;
end
```