ALGORÍTMICA NUMÉRICA

Apellidos: Serrano Arrese
Nombre: Julia

Incluir códigos empleados, resultados , gráficas y respuestas pedidos. No se darán por válidos los resultados que no se deriven de la secuencia de sentencias incluidas en la solución de cada ejercicio.

Ejercicio 2

```
clear;

clc;

valor_real = 1.839286755214161

%Metodo1: metodo de newton

x0 = 1.5;

[s,cif1,error1] = newton(@fun,x0,10)
```

```
 1 ->  2.000000000000000 Error: 8.74e-02
 2 ->  1.857142857142857 Error: 9.71e-03
 3 ->  1.839544513457557 Error: 1.40e-04
 4 ->  1.839286810068019 Error: 2.98e-08
 5 ->  1.839286755214164 Error: 1.45e-15
 6 ->  1.839286755214161 Error: 1.21e-16
 7 ->  1.839286755214161 Error: 1.21e-16
 8 ->  1.839286755214161 Error: 1.21e-16
 9 ->  1.839286755214161 Error: 1.21e-16
10 ->  1.839286755214161 Error: 1.21e-16
```

```
%funcion auxiliar

function [f,fp] = fun(x)

f = x.^3 - x.^2 -x -1;

if nargout == 1, end  %Si no se requiere derivada fin

fp = 3 * x.^2 - 2 * x - 1; %Se calcula la derivada
```

```
end

%funcion metodo newton

function [s,ncif,error]=newton(fun,x0,N)

% fun puntero a la función: debe devolver f(x) y f'(x),

% x0  punto inicial

% N número iteraciones

error = size(1,N);

s = 1.839286755214161;

for k=1:N

% Llamada a f para obtener valores de f(x) y f'(x)

[f fp] = fun(x0);% Llamada a f para obtener valores de f(x) y f'(x)

if f==0, break; end   % Ya he obtenido la raiz y termino

x1 = x0-(f/fp);        % Iteracion de Newton.

error(k) = abs(x1 - s)/s;

ncif = floor(-log10(error(k)));

fprintf('%2d -> %18.15f Error: %.2e \n',k,x1,error(k));  % Volcar datos iteracion.

x0=x1;       % Actualizar x0 con el ultimo valor x1 para volver a iterar.

end

s = x1;     % Al final devuelvo último término de la sucesión.

end


%Metodo 2

iteracion = 1;

error2 = size(1,10)

for i =1:10

   xn = nthroot(1 + x0 + x0.^2,3);

   error2(i) = abs(xn - x0)/valor_real;

   cif2 = floor(-log10(error2));
```

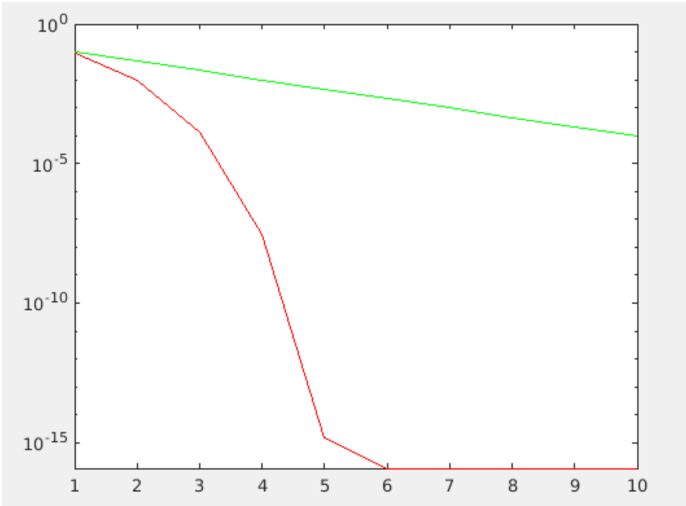
```
   fprintf('Iter: %2d x= %.16f  Error rel: %.2e\n', i,xn,error2(i))

   x0 = xn;

end
```

```
Iter:  1 x= 1.6809877033994816  Error rel: 9.84e-02
Iter:  2 x= 1.7658914314909382  Error rel: 4.62e-02
Iter:  3 x= 1.8053609700852258  Error rel: 2.15e-02
Iter:  4 x= 1.8236277119756392  Error rel: 9.93e-03
Iter:  5 x= 1.8320638965517977  Error rel: 4.59e-03
Iter:  6 x= 1.8359561888024496  Error rel: 2.12e-03
Iter:  7 x= 1.8377512023704059  Error rel: 9.76e-04
Iter:  8 x= 1.8385788376552950  Error rel: 4.50e-04
Iter:  9 x= 1.8389604024141673  Error rel: 2.07e-04
Iter: 10 x= 1.8391363073979421  Error rel: 9.56e-05
```

```
%cifras

cif1_final = cif1(end)          %Metodo 1: 15 cifras correctas

cif2_final = cif2(end)          %Metodo 2: 4 cifras correctas

%graficas error relativo

subplot(1,2,1), semilogy(error1,'*r'),title('Error Relativo 1')

subplot(1,2,2), semilogy(error2,'*r'),title('Error Relativo 2')
```

- **¿Cuántas iteraciones son necesarias para alcanzar la precisión de la máquina con el método de Newton?**

Serán necesarias 7 iteraciones.



2. Orden de convergencia.

%orden de convergencia

k2 = error2(2:end)./error2(1:end-1) %0.46   %orden 1

gan_iter=-log10(k2(end));   %0.3363

it_10=10/gan_iter;   %29.73

k1 = error1(2:end) ./ (error1(1:end-1).^2)   %orden 2

- ¿Qué tipo de convergencia proporciona el método 2?

Convergencia de orden 1

Determinar el valor de K para este método y usarlo para estimar cuántas iteraciones son necesarias para ganar una cifra decimal. ¿En qué iteración se alcanzarían 10 cifras decimales correctas?

k = 0.46

1 iteración necesarias para ganar cifra decimal

En la iteración 30 se conseguirán 10 cifras decimales correctas

- Como se aprecia en los resultados y como sabemos que el método de Newton tiene convergencia 2 cuadrática e n 1 K e n , determinar el valor de la constante K a partir de las estimaciones del error obtenidas.

Tiene convergencia cuadrática ya que en cada iteración se duplican las cifras decimales

k ~ 4.5 10^15