Cálculo numérico
Trabajo Obligatorio

Ingeniería en Informática

Profesor: Pablo Ortiz
Horario: Martes y Jueves de 18 a 20 horas

Victoria Aloy. Cédula: 4.713.341-6
Roberto Parma. Cédula: 4.990.207-9

Ejercicio 2*

Determinar las raíces reales de la función $f(x) = \ln(x^4) - 0.7$

a. Usando el algoritmo de búsqueda incremental determinar los intervalos posibles donde se encuentran raíces.
b. Realice 5 iteraciones del método de bisección en alguno de los intervalos encontrados.
c. Ídem 5 iteraciones del método de regula falsi.
d. Comprobar los resultados usando el comando `fzero()`. A partir del resultado real, ¿qué concluye de los métodos anteriores?.

a.

```
%Algoritmo búsqueda incremental
%ejecutar: func=@(x) log(x.^4) - 0.7;
%ejecutar: P2_E2_A(func, -50, 50, 1000)
% -1.2513  -1.1512
%  1.1512   1.2513
clc
disp('Ejecutando..')
function xb = P2_E2_A(func, xmin, xmax, ns)
  %incsearch(func, xmin, xmax, ns):
  %finds brackets of x that contain sign changes of a function on an interval
  %input:
  % func = name of function
  % xmin, xmax = endpoints of interval
  % ns= (optional) number of subintervals along x used to search for brakets
  %output:
  % xb(k, 1) is the lower bound of the kth sign change
  % xb(k, 2) is the upper bound of the kth sign change
  % If no brackets found, xb = []

  if nargin < 4, ns=50; end %if ns blank set to 50

  %Incremental search
  x=linspace(xmin, xmax, ns);
  f= feval(func, x);
  nb=0;
  xb=[]; %xb is null unless sign change detected
```

```
for k=1:length(x)-1
  if sign(f(k)) ~= sign(f(k+1)) %check for sign change
    nb=nb+1;
    xb(nb, 1)= x(k);
    xb(nb, 2)= x(k+1);
  endif
endfor

if isempty(xb) %display that no brackets were found
  disp('no brackets were found')
  disp('no intervals or increase ns')
else
  disp('number of brackets: ') %display number of brackets
  disp(nb)
end
endfunction
```

Command Window

```
>> func=@(x) log(x.^4) - 0.7;
>> P2_E2_A(func, -50, 50, 1000)
number of brackets:
 2
ans =

  -1.2513  -1.1512
   1.1512   1.2513

>>
```

b.

```
%Algoritmo de Biseccion
%ejecutar: f=@(x) log(x.^4) - 0.7;
%ejecutar: [raiz, fval, iter]=P2_E2_B(f, -1.2513, -1.1512, 10^-8, 5)
clc
disp('Ejecutando..')
function [raiz, fval, ea, iter] = P2_E2_B(f,xl,xu,tol,maxiter)
  %inicializacion
  fprintf('%3s %4s %6s %6s \n', 'iter', 'xl', 'xr', 'xu', 'ea%');
  fprintf('--------------------------------------\n');
  iter=0;
  xr=xl;
  ea=100;
  while(f(xr) !=0 && ea>tol && iter<=maxiter)
    fprintf('%3d %6.4f %6.4f %6.4f %6.4f \n', iter, xl, xr, xu, ea);
    xrold=xr;
    xr=(xl+xu)/2;
    %elegir el siguiente intervalo [xl, xu] usando biseccion
    signo=f(xl)*f(xr);
    if signo<0
      %la raiz esta en la parte izquierda [xl, xr] de [xl,xu]
      xu=xr;
     elseif signo>0
      % raiz en la parte derecha [xr, xu] de [xl,xu]
      xl=xr;
    else
      ea=0;
    end
    ea=abs((xr-xrold)/xr)*100; %error relativo porcentual aprox.
    iter++;
  endwhile

  if iter>maxiter
    fprintf('No converge para el maximo de iteraciones %d \n', iter);
  endif
  raiz=xr;
  fval=f(xr);
endfunction
```

```
Command Window
>>  f=@(x) log(x.^4) - 0.7;
>> [raiz, fval, iter]=P2_E2_B(f, -1.2513, -1.1512, 10^-8, 4)
iter   xl     xr     xu
ea% ---------------------------------------
  0 -1.2513 -1.2513 -1.1512 100.0000
  1 -1.2012 -1.2012 -1.1512 4.1665
  2 -1.2012 -1.1762 -1.1762 2.1276
  3 -1.2012 -1.1887 -1.1887 1.0526
  4 -1.1950 -1.1950 -1.1887 0.5235
No converge para el maximo de iteraciones 5
raiz = -1.1919
fval =  0.0020793
iter =  0.26246
>>
```

c.

```
%raiz 1.1912
clc
clear
fp = @(x) log(x.^4) - 0.7;

xl = -1.2513;
xu = -1.1512;
tol= 10^-8;
maxiter=5;
iter=0;

yl = fp(xl);
yu = fp(xu);

if yl==0
fprintf('%g is a root\n', xl)
endif

if yu==0
fprintf('%g is a root\n',xu)
endif

if yl*yu>0
fprintf ('There are no roots in the interval\n')
endif

```

```
if yl*yu<0
  xr=xu-((xu-xl)*yu/(yu-yl));
  yr=fp(xr);
  error=tol+1;
%error>tol  &&
  while iter<maxiter
    if yr*yu<0
      xl=xr;
      yl=fp(xl);
    else
      xu=xr;
      yu=fp(xu);
    endif

    xr=xu-((xu-xl)*yu/(yu-yl));
    yr=fp(xr);
    error=abs(xu-xr);
    error

    iter=iter+1;
  endwhile
end

```

```
if yr==0
  fprintf('%g is a root of the function\n', xr)
else
  if error<tol
    fprintf('%.10f is an aproxrmation to the root of the function \n with a tolerance of %10e\n', xr,tol);
  endif
  if iter==maxiter
    fprintf('%.10f is an aproxrmation to the root of the function \n with a iter of %10d\n', xr,iter);
  endif
endif

```

Command Window

```
error =   0.040063
error =   0.040047
error =   0.040046
error =   0.040046
error =   0.040046
-1.1912462166 is an aproxrmation to the root of the function
 with a iter of          5
>>
```

d.

```matlab

x0=0;
fun = @(x) log(x.^4) - 0.7;
x = fzero(fun, x0);
x

%x = -1.1912

%P2_E2_B
%iter   xl     xr     xu
%ea% ---------------------------------------
%  0 -1.2513 -1.2513 -1.1512 100.0000
%  1 -1.2012 -1.2012 -1.1512 4.1665
%  2 -1.2012 -1.1762 -1.1762 2.1276
%  3 -1.2012 -1.1887 -1.1887 1.0526
%  4 -1.1950 -1.1950 -1.1887 0.5235
%  5 -1.1919 -1.1919 -1.1887 0.2625
%No converge para el maximo de iteraciones 6
%raiz = -1.1903
%fval = -0.0031732
%iter =  0.13140
```

```matlab

%P2_E2_C
%error =  0.040063
%error =  0.040047
%error =  0.040046
%error =  0.040046
%error =  0.040046
%-1.1912462166 is an aproxrmation to the root of the function
% with a iter of          5

%El método de regula falsi converge más rápido que el de bisección ya que se acerca más al resultado en base a
%los mismos párametros.


```

Command Window

```
error =  0.040063
error =  0.040047
error =  0.040046
error =  0.040046
error =  0.040046
-1.1912462166 is an aproxrmation to the root of the function
 with a iter of          5
>> P2_E2_D

x = -1.1912
>>
```

Ejercicio 5*

Dada la función cúbica $f(x) = x^3 - 3x - 20 = 0$ la misma puede ser escrita como:

1. $x = \frac{(x^3-20)}{3}$
2. $x = \frac{20}{(x^2-3)}$
3. $x = (3x + 20)^{1/3}$

a. Seleccione la función que satisface el teorema de punto fijo en el intervalo [1,4].
b. Luego partiendo del punto $x_0 = 1.5$ itere hasta lograr un error relativo porcentual de $10^{-2}\%$.

a.

Práctica 2

Ej 5

$f(x) = x^3 - 3x - 20 = 0$

1) $x = \dfrac{(x^3 - 20)}{3}$

2) $x = \dfrac{20}{(x^2 - 3)}$

3) $x = (3x + 20)^{1/3}$

intervalo $[1, 4]$

1) $x = \dfrac{x^3 - 20}{3}$

1. $g(x)$ continua en el intervalo
2. $g(x)\ [1,4] \to [1,4]$ X

$g(1) = \dfrac{1^3 - 20}{3} = -6{,}3$

$g(2) = \dfrac{2^3 - 20}{3} = -4$

2) $x = 20/(x^2 - 3)$

1. $g(x)$ no es continua en $+\sqrt{3}$ X

3) $x = (3x + 20)^{1/3}$

1. $g(x)$ continua ✓
2. $g(x)\ [1,4] \to [1,4]$ ✓
3. $|g'(x)| \leq k\ ;\ k \in (0,1)$ ✓ $\to g'(x) = 1/3 \cdot (3x+20)^{-2/3} \cdot 3$

$g(1) = (3+20)^{1/3} = 23^{1/3} = 2{,}84$

$g(2) = (6+20)^{1/3} = 26^{1/3} = 2{,}96$

$g(3) = (9+20)^{1/3} = 29^{1/3} = 3{,}07$

$g(4) = (12+20)^{1/3} = 32^{1/3} = 3{,}17$

$g'(1) = 0{,}123$ $\quad g'(3) = 0{,}105$

$g'(2) = 0{,}113$ $\quad g'(4) = 0{,}009$

La función que satisface el teorema de punto fijo en el intervalo es la número 3.

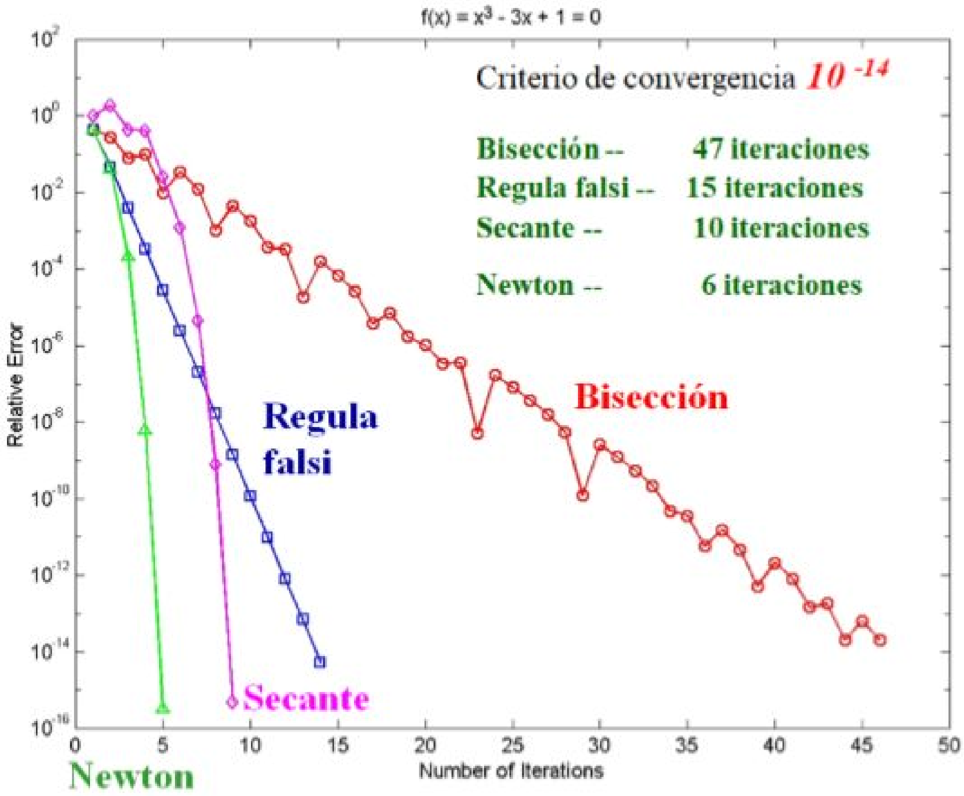
b.

```
%Punto fijo
disp('Ejecutando...')
g= @(x)  (3*x+20)^(1/3)
es=10^-2;
ea=100;

xr=1;
xr_old=0;
sigo=1;
iter=0;

fprintf('\nAprox raiz    Iteraciones    Error relativo porcentual aprox ')
do
  xr_old=xr;
  xr=g(xr_old);
  if xr>0 || xr<0
    ea=abs((xr-xr_old)/xr)*100;
    sigo=1;
  else
    sigo=0;
  endif
  iter=iter+1;
  fprintf('\n %10.3f           %d                   %d', xr, iter, ea)
until !(ea>es && sigo==1)

disp('Valores')
xr
xr_old

disp('Función evaluada')
f=@(x)x^3-3*x-20
evaluada=f(xr)
```

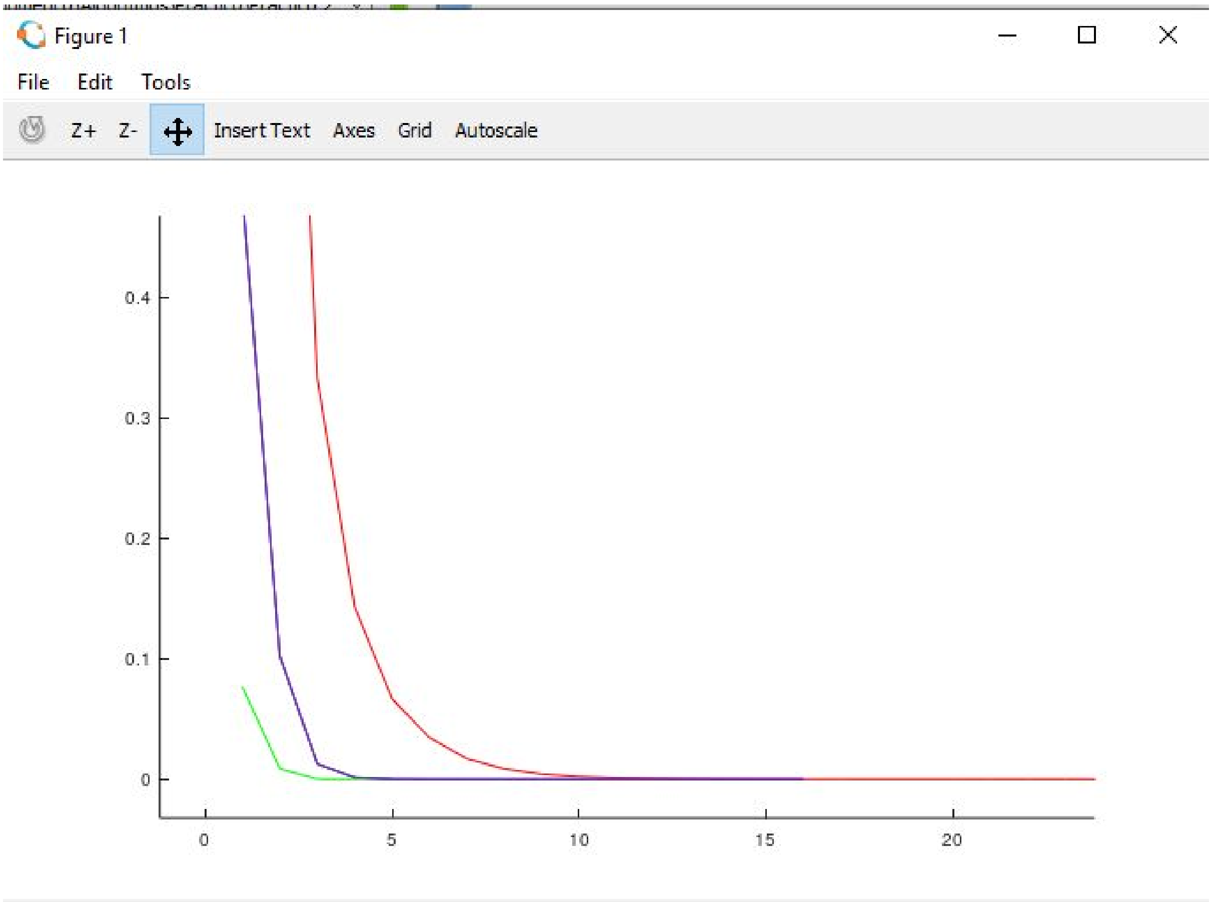
```
Ejecutando...
g =

@(x) (3 * x + 20) ^ (1 / 3)


Aprox raiz    Iteraciones    Error relativo porcentual aprox
      2.844           1                    64.8366
      3.056           2                    6.93196
      3.078           3                    0.731565
      3.081           4                    0.0770883
      3.081           5                    0.00812181Valores
xr =  3.0808
xr_old =  3.0806
Función evaluada
f =

@(x) x ^ 3 - 3 * x - 20

evaluada = -0.00075066
>>
```

Ejercicio 6*

Dada la función $f(x) = x^3 - 3x + 1$ se quiere graficar la cantidad de iteraciones vs. el error relativo para un error relativo máximo de 10^{-14} , asimismo imprimir la cantidad de iteraciones por cada método tal como se indica en el gráfico a continuación.



```
f=@(x) x.^3 - 3.*x + 1;
%raiz = -1.8794

xl=-2.0408;
xu=2.0408;
tol=10^(-14);
d=@(x) 3.*x.^2 -3;

[vectPasosBis, vectEaBis, raiz, fval, ea, iter] = P2_E6_biseccion(f,xl,xu,tol);
[vectPasosNew, vectEaNew] = P2_E6_newtonraphson(f,d,xl,tol);
[vectPasosRegla, vectEaRegla] = P2_E6_reglafalsa(f,xl,xu,tol);
[vectPasosSec, vectEaSec] = P2_E6_secante(f,xl,xu,tol);
hold on;
plot(vectPasosBis, vectEaBis, 'color','red');
plot(vectPasosNew, vectEaNew, 'color','green');
plot(vectPasosRegla, vectEaRegla, 'color','blue');
plot(vectPasosSec, vectEaSec);


fprintf('Evaluada: ')
f(-1.8794)
```



Ejercicio 9*

El método de aceleración de la convergencia de Aitken Δ^2 (delta cuadrado) se usa para acelerar otros métodos, por ej. la búsqueda de raíces, y se define como:

$$x_{n+3}{}^{*} = x_{n+2} - \frac{(x_{n+2} - x_{n+1})^2}{(x_{n+2} - 2x_{n+1} + x_n)} = x_{n+2} - \frac{(\Delta x_{n+1})^2}{\Delta^2 x_n} \quad n \geq 1$$

Se usan 3 valores previos los cuales se calculan por cualquier otro método (por ej. punto fijo) y el siguiente valor se calcula aplicando la fórmula anterior. De esta forma un algoritmo de convergencia lineal, como el de punto fijo, puede ser acelerado para converger de forma cuadrática.

Considere la función $f(x) = x - \frac{1}{2}e^{-x}$ para la cual se quieren determinar sus raíces partiendo de $x_1 = 1$. Se requiere completar el siguiente cuadro hasta lograr que $\varepsilon_a\,\% = \mathbf{10^{-6}}\%$

Iter	Punto fijo	Aitken	Ea PF	Ea Aitken
1	0.18393972	0.18393972	443.65636569	443.65636569
2	0.41599298	0.41599298	55.78297453	55.78297453
3	0.32984245	0.32984245	26.11868933	26.11868933
4	0.35951850	0.35316685	8.25438795	6.60435768
5	0.34900617	0.35122999	3.01207744	0.55145243
6	0.35269439	0.35191093	1.04572745	0.19349918
7	0.35139597	0.35173381	0.36950319	0.05035833
8	0.35185253	0.35173368	0.12975767	0.00003609
9	0.35169192	0.35173372	0.04566599	0.00001269
10	0.35174841	0.35173371	0.01605907	0.00000330
11	0.35172854	0.35173371	0.00564891	0.00000000
12	0.35173553	0.00000000	0.00198686	0.00000000
13	0.35173307	0.00000000	0.00069885	0.00000000
14	0.35173394	0.00000000	0.00024581	0.00000000
15	0.35173363	0.00000000	0.00008646	0.00000000
16	0.35173374	0.00000000	0.00003041	0.00000000
17	0.35173370	0.00000000	0.00001070	0.00000000
18	0.35173371	0.00000000	0.00000376	0.00000000
19	0.35173371	0.00000000	0.00000132	0.00000000

>> |

Ejercicio 10*

Contrariamente a los que ocurre con otros temas, por ej. sistemas de ecuaciones lineales, la solución de los sistemas no lineales es una cuestión delicada. Generalmente es difícil garantizar la convergencia a la solución exacta y además la complejidad computacional crece muy rápidamente con el tamaño del sistema.

Por ejemplo, considere el siguiente sistema:

$$\begin{cases} y - x^3 - \frac{1}{2} = 0 \\ 4y^2 - x + \boldsymbol{c} = 0 \end{cases} \quad ; \; x, y \in [-2,1]$$

a. Aplicando el método de Newton-Raphson resuélvalo para $c = \{0,\ 0.7,\ -1,\ -1.5\}$

b. Realice los 4 cuadros simultáneamente que muestren el comportamiento de las ecuaciones para los distintos valores de ***c***.

```
aa=0;
bb=0.7;
cc=-1;
dd=-1.5;

F1=@(x) [x(2)-x(1)^3-1/2;4*x(2)^2-x(1)+aa];
F2=@(x) [x(2)-x(1)^3-1/2;4*x(2)^2-x(1)+bb];
F3=@(x) [x(2)-x(1)^3-1/2;4*x(2)^2-x(1)+cc];
F4=@(x) [x(2)-x(1)^3-1/2;4*x(2)^2-x(1)+dd];

J=@(x) [-3*x(1)^2,1;-1,8*x(2)];

x=[-2;1];

P2_E10_newtonm(x,F1,J, aa);
fprintf('\n-------------------------------------------------------------')
P2_E10_newtonm(x,F2,J, bb);
fprintf('\n-------------------------------------------------------------')
P2_E10_newtonm(x,F3,J, cc);
fprintf('\n-------------------------------------------------------------')
P2_E10_newtonm(x,F4,J, dd);
fprintf('\n-------------------------------------------------------------')
```

```
Valor de c = 0.0000000000
No converges after 100 iterations

------------------------------------------------------------------


Valor de c = 0.7000000000
Solutions diverges
iterations=79

------------------------------------------------------------------


Valor de c = -1.0000000000
 x =

  -0.54669
   0.33664



------------------------------------------------------------------


Valor de c = -1.5000000000
 x =

  -0.95439
  -0.36933
```