MA-0501 Análisis Numérico I
SOLUCIONARIO - TAREA 1

El documento suministrado se publicó como `.tex` con el archivo de estilo `matlab2latex.tex` mediante el comando
Contents


```
publish('T1_Soto_sol','format','latex','stylesheet','matlab2latex.tex')
```

Respuesta Corta
1) Particularmente, no se puede utilizar a priori los métodos de bisección ni de Newton. A priori, no se podría utilizar el método de interación. Se considera entonces considerar la función $h(x) = |x^2 - 1|e^{-x^2}$. Esta función, comparte las mismas raices, además, se puede verificar que es una contracción sobre R. Ver que es una contracción es sencillo al calcular la derivada; no se presenta la prueba. Se puede utilizar el método de iteraciones sucesivas a la función h.
2) Al utilizar el Método de Newton para resolver la ecuación x^{100} =0 , con un valor inicial distinto de cero, esperaría que el método no converga. Lo anterior, por cuanto alrededor de cero, la derivada de la función f(x) = x^100 toma valores cercanos a 0, conforme x se toma más pequeño. En el curso se ha abordado que una de las limitaciones del Método de Newton es justamente esta. Geométricamente, la recta tangente a la curva en las valores cercanos a cero se vuelve cada vez más horizontal.
3) Una primera opción es calcular numéricamente las derivadas de la función en la raíz, hasta encontrar una derivada que no se anule, lo que resultaría en el cálculo de la multiplicidad. Esto puede ser computacionalmente costoso, pero es una posibilidad. Otra posiblidad sería pensar medir la rápidez convergencia de métodos como el Newton modificado.

4) Para precisión doble, el epsilon de la maquina viene dado por 2ˆ{-52}. Por lo tanto, a la iteración 52, la longitud del intervalo será de 2ˆ{-52}. De esta manera, a partir de realizar la iteración 52, se sobrepasaría el epsilon de la maquina. Por esta razón, no tiene sentido realizar más de 52 iteraciones.
5) Inciso a) Dado x_0 real, se necesitan $j+1$ multiplicaciones para calcular $a_j x^j$, con $0 \leq j \leq n$. Por lo tanto, la cantidad de multiplicaciones para evaluar $p(x_0)$ de forma directa corresponde a:

$$\sum_{j=0}^{n}(j+1) = \frac{1}{2}(n+1)(n+2)$$

Por su parte, para evaluar $p(x_0)$ directamente, se necesitan n sumas. En total, se necesitan $\frac{1}{2}(n+1)(n+2)+n$ operaciones.
Inciso b) Considere la sucesión dada en el enunciado. En este caso, para el primer paso, no se ocupa ninguna operación. Posteriormente, para el paso k, con $1 < k \leq n+1$, se ocupan 2 operaciones, una suma y una multiplicación. Con lo que al final, para calcular b_0, se necesitan un total de $2n$ operaciones.
Por lo tanto, para evaluar directamente el polinomio en x_0, se ocupan más operaciones que el método del inciso b, a partir de polinomios de grado 2.
Si r_n denota el cociente entre la cantidad de operaciones de evaluar directamente y el método del inciso b), entonces $r_n = \frac{5}{4} + \frac{1}{4}n + \frac{1}{2}n = O(n)$. Es decir, la razón se comporta casi linealmente en n.
6) ¿Cuál es el polinomio de Lagrange de la función f(x) = xˆ2 + 3x + 1 que interpola a f en los nodos x_{0} = 0, x_1 = 1, x_2 = 2.

```
% Por cuanto se utilizan 3 nodos para interpolar a una funci n polinomial de
% grado 2, el polinomio de Lagrange tiene grado 2. Adem s , en este caso, la
% funci n f coincide con el polinomio de lagrange en los 3 nodos. Por unicidad,
% tiene que el polinomio de Lagrange es justamente la funci n  f(x) = x^2 + 3x +1
```

Desarrollo

Ejercicio 1)

Inciso a) Al ejecutar el código suministrado, se obtiene para $n = 10$ (note que en este caso se ejecuta el código)

```
n = 10;
```

```
f = []; g = [];
f(1)=0;
f(2)=1;
for i=2:(n-1)
    f(i+1)=f(i)+f(i-1);
end

g(n)=f(n);
g(n-1)=f(n-1);
for i=(n-1):-1:2
    g(i-1)=g(i+1)-g(i);
end

% Inciso a)
% El valor m ximo que se puede guardar en precisi n doble en este caso,
% corresponde a
m_max = 10^16;

% Para estimar a mano el menor valor para el que se sobrepasa dicho valor
% m_max. Se utiliza la aproximaci n
```

$F_{n+N} = \phi^n F(N)$, para N lo suficientemente grande.
El código del inciso, solo me corre hasta $n = 46$, por lo que este será el valor de N utilizado. La aproximación es coherente, al considerar el ratio F(46)/F(45).
De esta manera, una aproximación para el valor de n máximo viene dada por:

```
%phi = (1 + sqrt(5))/2;

%n_max= log(m_max/f(46))/log(phi)+46 + 1;

% Inciso c) Particularmente no pude evidenciar el fen meno porque el c digo
% solamente me corre hasta  n = 46.
```

Ejercicio 2)

Inciso a) Sea dado $b \in R$, distinto de cero. Considere la función $f(x) = 1/x - b$. Observe que dicha función es de clase C^2 sobre $R \setminus \{0\}$. Particularmente, $f'(x) = -\frac{1}{x^2}$, y $f''(x) = -\frac{2}{x^3}$, para x distinto de cero.
De esta manera, dado b, se puede elegir un vecindario cerrado, sobre el cual se satisfacen las hipótesis del Teorema de Newton que garantizan la convergencia; particularmente, que la derivada de la función no se anule, y que la segunda derivada sea continua.

```
% Note que dado un valor inicial  x_{0} , la f rmula de iteraci n de Newton
% viene dada por:
```

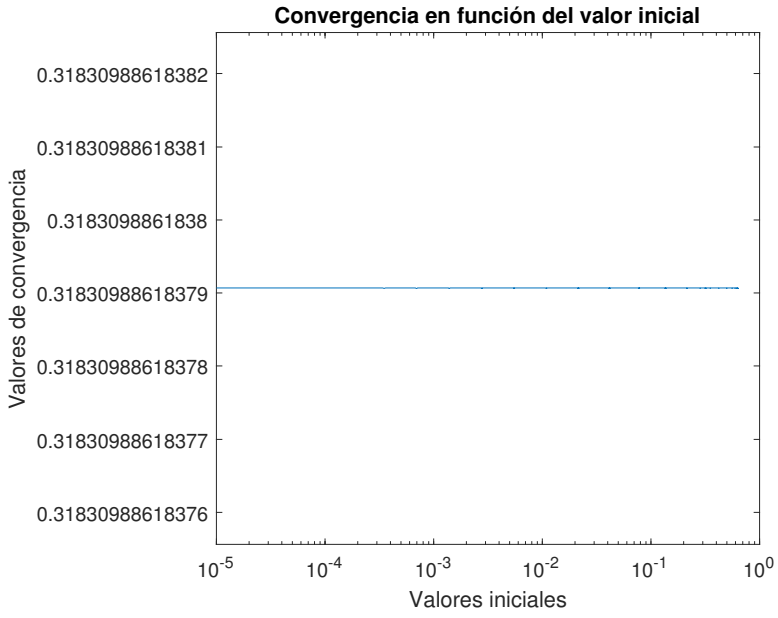
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = 2x_n - bx_n^2$$

para $n \geq 1$.
Inciso b) Implementación de la función myDivision. La función implementada se encuentra al final del presente trabajo. Esta, devuelve como resultado la aproximación de la raíz, como también la sucesión de las iteraciones realizadas por el algoritmo.
Inciso c) Ahora, se fija el valor $b = \pi$. Se comprueba el comportamiento del código para valores iniciales $x_0 \in$ linspace(0,1,1e5).
Se fija el valor deseado.

```
b = pi;

% Se crean los vectores para comprobar el comportamiento del c digo:
valores_iniciales = linspace(0,1,1e5);
valores_convergencia = nan(size(valores_iniciales));

% Se llenan los valores de convergencia con un for
for j = 1:length(valores_iniciales)
    valores_convergencia(j) = myDivision(b,valores_iniciales(j));
end

% Se procede a graficar el valor de convergencia, para los valores iniciales
% seleccionados. Se utiliza escala logar tmica en el eje x.

semilogx(valores_iniciales,valores_convergencia, 'MarkerSize', 20)
title('Convergencia en funci n del valor inicial')
xlabel('Valores iniciales')
ylabel('Valores de convergencia')
```

En este caso, teóricamente, el intervalo de convergencia que garantizado por la prueba del Teorema de Newton, es un subconjunto propio del intervalo $(0, \frac{2/pi}{})$, como se observa en el gráfico anterior.

```
% Inciso d)

% Particularmente, la prueba del Teorema considera la constante A,
% definida de la siguiente manera:
```



$$A := \max_{x,y \in I_\delta} \left| \frac{f''(x)}{f'(y)} \right| = 2 \max_{x,y \in I_\delta} \left| \frac{y^2}{x^3} \right|$$

Sobre un intervalo I_δ centrado en $1/b$, en el cual la derivada no se anule. En este caso, se puede tomar $\delta = 1/b - \epsilon$, $\delta = 1/b + \epsilon$, para los casos respectivos $b > 0$, $b < 0$, $\epsilon > 0$. Suponga sin perder generalidad que $b > 0$.

`latex`

A partir de lo cual, el intervalo de convergencia garantizado por la prueba del teorema viene dado por ($\frac{1}{b}-$\Tilde{\delta}$,\frac{1}{b}+$\Tilde{\delta}$$, con $\tilde{\delta} = \text{mín}\{\frac{1}{b}, 1/A\}$

`latex`

De esta manera, A viene dado por

$$A := 2\frac{(\frac{2}{b} - \varepsilon)^2}{\varepsilon^3}$$

Numéricamente, se puede obtener el valor de $\tilde{\delta}$ que garantiza el teorema de Newton, para distintos valores de ϵ.

Por su parte, se puede observar que $A > 2b$, con lo que para ser un potencial valor inicial x_0, en el intervalo garantizado por el teorema, se tendría que $\frac{1}{2b} < x_0 < \frac{3}{2b}$; equivalentemente, $1\frac{}{2<x_0 b<\frac{3}{2}}$.
Las respectivas modificaciones se pueden realizar para el caso de b<0.

Ejercicio 3)

Considere $N_c = 16$ colonias de insectos. Sea Y_i una variable aleatoria que denota el número de insectos en cada grupo. Considere que las variables Y_i son independientes con distribución binomial y parámetro θ.
Inciso a) Sea $f(y_i;\theta)$ la función de masa de probabilidad de la variable Y_i. La función de máxima verosimilitud, denotada como $L(y_1, ..., y_{N_c};\theta)$, viene dada por:

$$L(y_1, ..., y_{N_c};\theta) = \prod_{i=1}^{N_c} f(y_i;\theta) = \alpha\theta^y(1-\theta)^m \tag{1}$$

```
% <latex>
% donde $ \displaystyle \alpha = \prod \binom{n_{i}}{y_{i}} $,
% $ h = \sum_{i=1}^{N_{c}}y_{i}$, $ m = \sum_{i=1}^{N_{c}}n_{i} -
% \sum_{i=1}^{N_{c}}y_{i}$. Observe que en este caso, $ h $, $ m $
% representan la cantidad total de hembras y macho en las 16
    colonias,
% respectivamente.
% </latex>
```

Se procede a encontrar el valor que maximiza la función $l(\theta) = \log f(y_1, ..., y_{N_c};\theta)$, para $\theta \in (0, 1)$, cond $y_1, ..., y_{N_c}$ dados. Observe que en este caso,

$$l(\theta) = \log(\alpha) + h\log(\theta) + m\log(1 - theta) \tag{2}$$

Se puede observar que dicha función es derivable, y en particular, $l'(\theta) = \frac{h}{\theta} - \frac{m}{1-\theta}$, para cada $\theta \in (0, 1)$. Con el fin de maximizar dicha función, se puede observar que el único punto crítico de la función ocurre en $\theta = \frac{h}{h+m}$. Particularmente, este valor maximiza la función $l(\theta)$, por cuanto la segunda derivada de la función es negativa en su dominio.

```
% Inciso b) Se cargan los datos de la distribuci n de insectos
    seg n
% hembras y machos, de las colonias.

T = readtable('data.txt');

% Se visualiza la tabla cargada:
disp(T);
```

```
8
9  % Inciso c)
10
11 % Se convierte el contenido de cada columna a un vector numérico a
       un vector
12 % numérico, con los siguientes comandos:
13
14 M = T.Machos ;
15 H = T. Hembras ;
16
17 % Se obtienen los valores de interés para el c lculo de la
       estimaci n de
18 % M xima Verosimilitud.
19 m = sum(M);
20 h = sum(H);
21
22 % La estimaci n para el par metro $ \theta $ es:
23 MLE = (h)./(m+h);
24
25 % Inciso d) Implementaci n del método de la secante para aproximar
       la
26 % soluci n numérica de la ecuaci n $\dfrac{dl}{d\theta} = 0$.
27
28 % El c digo del método de la secante se puede encontrar al final
       del
29 % presente documento.
30
31 % Se utiliza un function handle de matlab, para especificar la
       funci n
32 % derivada de $l(\theta)$, que es la funci n a la cual se quiere
       aproximar
33 % la ra z.
34 l_derivada = @(x) h/x - m/(1-x) ;
35
36 % Como condiciones iniciales para el método de la secante, se usan
       las
37 % siguientes:
38 x0 = 0.25;
39 x1 = 0.75;
40
41 % El valor de la ra z , con 15 decimales de exactitud viene dado
       por:
42 c = MLE;
43
44 % Se utiliza el método de la secante para obtener la ra z
       aproximada,
45 % como la sucesi n de las iteraciones.
```

```
46 [rS,secS] = Secante(l_derivada,x0,x1) ;
47
48 % error absoluto
49 D_3d_iterSec_errA = abs(c-rS);
50 D_3d_iterSec_err = abs(c-rS)/c;
51
52 % Se observa que la aproximaci n obtenida con el método de la
     secante,
53 % concuerda en 16 decimales con el valor obtenido en el inciso
     anterior.
54
55 % Inciso e) Ahora, se utiliza también la funci n fsolve de MATLAB
     para
56 % aproximar la ra z deseada.
57
58 % Se utiliza la funci n fsolve, introduciendo dos par metros:
59 % La funci n a la que se desea encontrar la ra z.
60 % El segundo par metro corresponde al valor inicial.
61
62 [x,fval,exitflag,output,jacob] = fsolve(l_derivada,0.1);
63
64 % Al utilizar el comando anterior, entre los retornos de la
     funci n se
65 % encuentran:
66 % x: La soluci n dada por el algoritmo
67 % fval: el valor de la funci n en la soluci n encontrada
68
69 % exitflag: indica la raz n por la que se detuvo el algoritmo.
     Toma las
70 % siguientes representaciones seg n la documentaci n:
71 % Si la ecuaci n fue resulta:
72 % 1: La optimalidad de primer orden es peque a.
73 % 2: El cambio en las iteraciones es peque a con respecto
74 % a la tolerancia indicada (o la default si no se indic ).
75 % 3: El cambio en el valor residual es menor que la
76 % tolerancia especificada.
77 % 4: La magnitud de la direcci n de b squeda es menor que la
     tolerancia especificada.
78
79 % Adicionalmente:
80 %  0: Se ha detenido por la cantidad m xima de iteraciones o
     evaluaciones
81 %  de la funci n.
82 % -1: La funci n de salida o la funci n de gr fica ha detenido
     el algoritmo.
83 % -2: Ecuaci n no resulta. El mensaje de salida puede tener m s
     informaci n
```

```
% -3: Ecuaci n no resulta. El radio de la regi n de confianza del
    algoritmo
% se ha vuelto demasiado peque o

% output: especifica informaci n sobre el proceso de optimizaci n
    ,
% devuelta como estructura con campos:

% Con el fin de indicar el algoritmo que se utiliz , se revisa:

disp(output);

% Se aprecia que el algoritmo utilizado fue 'trust-region-dogleg',
    que
% seg n la documentaci n es una modificaci n del método dogleg
    propuesto por
% Michael J. D. Powell.

D_3e_fsolve_errA = abs(x-rS);
D_3e_fsolve_err = abs(x-rS)/c;
```

```
    Poblacion    Machos    Hembras
    ---------    ------    -------

        1          11        18
        2          22        31
        3          27        34
        4          29        33
        5          24        27
        6          29        33
        7          25        28
        8          26        23
        9          38        33
       10          14        12
       11          23        19
       12          31        25
       13          20        14
       14           6         4
       15          34        22
       16          12         7

    0.4946


Equation solved.

```

```
fsolve completed because the vector of function values is near zero
as measured by the value of the function tolerance, and
the problem appears regular as measured by the gradient.

       iterations: 6
        funcCount: 14
        algorithm: 'trust-region-dogleg'
    firstorderopt: 3.3382e-10
          message: 'Equation solved.  fsolve completed because
              the vector of function values is near zero as
              measured by the value of the function tolerance,
              and the problem appears regular as measured by the
              gradient.      <stopping criteria details>
                  Equation solved. The sum of squared function
              values, r = 1.292470e-26, is less than sqrt(options.
              FunctionTolerance) = 1.000000e-03. The relative norm
              of the gradient of r, 3.338242e-10, is less than
              options.OptimalityTolerance = 1.000000e-06.'
```

Ejercicio 4)

Inciso a)

`latex`

Considere la función g, con criterio $g(x) = x - \frac{x^2-3}{2}$, para $x \in [1, 2]$

`/latex`

```
% <latex>
% Dicha funci n $g$, es derivable, y su derivada viene dada por
% g'(x) = 1 - x %. De esta manera, para cada $ x \in (1,2) $, se
    satisface que
% $|g'(x)| < 1$. Lo que muestra que $ g $ es una contracci n débil
    .
% </latex>
```

Por su parte, por el cálculo de la función derivada, se observa que la función g es decreciente sobre el intervalo $[1, 2]$. Por lo tanto, se satisface que $g([1, 2]) = [g(2), g(1)] = [\frac{3}{2}, 2]$.
De esta manera, por cuanto el intervalo $[1, 2]$ es compacto, se tiene que la contracción débil, g, tiene un único punto fijo en [1,2], y además, la iteración simple converge a dicho punto.

```
% Una prueba del resultado anterior puede ser encontrada en TKTKT.
```

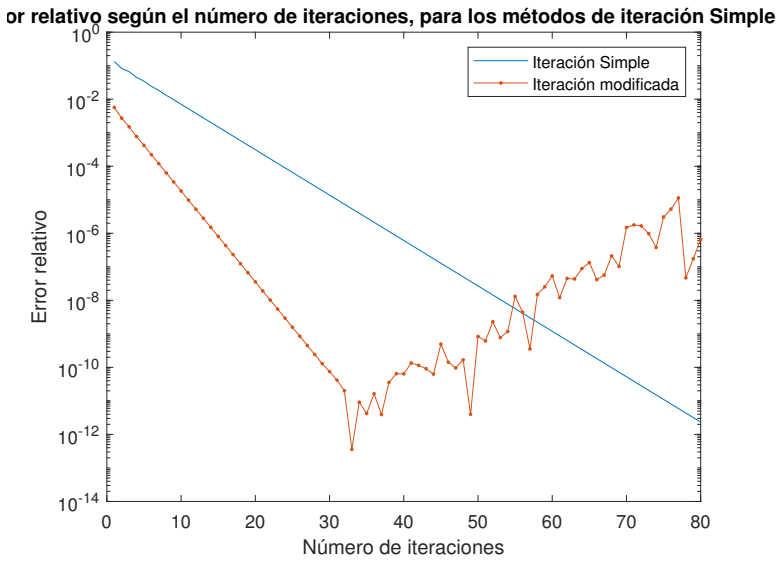
En este caso, se sencillo observar que el punto fijo en cuestión corresponde a $c = \sqrt{3}$.

```
% Inciso b)

g = @(x) x - (x^2 -3)/2 ;

% Para la iteraci n , se utiliza el siguiente valor inicial:
c0 = 1.5;

% Punto fijo de la funci n $g$, con 16 cifras de exactitud.
c = sqrt(3); % Ra z exacta

% Se utiliza la iteraci n simple:
[rSimple,sSimple] = iterSimple(g,c0,80);

% Ahora, la secuencia que contiene el error relativo de cada
    iteraci n , para
% el punto inicial seleccionado, viene dada por:

D_4b_iterS_err = abs(c-sSimple)/c;

% Graficaci n del error relativo en funci n del n mero de
    iteraciones:
% En el eje y se utiliza una escala logar tmica para apreciar de
    mejor
% manera el comportamiento del error:

semilogy(1:80,D_4b_iterS_err,"- .")
xlabel ('N mero de iteraciones')
ylabel('Error relativo')
title(['Error relativo seg n el n mero de iteraciones, con el mé
    todo de ' ...
     'Iteraci n Simple para la funci n g'])

% Inciso c)

% El c digo que calcula la sucesi n $a_{k}$ se presenta al final
    del
% presente documento.

[rMod,sMod] = iterModified(g,c0,80);

% An logamente , se considera el error relativo en funci n del
    n mero de
% iteraciones para el nuevo método.
```

```
D_4c_iterM_err = abs(c-sMod)/c;

% Se procede a graficar los errores relativos de ambos métodos empleados
% para observar el comportamiento.

semilogy(1:80,D_4b_iterS_err,1:80,D_4c_iterM_err, ...
    "- .");
xlabel ('N mero de iteraciones')
ylabel('Error relativo')
title(['Comparaci n de error relativo seg n el n mero de iteraciones, para ' ...
    'los métodos de iteraci n Simple e iteraci n Modificada'])
legend('Iteraci n Simple','Iteraci n modificada','Location','northeast')

% Para este caso, se observa que el método de iteraci n modificado genera
% una convergencia m s r pida, en el sentido de una menor cantidad de
% iteraciones para obtener errores similares a los obtenidos con la
% iteraci n simple.

% También as , en el gr fico se observa un comportamiento decreciente
% del error hasta la iteraci n 32 del método de iteraci n modificada.
% A partir de dicha iteraci n, el error var a, manteniendo en general,
% una tendencia a la alza.

% Este efecto puede deberse a que cuando las iteraciones llegan a un error
% relativo bajo, el denominador del término general $a_{k}-c$ disminuye en
% menor cuant a con respecto al denominador, lo que provoca que el error
% relativo aumente conforme al n mero de iteraciones.
```

Ejercicio 5)

```
% Inciso a) Se cargan los datos del archivo .mat ejecutando el siguiente
% comando:
load('dataPolin.mat')
```

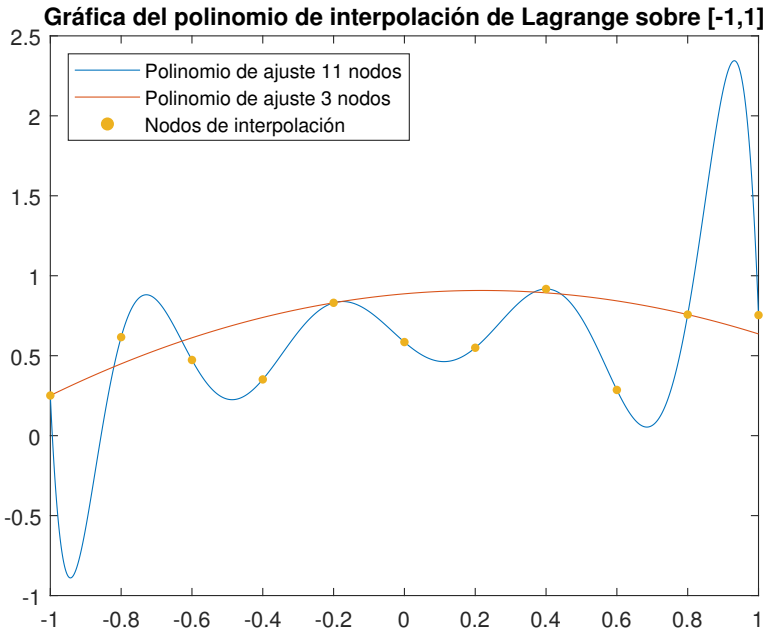



```
4
5  % Para verificar el tama o de los vectores que fueron cargados se
       utilzia
6  % el siguiente comando:
7  whos("dataX","dataY")
8
9  % Se aprecia que ambos vectores tienen el mismo tama o. Dicho
       tama o se
10 % guarda en una variable:
11 n = length(dataX);
12
13 % Inciso b) Construcci n del polinomio de interpolaci n de
       Lagrange que
14 % pasa por los puntos (x_{i},y_{i}) suministrados.
15
16 % Dada una cantidad de 80 nodos, {x_{i}}, el polinomio de Lagrange
17 % corresponde a la  nica  funci n polinomial de grado menor o
       igual que 80,
18 % que en el nodo x_{i}, tiene el valor de y_{i}.  El grado del
19 % polinomio de Lagrange en este caso corresponde a 79.
20
21 % En este sentido, la funci n polyfit de matlab, con el comando
22 % polyfit(x,y,m) retorna los coeficientes de un polinomio de grado
       m, que
23 % mejor ajusta los datos de x con los de y, siendo el ajuste con el
        método
24 % de m nimos cuadrados.
25
```

```
% De esta manera, por la unicidad del polinomio de Lagrange, si se utiliza
% m = n-1 = 79, los coeficientes que retorna la funci n polyfit corresponden
% a los respectivos coeficientes del polinomio de Lagrange.

% Por lo tanto, se utiliza el siguiente comando para encontrar el polinomio
% de Lagrange, con los respectivos puntos suministrados:

pn = polyfit(dataX,dataY,n-1);
% Coeficientes polinomiales con ajuste de m nimos cuadrados, devueltos como vector.

% Inciso c) Ahora, se utiliza la funci n polyval para evaluar el polinomio
% que se obtuvo en el inciso anterior.

% La funci n polyval recibe como par metros un vector de coeficientes, que
% corresponden a los coeficientes del polinomio que se desea evaluar.
% Adem s, un vector de puntos donde el polinomio es evaluado.
xx = linspace(-1,1,1e5);
yy = polyval(pn,xx);

% Se grafica el polinomio sobre el intervalo $[-1,1]$, junto con los nodos
% de interpolaci n.
plot(xx,yy)
hold on
scatter(dataX,dataY,15,'filled')
title('Gr fica del polinomio de interpolaci n de Lagrange sobre [-1,1]')
legend('Polinomio de ajuste','Nodos de interpolaci n','Location','northwest')
hold off
% Inciso d) Considere ahora los nodos $x_1,x_5,x_10$

data_Xmod = [-1 -0.2 0.8];
data_Ymod = [0.251083857976031,0.830828627896291,0.757200229110721];

% Nuevo polinomio:
new_p = polyfit(data_Xmod,data_Ymod,2);

xx_mod = linspace(-1,1,1e5);
```

```
62 yy_mod = polyval(new_p,xx_mod);
63
64 plot(xx,yy,xx_mod,yy_mod)
65 hold on
66 scatter(dataX,dataY,15,'filled')
67 title('Gr fica del polinomio de interpolaci n de Lagrange sobre
       [-1,1]')
68 legend('Polinomio de ajuste 11 nodos', 'Polinomio de ajuste 3 nodos
       ', ...
69     'Nodos de interpolaci n','Location','northwest')
```

```
1  Name          Size              Bytes  Class     Attributes
2
3  dataX        11x1                 88  double
4  dataY        11x1                 88  double
```




```
1 close all
```

Código de Funciones: Implementación de la función del Ejercicio 2.b

```
1 function [root,seq] = myDivision(b,x0)
2   Tol = eps;
3   seq = x0;
4   if(abs(x0.*b - 1) < Tol.*abs(x0)) root = x0;
```

```
  else
      xNew = 2.*x0 - b.* x0^2;
      seq = [seq; xNew];
      while(abs(xNew-x0)>Tol)
        x0 = xNew;
        xNew = 2.*x0 - b.* x0^2;
        seq = [seq; xNew];
      end
      root = xNew;
  end
end

% Método de la secante. Inciso 3.d

function [root,seq] = Secante(f,c0,c1)
  Tol = 1e-8;
  iterMax = 100;
  count = 0;
  f0 = f(c0);
  f1 = f(c1);
  if(abs(f0)<Tol)     root = c0; seq = c0;
  elseif(abs(f1)<Tol) root = c1; seq = c1;
  else
      seq = zeros(iterMax,1);
      cNew = c1 - f1*(c1-c0)/(f1-f0);
      fNew = f(cNew);
      seq(1) = c0;
      seq(2) = c1;
      seq(count+3) = cNew;
      while(count<iterMax && abs(c1-c0)>Tol)
        count = count + 1;
        c0 = c1;
        c1 = cNew;
        f0 = f1;
        f1 = fNew;
        cNew = c1 - f1*(c1-c0)/(f1-f0);
        fNew = f(cNew);
        seq(count+3) = cNew;
      end
      disp(cNew)
      root = cNew;
      seq = seq(1:count+3);
  end
end

% Iteraci n simple, ejercicio 4.b

```

```
function [root,seq] = iterSimple(f,x0,iterMax)
Tol = 1e-8;
count = 1;
seq = zeros(iterMax,1);
seq(1) = x0;
while(count<iterMax)
    seq(count+1) = f(seq(count));
    count = count+1;
end
root = seq(end);
end

function [root,a_seq,aux_seq] = iterModified(f,x0,iterMax)
Tol = 1e-8;
k = 1;
f0 = f(x0);
aux_seq = zeros(iterMax+3,1);
aux_seq(1:3) = [x0 f0 f(f0)];
a_seq = zeros(iterMax,1);
while(k<=iterMax)
    a_k = (aux_seq(k)*aux_seq(k+2) - aux_seq(k+1)^2)./( ...
        aux_seq(k) + aux_seq(k+2) - 2*aux_seq(k+1));
    a_seq(k) = a_k;
    aux_seq(k+3) = f(aux_seq(k+2));
    k = k+1;
end
root = a_seq(end);
end
```