MA-0501 Análisis Numérico I
TAREA 1

```
publish('T1_Murillo','format','latex','stylesheet','matlab2latex.tex')
```

Respuesta Corta

1) Para aproximar numéricamente la solución de la ecuación $|x^2 - 1| = 0$ se puede usar el método de aproximaciones sucesivas pues se puede encontrar un conjunto convexo, compacto y no vacío en R que contenga las raíces y como la función es continua, entonces existe el punto fijo por el teorema 2.5. Además, el método de bisección no es factible usarlo pues la función no tiene imagánes negativas, el de Newton tampoco pues la derivada se indefine en dos puntos, igual que con el de secante.

2) Al usar el método de Newton en la ecuación $x^{100} = 0$ se espera que este falle, pues $\frac{d(x^{100})}{dx} = 100x^{99}$, es importante notar que conforme se efectúe el proceso, como la raíz de esa ecuación es cero, la derivada se va a acercar a cero, por lo que se indefiniría.

3) Para determinar numéricamente la multiplicidad de dicho cero, basta con calcular la derivada de f, después la de f´ y así sucesivamente, la multiplicidad estara dada por la cantidad de derivadas que hemos realizado hasta que se anule al evaluar c, i.e. la primera vez que se anule una de las derivadas al evaluar en c.

4) No tiene sentido realizar mas de 52 iteraciones pues note que si realizamos 52 iteraciones el error estara dado por:

$$e_{52} < \frac{1}{2^{53}}(2-1)$$
$$\Rightarrow e_{52} < \frac{1}{2^{53}}$$

el cual es menor que el epsilon de la maquina en precision doble.

5a) Para el polinomio $p(x) = a_n x^n + ... + a_1 x + a_0$, si se desea conocer $p(x_0)$ es necesario efectuar $(n^2 + n)/2$ y n sumas.

5b) Para este algoritmo, se deben realizar en cada iteración una suma y una multiplicación, como son n iteraciones, en total se realizan n sumas y n multiplicaciones.

6) El polinomio de Lagrange para $f(x) = x^2 + 3x + 1$ es él mismo, pues es un polinomio, de igual forma si se calcula obtenemos:

$$\frac{(x-1)}{-1}\frac{(x-2)}{-2} + 5\left(\frac{x}{1}\frac{(x-2)}{1-2}\right) + 15\left(\frac{x}{2}\frac{(x-1)}{2-1}\right) = x^2 + 3x + 1$$

Desarrollo

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

```
n = 1478;
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
```

Sabemos que el término n de la sucesión de Fibonacci está dado por:

$$f(n) = \frac{1}{\sqrt{5}}\left(\phi^n - \frac{(-1)^n}{\phi^n}\right)$$

tq $\phi = \frac{1+\sqrt{5}}{2}$

Note que para n suficientemente grande la siguiente expresión tiende a 0

$$\frac{(-1)^n}{\phi^n}$$

Entonces, basta con resolver la siguiente desigualdad para n, pues en precisión doble el último valor que puede almacenar en la computadora es 2^{1023}.

$$2^{1023} < \frac{\phi^n}{\sqrt{5}}$$

Al calcular la expresión anterior se obtiene que $n > 1475,22$, i.e. n debe ser aproximadamente 1476.

1b) Para números medianos f(1) y g(1) serán diferentes debido al redondeo pues la computadora únicamente puede guardar números completos si son menores de 10^16, es por ello que para encontrar el mínimo valor de n donde f(1) y g(1) son diferentes basta con calcular:

$$10^{16} < \frac{\phi^n}{\sqrt{5}}$$

Al calcular la expresión anterior se obtiene que $n > 78,23$, i.e. n debe ser aproximadamente 79. Es importante notar que igual al ejercicio anterior despreciamos la expresión que tiende a 0.

1c)

```
%Estimacion del inciso (a)

f(1476)
```

```

ans =

   8.0776e+307
```

Note que en el inciso (a) obtuvimos que el mínimo valor en el que el algoritmo comienza a fallar es en 1476, sin embargo, esto no sucede así.

```
f(1478)
```

```

ans =

    Inf
```

No obstante, el algoritmo comienza a fallar tan solo 2 números después, en 1478.

```
%Estimacion del inciso (b)

n = 79;
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

f(1)
g(1)
```

```

ans =

     0


ans =

     0
```

Observe que ambos valores siguen siendo el mismo

```
n = 80;
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

f(1)
```

```
g(1)
```

```
ans =

     0


ans =

   8.9444e+15
```

Note que en este caso g(1) y f(1) son distintos, tan solo un numero mayor a lo calculado en el inciso (b)

2a) Como la computadora no tiene implementada la división, entonces debemos intentar encontrar otra forma de usar Newton.
Note que: $f'(x) = \frac{-1}{x^2}$, entonces podemos escribir Newton como:

$$c_k = c_{k-1} - x^2 \left(\frac{1}{x} - b \right)$$

Al simplificar obtenemos:

$$c_k = c_{k-1} + c_{k-1}(1 - bc_{k-1})$$

Para verificar las condiciones, suponga que $\lim_{n\to\infty} x_{n-1} = x$
Note que

$$\lim_{n\to\infty} x_n = \lim_{n\to\infty} x_{n-1} + x_{n-1} - bx_{n-1}^2$$

Entonces

$$x = x + x - bx^2 \Rightarrow x = \frac{1}{b}$$

Como la función es de clase C^2, $f(1/b) = 0$ y $f'(1/b) \neq 0$, por el teorema 2.1 podemos asegurar que existe un intervalo donde converge.
2b) Implementamos el algoritmo anterior:

```
function c1 = myDivision(c0, b)
c1 = c0 + c0*(1 -b*c0);
while(abs(c1-c0)\ensuremath{>}=eps)
c0 = c1;
c1 = c0 + c0*(1 -b*c0);

end
end
```

2c)

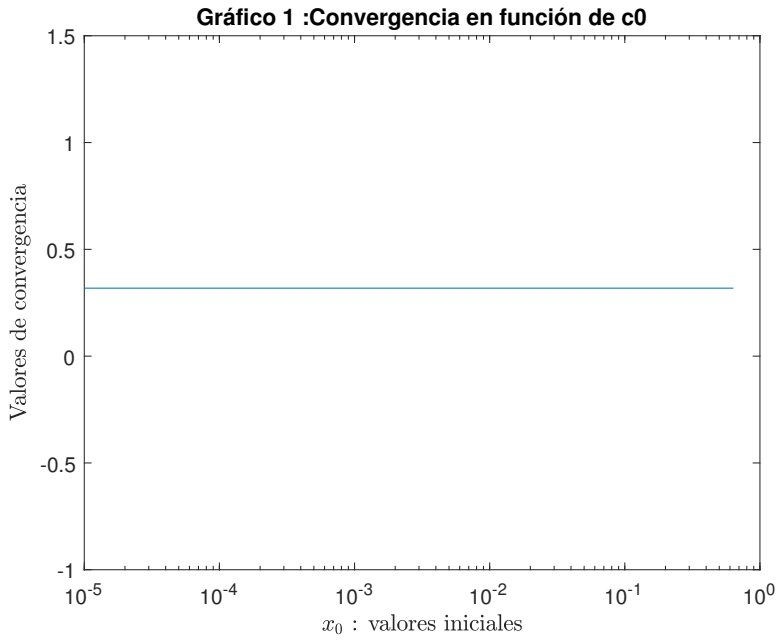
```
b = pi;
c0= linspace(0,1,1e5); %valores que toman los c0

c1 = myDivision(c0, b);

%Valor de convergencia en funcion de c0

xx = c0;
yy = nan(size(xx));

for i = 1: numel(xx)
    yy(i) = myDivision(xx(i), b);
end

figure
semilogx(xx,yy, 'MarkerSize', 20)

%Titulo
title('Gr fico 1 :Convergencia en funci n de c0')

%Nombrar ejes
xlabel('$x_0$ : valores iniciales', 'Interpreter', 'latex')
ylabel('Valores de convergencia', 'Interpreter', 'latex')
```

Tome delta de la siguiente forma:

$$\delta = \frac{1}{5}$$

Entonces, tome $I = [\frac{1}{\pi} - \frac{1}{5}, \frac{1}{\pi} + \frac{1}{5}]$

$$A = \max_{(x,y)\in I} |\frac{f''(x)}{f(y)}| = |\frac{2}{x^3 y^2}|$$



Tome $\eta = \text{mín}(1, \frac{1}{A})$, entonces podemos asegurar convergencia por el teorema 2.21 en $[\frac{1}{\pi} - \eta, \frac{1}{\pi} + \eta]$, lo cual coincide con lo graficado.
2d) Aplicaría bisección primero, para poder encontrar un x0 lo suficientemente cerca de su raíz.

3a) Note que por conmutatividad del producto se sigue que

$$\prod_{i=1}^{N_c} f(y_i, \theta) = \prod_{i}^{N_c} \binom{n_i}{y_i} \prod_{i}^{N_c} \theta^{y_i} \prod_{i}^{N_c} (1-\theta)^{n_i - y_i} \tag{1}$$

$$\prod_{i}^{N_c} \binom{n_i}{y_i} \theta^H (1-\theta)^M \tag{2}$$

donde $H = \sum_{i}^{N_c} y_i$ y $M = \sum_{i}^{N_c} n_i - y_i$.
Procedemos a calcular $l'(\theta)$ Note que

$$l(\theta) = \log(L(y_1, ..., y_N, \theta) = \log\left(\prod_{i=1}^{N_c} f(y_i, \theta)\right)$$

Por (1) y (2), se sigue que

$$\begin{aligned}\log(L(y_1,...,y_N,\theta) &= \log\left(\prod_i^{N_c}\binom{n_i}{y_i}\theta^H(1-\theta)^M\right)\\ &= \log\left(\prod_i^{N_c}\binom{n_i}{y_i}\right) + H\log(\theta) + M\log(1-\theta)\end{aligned}$$

Entonces

$$l'(\theta) = \frac{H}{\theta} - \frac{M}{1-\theta}$$

Al despejar e igualar a 0, obtenemos los puntos críticos de $l(\theta)$

$$\begin{aligned}l'(\theta) = \frac{H}{\theta} - \frac{M}{1-\theta} &= 0\\ \Rightarrow \theta &= \frac{H}{N}\end{aligned}$$

donde N = H+ M.
Además, este punto crítico en efecto es el máximo pues $l''(\theta) < 0$ y por el criterio de la segunda derivada podemos asegurar que es el máximo.

3b)

```
%Leemos el archivo
T = readtable('data.txt');
```

3c)

```
%Vectorizamos los datos
m = T.Machos;
h = T.Hembras;

%Procedemos a calcular el valor que maximiza los datos
    suministrados

%Cantidad total de machos
suma_m = sum(m);

%Cantidad total de hembras
```

```
suma_h = sum(h);

%Valor que maximiza
maximo = suma_h/(suma_m + suma_h)
```

```

maximo =

    0.4946
```

3d)

```
f = @(x) suma_h/x - suma_m/(1-x);

Secante(f,0.30,0.60)
```

```

ans =

    0.4946
```

3e)

```
[x,fval,exitflag,output,jacob] = fsolve(f,0.50);

% Resultado del fsolve
disp(x)

%Algoritmo usado
disp(output)
```

```

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.

    0.4946

       iterations: 1
        funcCount: 4
        algorithm: 'trust-region-dogleg'
    firstorderopt: 3.3382e-10
```

```
14            message: 'Equation solved.    fsolve    completed because
                  the vector of function values is near  z e r o as
                  measured by the value of the function tolerance,
                   a n d 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  t h a n 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.'
```

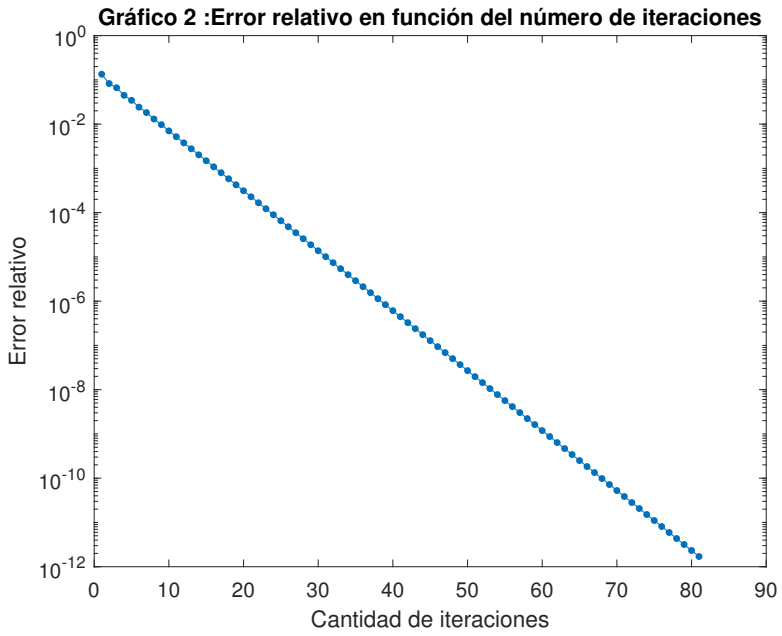
4a) Por el teorema 2.10 de las notas sabemos que basta con demostrar que una función es continua y que el valor absoluto de su derivada se puede acotar por una constante menor que 1 para que el proceso de iteración simple converja a su punto fijo. Note que $g(x) = x - (x^2 - 3)/2$ es claramente continuo pues es un polinomio. Procedemos a analizar la derivada de g Note que $|g'(x)| = |1 - x|$, la cual es una función creciente y alcanza su máximo en su extremo derecho en el valor de 1, entonces podemos asegurar que para todo $x \in (a, b)$ sucede que $|g'(x)| < 1$.

4b)

```
g = @(x) x -(x.^2 -3)./2;
raiz = 1.732050807568877; %Valor de sqrt{3} con 15 decimales

[c, secS] = iterSimple(g, 1.5);


semilogy(1:numel(secS),abs(secS-raiz)/abs(raiz),'.-','MarkerSize'
    ,10);


%Titulo
title('Gr fico 2 :Error relativo en funci n del n mero de
    iteraciones')

%Nombrar ejes
xlabel('Cantidad de iteraciones')
ylabel('Error relativo')
```

¿Cual es la pendiente de la recta en su grafica? ¿Qué representa dicha pendiente? Procedemos a calcular la pendiente de la recta

```
pendiente = ((abs(secS(36)-raiz)/abs(raiz)) - (abs(secS(30)-raiz)/
    abs(raiz)))/(36-30)
```



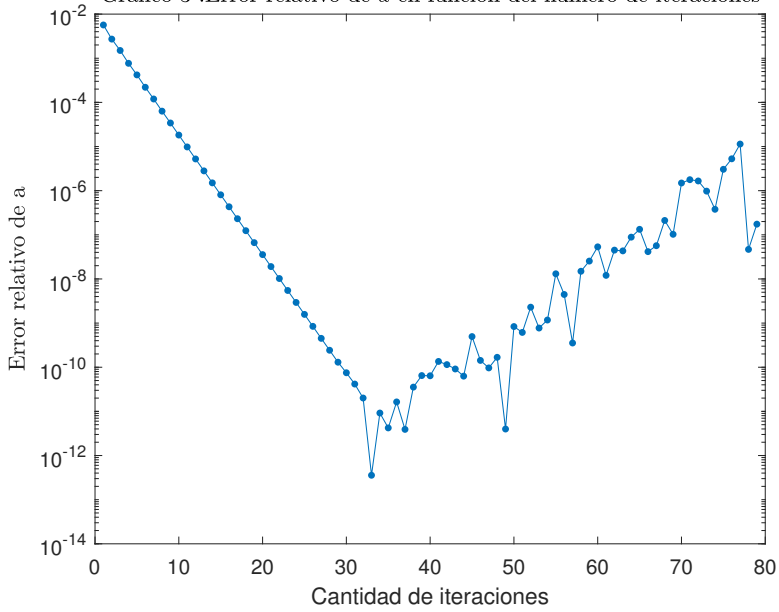

```
pendiente =

   -1.9444e-06
```

La pendiente de la recta representa la velocidad a la que disminuye el error, en este caso podemos observar que el error decrece de forma lineal.
4c)

```
a = nan(1,79);

for k = 1:79
    a(k) = ( secS(k)*secS(k+2) - (secS(k+1)^2) )/ (secS(k) + secS(k+2) - 2*secS(k+1));
end
%Graficar:

semilogy(1:numel(a),abs(raiz - a)/abs(raiz),'.-','MarkerSize',10)

%Titulo
title('Grafico 3 :Error relativo de $a$ en funcion del numero de iteraciones', 'Interpreter', 'Latex')

%Nombrar ejes
xlabel('Cantidad de iteraciones')
```

```
ylabel('Error relativo de a','Interpreter', 'Latex' )
```

Grafico 3 :Error relativo de a en funcion del numero de iteraciones



La convergencia parece ser más rápida en el 4c sin embargo, no para todas las iteraciones pues después de treinta y tres el error comienza a oscilar.
El gráfico decrece de forma lineal aproximadamente hasta la iteración treinta, después de eso oscila, después de la iteración treinta no presenta monotonía, i.e. el error decrece linealmente hasta 30 y después oscila.
5a)

```
load('dataPolin.mat')

size(dataX)
size (dataY)
```

```

ans =

    11     1


ans =

    11     1
```

5b)

```
pn = polyfit(dataX, dataY, 10)
```

```

pn =

  Columns 1 through 7

   31.2177 -120.1321  -90.8847  245.8358   90.1895 -159.6097
      -34.4710

  Columns 8 through 11

   36.0627    3.8656   -1.9053    0.5853
```

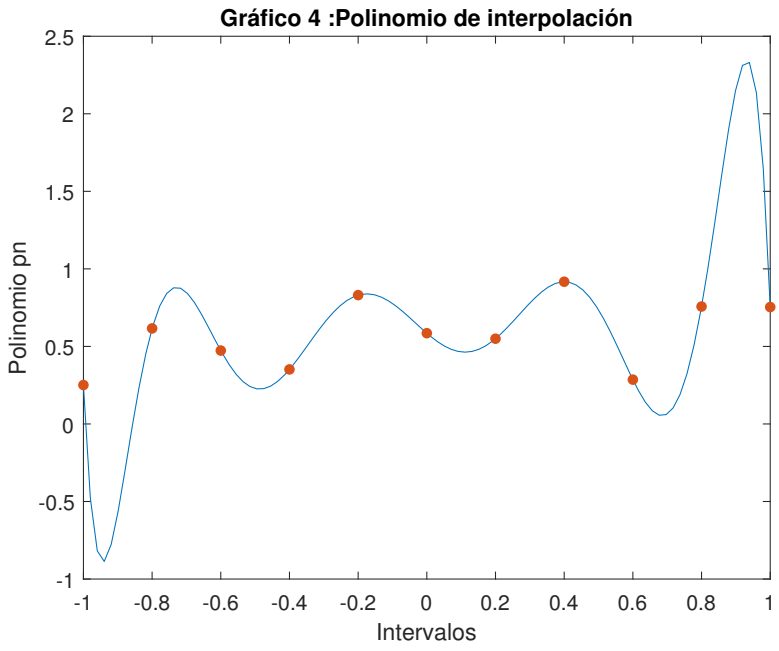
La salida pn se interpreta como los coeficientes de un polinomio de grado 10. Este debe ser de grado 10 pues hay 10 + 1 nodos.

5c)

```
xx = linspace(-1,1);
yy = polyval(pn, xx);

figure
plot(xx,yy), hold on
plot(dataX,dataY,'.', 'MarkerSize', 15)

%Titulo
title('Gr fico 4 :Polinomio de interpolaci n')

%Nombrar ejes
ylabel('Polinomio pn')
xlabel('Intervalos')
```

5d)

```
%Creamos dos vectores que contengan los datos de los nodos 1,5,10
nodos_nuevosX = [dataX(1) dataX(5) dataX(10)];
nodos_nuevosY = [dataY(1) dataY(5) dataY(10)];

pn_nuevo = polyfit(nodos_nuevosX, nodos_nuevosY, 2);
yy_1 = polyval(pn_nuevo, xx);

figure
plot(xx,yy_1), hold on
plot(nodos_nuevosX, nodos_nuevosY,'.', 'MarkerSize', 15)
```




```
11
12 %Titulo
13 title('Gr fico 5 :Polinomio de interpolaci n')
14
15 %Nombrar ejes
16 ylabel('Polinomio pn')
17 xlabel('Intervalos')
```

Cuál es mejor? No tiene sentido realizar esta pregunta pues son funciones realizadas con diferente cantidad de nodos.

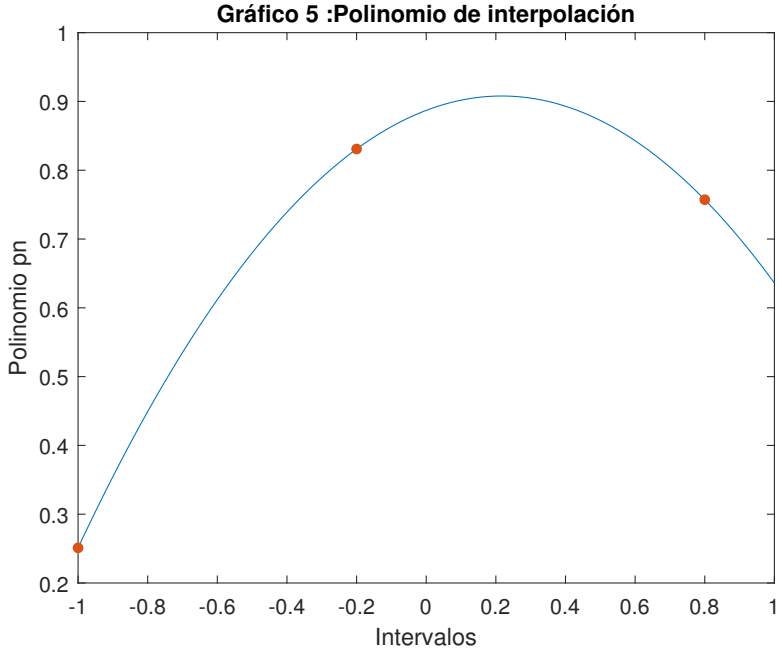
```
1 close all
```

Código de funciones

Funciones

```
1 function c1 = myDivision(c0, b)
2     c1 = c0 + c0.*(1 -b.*c0);
3
4     while(abs(c1-c0)>=eps)
5         c0 = c1;
6         c1 = c0 + c0.*(1 -b.*c0);
7
```

Gráfico 5 :Polinomio de interpolación




```
    end
end

function [root,seq] = iterSimple(f,x0)
    Tol = eps;
    iterMax = 80;
    k = 1;
    seq = [x0 f(x0)];
    while(Tol && k<iterMax)
        seq = [seq f(seq(end))];
        k = k+1;
    end
    root = seq(end);
end

function [root,seq] = Secante(f,x0,x1)
  Tol = 1e-8;
  iterMax = 100;
  count = 0;
  f0 = f(x0);
  f1 = f(x1);
  if(abs(f0)<Tol)
      root = x0; seq = x0;
  elseif(abs(f1)<Tol)
      root = x1; seq = x1;
```

```
33   else
34       seq = zeros(iterMax,1);
35       xNew = x1 - f1*(x1-x0)/(f1-f0);
36       fNew = f(xNew);
37       seq(count+1) = xNew;
38       while(count<iterMax && abs(x1-x0)>Tol)
39         count = count + 1;
40         x0 = x1;
41         x1 = xNew;
42         f0 = f1;
43         f1 = fNew;
44         xNew = x1 - f1*(x1-x0)/(f1-f0);
45         fNew = f(xNew);
46         seq(count+1) = xNew;
47       end
48       root = xNew;
49       seq = seq(1:count+1);
50   end
51 end
```