chuletacomput.pdf

Anónimo

Algorítmica Numérica

2º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

Consideramos las dos siguientes expresiones que dan resultados similares para valores de h cercanos a cero:

$$\cosh(1) \cong \frac{\cosh(1+h) - 2\cosh(1) + \cosh(1-h)}{h^2}$$

Se pide:

- Construir un vector n con valores 1, 2, 3,…, 8.
- A partir de n construir un vector h con valores 0.1, 0.01, 0.001,….., 0.00000001.
- Construir un vector con los resultados de evaluar la expresión de la derecha para el vector h.
- Calcular el error relativo de los resultados de la expresión de la derecha con respecto a la de la izquierda.
- Dibujar, en la escala adecuada, la relación entre n y el error relativo (los valores de n deben ir en el eje horizontal y los del error relativo en el vertical).
- Calcular el número de cifras decimales correctas entre ambas expresiones.
- Dibujar, en la escala adecuada, la relación entre h y las cifras (los valores de h deben ir en el eje horizontal y los de las cifras en el vertical).
- Para que valor de h se consiguen 8 cifras correctas.

Ejercicio 2.

Ejercicio 1

```
h=10.^-[1:8]
n=[1:8]
h=10.^-n
der = (cosh(1+h)-2*cosh(1)+cosh(1-h))./(h.^2)
izq = cosh(1)
er=abs(der-izq)/abs(izq)
semilogy(n,er)
figure(1),semilogy(n,er)
cif = -log10(er)
figure(2),semilogx(h,cif)
% No se consiguen 8 cifras. Lo más cercano (casi 8) es para h = 10^ -4 que es
0.0001
```

Ejecutar los comandos xi=pi*[-1.5:1:1.5]',yi=sin(xi). Sea la tabla de datos $\{\text{xi}, \text{yi}\}$.

a) Interpolar la tabla de datos mediante un polinomio de grado mínimo.

Dibujar la gráfica del polinomio de interpolación en el intervalo [-5, 5] ('b.' en azul) junto con los valores de la tabla ('ro' en rojo).

b) Interpolar la misma tabla de datos con un polinomio que verifique la condición $p(0)=1$.
Dibujar la gráfica del polinomio en el intervalo [-5, 5] ('g.' en verde), con los valores de la tabla (en rojo), junto con el punto $(0,1)$ ('sr' cuadrado rojo).

c) Ajustar los datos de la tabla con un polinomio de grado 3 que verifique la condición $p'(0)=1$.
Dibujar la gráfica del polinomio de ajuste en el intervalo [-5, 5] ('k.' en negro), con los valores de la tabla (en rojo). Calcular el vector residuo y el error del ajuste.

Dibujar en una misma gráfica los tres polinomios obtenidos en los apartados anteriores, junto con los valores de la tabla y el punto $(0,1)$.

```
clear
xi=pi*[-1.5:1:1.5]',yi=sin(xi)

a)
H=[xi.^0 xi.^1 xi.^2 xi.^3];b=yi;c=H\b;
xx=-5:0.01:5;px=c(1)+c(2)*xx+c(3)*xx.^2+c(4)*xx.^3;
plot(xi,yi,'ro',xx,px,'b.')

b) Opción 1

xi(5)=0,yi(5)=1,
H21=[xi.^0 xi.^1 xi.^2 xi.^3 xi.^4];b21=yi;c21=H21\b21;
px21=c21(1)+c21(2)*xx+c21(3)*xx.^2+c21(4)*xx.^3+c21(5)*xx.^4;
plot(xi,yi,'ro',xx,px,'b.',xx,px21,'g.',0,1, 'rs')

b) Opción 2
% p(0)=1, a=1, px=1+bx+c x^2+d x^3+e x^4;
H2=[xi.^1 xi.^2 xi.^3 xi.^4];b2=yi-xi.^0;c2=H2\b2;
px2=1+c2(1)*xx+c2(2)*xx.^2+c2(3)*xx.^3+c2(4)*xx.^4;
plot(xi,yi,'ro',xx,px,'b.',xx,px2,'g.',0,1, 'rs')

c)
% p'(0)=1, b=1, px=a+x+c x^2+d x^3;
H3=[xi.^0 xi.^2 xi.^3];b3=yi-xi;c3=H3\b3;
px3=c3(1)+xx+c3(2)*xx.^2+c3(3)*xx.^3;
plot(xi,yi,'ro',xx,px,'b.',xx,px2,'g.',0,1, 'rs',xx,px3,'k.')
r=c3(1)+xi+c3(2)*xi.^2+c3(3)*xi.^3-yi
E=sum(r.^2)
```

Dada la función $f(x) = x^2 - e^{-x}$.

a) Realizar una gráfica de la función f(x) en el intervalo [-2, 2]. Dar un intervalo aproximado con una longitud máxima de 1 donde se encuentre la raíz a partir de la gráfica. Demostrar analíticamente que en dicho intervalo existe al menos una raíz.

b) Implementar y ejecutar el siguiente método $x_{n+1} = \sqrt{e^{-x_n}}$ para encontrar la raíz de f(x) partiendo de $x_0 = 1$ El método deberá iterar hasta que el error $e_n \approx |x_n - x_{n-1}| < 1e-10$. El código deberá en cada iteración imprimir, el número de iteración, la raíz obtenida y el error utilizando el siguiente formato:

'Iter %d Sol %.15f Error %e\n'

c) Implementar y ejecutar el siguiente método

$$z_0 = a, z_{n+1} = z_n - \frac{f(z_n)}{f'(x_n)}$$
$$x_0 = b, x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

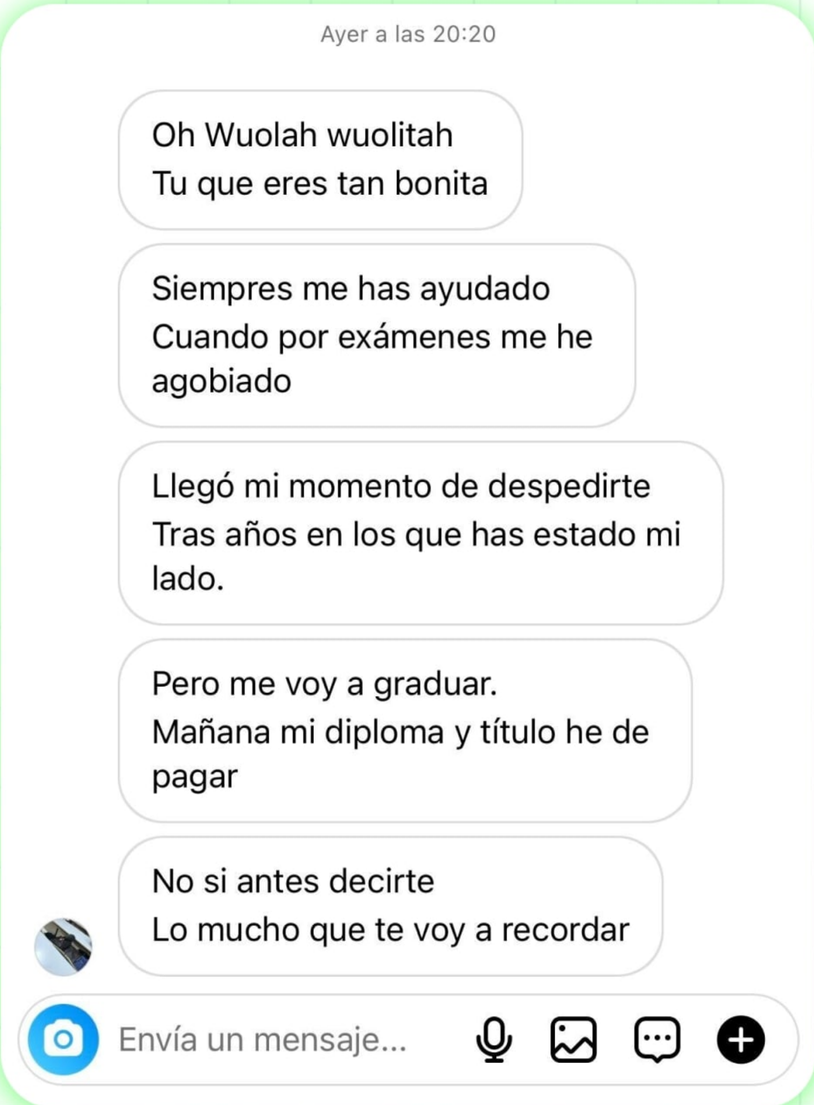
que calculará la raíz x_n de f(x) partiendo de $a = 0.5, b = 1.0$ e iterando hasta que el error $e_n \approx |x_n - z_n| < 1e-10$. El código deberá en cada iteración imprimir, el número de iteración, la raíz obtenida y el error utilizando el siguiente formato:

'Iter %d Sol %.15f Error %e\n'

d) A la vista de los resultados obtenidos con ambos métodos: ¿Cuál es el orden de convergencia de cada método? Justificar la respuesta.

Que no te escriban poemas de amor cuando terminen la carrera 🚩🚩🚩🚩🚩🚩

(a nosotros por suerte nos pasa)



```
x=(-2:0.01:2);
fx=x.^2-exp(-x);
plot(x,fx,x,fx*0)
f0=0.^2-exp(-0);
f1=1.^2-exp(-1);
f0*f1

error=100.0;
iteracion=0;
xn=1.0;
while (error>1e-10)
    iteracion=iteracion+1;
    xn1=sqrt(exp(-xn));
    error=abs(xn1-xn);
    fprintf('Iter %d Sol %.15f Error %e\n', iteracion,xn1,error)
    xn=xn1;

end

fprintf('\n \n')
z=0.5;
x=1.0;
iteracion=0;
error=100.0;
while(error>1e-10)
    iteracion=iteracion+1;
    z=z-((z^2-exp(-z)) / (2*x+exp(-x)));
    x=x-((x^2-exp(-x)) / (2*x+exp(-x)));
    error=abs(z-x);


fprintf('Iter %d Sol %.15f Error %e\n', iteracion,x,error)
```

2. - Calcular la función u(x) del tipo dado que mejor ajusta (sentido de mínimos cuadrados) todos los datos de la tabla dada.
- Calcular el vector *res* de residuos y la norma *e* del error que produce dicho ajuste ¿En qué punto se produce la máxima desviación y cuánto vale?
- Superponer a la gráfica del apartado anterior, la gráfica de la nueva función pedida (en rojo) en el intervalo [0, 0.8] y los puntos donde se realiza el ajuste (usando el símbolo * rojo).

2. Ajuste por función u(x). Código para calcular coeficientes, residuos, error, máximo residuo y gráfica
Incluir valores y gráfica pedida en tabla adjunta.

```
% Sistema sobredeterminado a resolver:
H=[x.^0 sin(pi*x) sin(2*pi*x)];
c=H\y;
% Residuos. Error
res=abs(c(1)+c(2)*sin(pi*x)+c(3)*sin(2*pi*x)-y);
e=norm(res);
[resmax,m]=max(res)
% Gráfica
xx=0:0.01:0.8;
yyi=ci(1)+ci(2)*sin(pi*xx)+ci(3)*sin(2*pi*xx);
yy=c(1)+c(2)*sin(pi*xx)+c(3)*sin(2*pi*xx);
plot(xx,yyi,'g',xi,yi,'go', xx,yy,'r', x,y,'*r')
```

Ejercicio 2. Se quiere calcular la solución que la ecuación x-log(x)=2 tiene en el intervalo [3,4].

- Comprobar que la función f(x)=x-log(x)-2 tiene una raíz en el intervalo [3,4].

- Programar un bucle que aplique 7 iteraciones del método de Newton para estimar dicha raíz partiendo del valor inicial x_0=3. Para ello crear un vector x=zeros(8,1) con x(1)=x_0=3 y guardar en él los resultados de las iteraciones. Sea s=x(end) ¿es s la raíz de f(x)? Comprobarlo.

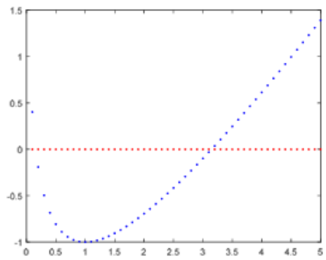
- A partir del vector x, calcular el vector *Erel* con los errores relativos (de cada iteración) con respecto a s. Calcular el vector *Ncifras* que contiene el nº de cifras significativas de precisión estimadas para cada iteración.

- Dibujar los vectores *x*, *Erel* y *Ncifras* respecto del nº de iteración en tres gráficas independientes usando en cada caso la escala adecuada.

- Utilizando únicamente los resultados numéricos obtenidos, responder a las siguientes preguntas justificando las respuestas:

- ¿Cuál sería el nº mínimo de iteraciones que produciría la precisión final obtenida?
- ¿Cuál es la velocidad (el orden) de convergencia del método (lineal, cuadrática,...)?
- ¿Con cuántas iteraciones se obtienen al menos 5 cifras de precisión?
- Los valores s y x(4) ¿cuántas cifras significativas de precisión coincidentes tienen?
- Con el comando fprintf mostrar los valores de s y x(4) y contad las cifras coincidentes ¿cuántas son?

```
clear;
x=0.1:0.1:5;fx=x-log(x)-2;plot(x,fx,'.b',x,0,'.r')
x=3;f3=x-log(x)-2,x=4;f4=x-log(x)-2,f3*f4
```



La función f(x) tiene una raiz en el interval [3,4]: es una función continua y f(3)*f(4) < 0.

```
x=zeros(8,1);x(1)=3;
for k=1:7
  x(k+1)=x(k)*(1+log(x(k)))/(x(k)-1),
end
subplot(131);plot(x,'bo');
s=x(end),fs=s-log(s)-2
Erel=abs(x-s)/abs(s);
```

```
Ncifras=floor(-log10(Erel))
subplot(132);semilogy(Erel,'bo');
subplot(133);plot(Ncifras,'ro')
```

Ejercicio 3. Dada la ecuación $f(x) = x^2 - e^{-x} = 0.$

a) Realizar una gráfica de la función f(x) en el intervalo [-2, 2]. A partir de la gráfica, determinar un intervalo de longitud 1 donde se encuentre la raíz. Demostrar analíticamente que en dicho intervalo existe al menos una raíz.

b) Método 1: Implementar y ejecutar el método $x_{n+1} = \sqrt{e^{-x_n}}$ para encontrar la raíz de f(x) partiendo de $x_0 = 1$ El método debe iterar hasta que el error $e_n \approx |x_n - x_{n-1}| < 1e-10$. El código deberá en cada iteración imprimir el número de iteración, la aproximación de la raíz obtenida y el error, utilizando el formato fprintf(Iter %d Sol %.15f Error %.2e\n',.....).

c) Método 2: Implementar y ejecutar el siguiente método

$$z_0 = 0.5, x_0 = 1, \begin{cases} z_{n+1} = z_n - \dfrac{f(z_n)}{f'(x_n)} \\ x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)} \end{cases}$$

que calculará la solución aproximada x_n de la raíz, iterando hasta que el error $e_n \approx |x_n - z_n| < 1e-10$. El código deberá en cada iteración imprimir: el número n de iteración, la solución x_n obtenida y una estimación del error utilizando el formato 'Iter %d Sol %.15f Error %e\n'

```
% Visualizar y comprobar hay raíz en [-2,2]
x=(-2:0.01:2); fx=x.^2-exp(-x);
plot(x,fx,x,fx*0)
f0=0.^2-exp(-0); f1=1.^2-exp(-1);f0*f1


% Método 1
error=100.0;iteracion=0;xn=1.0;
while (error>1e-10)
    iteracion=iteracion+1;
    xn1=sqrt(exp(-xn));
    error=abs(xn1-xn);
    fprintf('Iter %d Sol %.15f Error %e\n', iteracion,xn1,error)
    xn=xn1;
end
fprintf('\n \n')
% Método 2
z=0.5;x=1.0;iteracion=0;error=100.0;
while(error>1e-10)
    iteracion=iteracion+1;
    z=z-((z^2-exp(-z)) / (2*x+exp(-x)));
    x=x-((x^2-exp(-x)) / (2*x+exp(-x)));
    error=abs(z-x);
    fprintf('Iter %d Sol %.15f Error %e\n', iteracion,x,error)
end
```

Ejercicio 5. Se considera la función f(x)=x-exp(-x)-1.

1. Verificar y visualizar que dicha función tiene una raíz en el intervalo [1 , 2] .
2. Aplicar el método de Newton para calcular dicha solución iterando 8 veces y tomando como valor inicial el punto medio del intervalo. El código empleado debe proporcionar: el número de iteración k (%d), la solución x_k obtenida en cada iteración con 16 decimales (%.16f), y una estimación del error e_k cometido en cada iteración (formato: %5.2e). Incluir código y resultados pedidos.

Nota: Para estimar el error e_k en iteración k-ésima se puede utilizar $e_k \approx |x_{k+1} - x_k|$.

Primero creamos la función 'fun7' que evalúa la función y su derivada en x y nos será de utilidad a lo largo del ejercicio:

```
function [f fp]= fun7(x)
f = x-exp(-x)-1;
fp=1+exp(-x);
end
```

<u>Verificación y visualización existencia raiz en intervalo [1,2]:</u>

La función f(x) dada es continua y comprobamos que distinto tiene signo en los extremos del intervalo:

```
>> fun7(1)*fun7(2)
ans =    -0.3181
```

Por tanto, aplicando el Teorema de Bolzano, concluimos que la función f(x) tienen al menos una raiz en el intervalo [1,2].

Pintamos la función f(x) y el eje x en el intervalo [1,2], para visualizar la raiz de f(x) en ese intervalo:

```
xx=1:0.001:2;
yy=fun7(xx);
plot(xx,0*xx,'-k',xx,yy,'-r')
```

<u>Código método Newton y aplicación:</u>

- Función que implementa el método de Newton:

```
function s=newton(fx,x0,N)
% Entrada  puntero a fx  nombre de la función (debe devolver f(x) y f'(x))
%          x0  punto inicial
%          N   N max iteraciones.
  % Inicio
for k=1:N,
  [f fp]=fx(x0); % Pido valores de f(x) y f'(x)
  if f==0, break; end  % Ya estoy en la raiz
  x = x0-f/fp;    % Iteracion de Newton.
  e=abs(x-x0);     % Estimación del error en cada iteración.
  x0=x;
  fprintf('k: %d -> x_k:%.16f e_k:%5.2e\n',k,x,e);  % Vuelco valores.

end
 s = x;  % Devuelvo último término de la sucesión.
end
```

Ejercicio 7. Aplicar el método de la bisección para calcular la raíz de la ecuación del Ejercicio 3 y comparar los resultados con los obtenidos mediante el método de Newton.

1. Crear la función biseccion.m que implementa el cálculo de N iteraciones (4er argumento de entrada) del método de la bisección aplicado a la función fun (1er argumento de entrada) empezando en el intervalo [a,b] (2º y 3er argumentos de entrada):

```
function s=biseccion(fun,a,b,N)     % Método de la bisección.
% Argumentos Entrada: fun puntero a la función;  [a,b] intervalo donde está la solución
%  N: número máximo de iteraciones
% Argumento Salida:   s es la aproximación de la raiz
fa=fun(a);fb=fun(b); % Evalúo la función f en a y en b.
if fa*fb > 0
   fprintf('Método no aplicable en intervalo [a,b]\n')
 end
for i=1:N
   c=(a+b)/2
   fc=fun(c);    %evalúo la función en punto medio intervalo
   if fa*fc < 0
      b=c;     %fb=fc; (no es necesario)
   else
      a=c;
      fa=fc;
   end
end

      s=a % pueder ser s=b o s=(a+b)/2,
      fprintf('La raiz aproximada es %12.8f\n',s)
      end
```

- Aplicar en el intervalo [0,1] para que itere 10 veces ¿Cuál es el error de la última solución?

2. Modificar el código anterior para que el <u>método itere hasta calcular la solución con una precisión</u> (tol) dada (error<=tol) y en todo caso el número de iteraciones no sea mayor que N. Como argumentos de salida se espera que proporcione la solución aproximada s y el nº n de iteraciones realizadas

```
function [s,n]=biseccion2(fun,a,b,tol,N)
% Entrada: como en función anterior, se añade tol  precisión deseada.
%          N: nº max de iteraciones (opcional, por defecto 20)
% Salida : s, estimación de la raíz y  n, número iteraciones realizadas.
s=NaN;n=NaN;
if nargin==4, N=20; end
fa=fun(a); fb=fun(b);  % Evaluación de fx en extremos intervalo
if (fa*fb>0), fprintf('ERROR: Método no aplicable en intervalo [a,b]\n');  end
n=1;
while ( ((b-a)/2 > tol) & (n<=N) )  % Condiciones salida
s = (a+b)/2;
fs=fun(s);    % Evalúa la función en la estimación de la raiz
if (fs*fa <0), b=s; fb=fs; else a=s; fa=fs; end
n=n+1;   end
s = (a+b)/2;     % Mejor hipótesis dado el intervalo final [a,b]
end
```

Ejercicio 7:

- Aplicamos método de la bisección para calcular la solución con un error menor o igual que 10^(-10), iterando un nº máximo de 20 iteraciones :

```
>>[s,n]=biseccion2('fun1',0,1,1e-10,20)
s =   7.3908e-01 (Solución aproximada)
n =   21        ¿Se ha calculado la solución con la precisión pedida?
```

- Comprobar que s es efectivamente la raíz de f(x). ¿Cuál es el valor de f(s)?

```
>> fs=fun1(s)
fs = -6.9054e-007
```

- Volver a calcular la raiz para obtener un mejor valor aproximado de s que verifique abs(f(s))<=10^(-15):

```
>> [s,n]=biseccion('fun1',0,1,1e-15,100)
   s =   0.7391      n =   50

>> fs=fun1(s)      fs = -1.1102e-015
```

- Modificar la función biseccion2 para que en cada iteración, usando fprintf() listar la estimación de la raiz (x_n = punto medio del intervalo en el que estemos) con 16 decimales (%.16f). Listar también el valor absoluto de f(x_n), y una cota del error (b-a)/2, ambas con el formato %5.2e.

```
function [s,n]=biseccion(fx,a,b,tol,N)
% Entrada: fx string con nombre de la función cuyo cero buscamos
%         I  intervalo [a b]  conteniendo raíz
%         tol  precisión deseada.
%         N num max de iteraciones (opcional, por defecto 20)
% Salida : s, estimación de la raíz y  n, número iteraciones realizadas.
s=NaN;n=NaN;
if nargin==3, N=20; end
fa=feval(fx,a); fb=feval(fx,b);  % Evaluación de fx en extremos intervalo
if (fa*fb>0), fprintf('ERROR: La función no tiene raíz simple en el intervalo\n'); return; end
n=1;
while ( ((b-a)/2 > tol) & (n<=N) )  % Condiciones salida
s = (a+b)/2; fs=feval(fx,s);
fprintf('v_aprox:%.16f abs(f(v_aprox)):%5.2e error:%5.2e\n',s,fs,(b-a)/2);
if (fs*fa <0), b=s; fb=fs; else a=s; fa=fs; end
```

b) Metodo iterativo 1:

- Implementar y ejecutar el siguiente método para encontrar la raíz de f(x):

$$\boxed{x_{n+1} = g(x_n) = \sqrt{e^{-x_n}} \quad \text{con } x_0 = 1}$$

- El método deberá iterar hasta que el error $\boxed{e_n = |x_n - s| \approx |x_n - x_{n-1}| < 1e-10}$

- En cada iteración imprimir: número de iteración, la raíz obtenida y el error (con el formato 'Iter %d Sol %.15f Error %0.2e\n')

```
error=100.0;                                   % Control.
iteracion=1;                                   % Contador iteraciones, iniciar
x0=1.0;                                        % Valor inicial
while (error>1e-10)                            % Criterio parada
  x1=sqrt(exp(-x0));                           % Método iterativo
  error=abs(x1-x0);                            %Estimación error
  fprintf('Iter %d Sol %.15f Error %0.2e\n', iteracion,x1,error)   %Imprimir soluciones aprox,...
  iteracion=iteracion+1;
  x0=x1;
end
s=x0                                           % Solución final
```

ALGORITMO:

```
function s = biseccion ( fx,a,b,N)

% Evaluamos la función fx (previamente creada) en a y en b:
fa=feval(fx,a);fb=feval(fx,b);
if fa*fb > 0
   fprintf('La función no tiene raices simples en el intervalo [a,b]\n')
end
for i=1:N
   c=(a+b)/2
   fc=feval(fx,c);
   if fa*fc < 0
      b=c;
   else
      a=c;
      fa=fc;  % Ahorramos una evaluación en la siguiente iteración
   end
end
s=a   % s solución aproximada, podría ser s=b ó s=(a+b)/2,
fprintf('La raiz aproximada es %12.8f\n',s)
end
```

Argumentos entrada:

- nombre de la función fx (previamente creada)
- [a,b] intervalo que contiene solución
- N es el numero máximo de iteraciones

Argumento salida:

- s: aproximación de la raiz

Observaciones:

- En lo anterior se supone que previamente se ha construido un afunción fx.m que evalúa la función f(x) a la que se va a aplicar el método de la bisección.

- Variantes: Introduciendo criterio parada en función de la precisión pedida.

- Evaluar f(x_k)

x_i	-1	0	1	2	3	4	5
y_i	0	1	2	1	0	1	3

$u(x) = a + b\cos(x) + c\sin(x)$

```
xi=[-1:5]'
yi=[0 1 2 1 0 1 3]'
H=[ones(size(xi)) cos(xi) sin(xi)]
c=H\yi
xx=-1.5:0.01:5.5;
yy=c(1)+c(2)*cos(xx)+c(3)*sin(xx);
```
no puedo usar polyval no es polinomio

```
plot(xx,yy,'g',xi,yi,'ro')
```

$\boxed{Hc=b}$ esto es lo q resolvemos en un problema sin pesos

Pesos 0.1 0.1 0.1 1 1 1 1

```
wi=[0.1 0.1 0.1 1 1 1 1]
```

con pesos $y_1 = a + b\cos 1 + c\sin 1 = 2$

$\sqrt{0.1} + b\sqrt{0.1}\cos 1 \ldots = 2\sqrt{0.1}$

~~diag(wi)~~

```
D=diag(sqrt(wi))
cp=(D*H)\(D*yi)
yp=cp(1)+cp(2)*cos(xx)+cp(3)*sin(xx);
hold on
plot(xx,yp,'b')
```

Residuos

```
zi=cp(1)+cp(2)*cos(xi)+cp(3)*sin(xi)
plot(xi,zi,'y*')
yi-zi
```