

# Algoritmos Numéricos por Computadora

## Examen final - Primavera 2023

Sube tu archivo resultado a Canvas → Examen final antes de las 14:30 horas.

**Aseguráte que tus resultados se despliegan en el archivo que subas.**

**Al final del script debes incluir las funciones usadas para resolver las preguntas (sin comentarios).**

Cada pregunta vale 5 puntos.

Los exámenes son trabajos individuales. Está estrictamente prohibido dar o recibir ayuda de cualquier persona.

El artículo 29 del Reglamento de Alumnos establece que "se calificará como no acreditada (N.A.) cualquier materia cuando el alumno incurra en alguna práctica fraudulenta".

### 1. Identificar los errores numéricos de las soluciones computacionales. Velocidad y exactitud.

Utilizando los métodos de Euler y RK4, encuentra la solución numérica del problema con valor inicial

$$\frac{dy}{dt} = 3 - 2t - 0.5y \quad ; \quad y(0) = 1$$

en el intervalo  $tspan = [0, 1]$ .

```
h = 0.00001;
f = @(t,y) 3-2*t-0.5*y;
y0 = 1;
t0 = 0;
tf = 1;
```

1) Resuelve el problema con el método de Euler y  $h=0.00001$ , guarda los resultados en la variable  $yE$  y mide el tiempo de ejecución.

```
%odeEuler(f, to, yo, h, tf)
tic
[t1, yE] = odeEuler(f, t0, y0, h, tf);
toc
```

Elapsed time is 0.009482 seconds.

2) Resuelve el problema con el método RK4 y  $h=0.00001$ , guarda los resultados en la variable  $yRK4$  y mide el tiempo de ejecución.

```
%RK4(f,t0,y0,h,tf)
tic
[t2, yRK4] = RK4(f, t0, y0, h, tf);
toc
```

Elapsed time is 0.018316 seconds.

### 3) ¿Cuál método es más rápido? ¿Por qué?

```
%Es más rápido el método de Euller porque SOLO evalúa una vez la función,  
%cuando el método RK4 la evalúa 4 veces.
```

Resuelve nuevamente el problema con ambos métodos pero ahora con h=0.1 y sin medir tiempos.

```
h = 0.1;  
[t1, yE] = odeEuler(f, t0, y0, h, tf);  
[t2, yRK4] = RK4(f, t0, y0, h, tf);
```

La solución exacta del problema es

$$y = 14 - 4t - 13 \exp\left(-\frac{t}{2}\right)$$

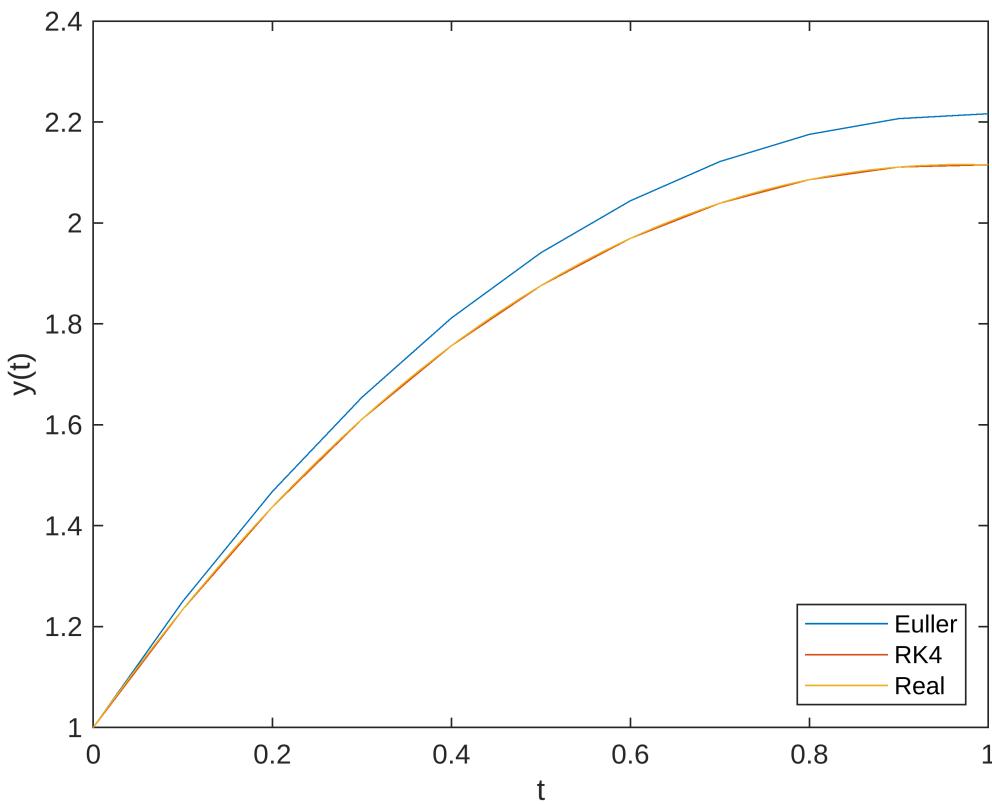
### 4) Gráfica en una misma figura los valores de $y$ , $yE$ y $yRK4$ .

```
syms y(t)  
y(t) = 14 - 4*t-13*exp(-t/2)
```

$y(t) =$

$$14 - 13 e^{-\frac{t}{2}} - 4 t$$

```
plot(t1,yE)  
hold on  
plot(t2, yRK4)  
f2 = matlabFunction(y(t));  
fplot(f2,[t0 tf]);  
hold off  
legend("Euller", "RK4", "Real", Location="southeast")  
ylabel("y(t)")  
xlabel("t")
```



5) ¿Cuál método tiene más error? ¿Por qué?

```
%Notamos que el método RK4 se aproxima casi perfectamente a la solución
%simbólica (exacta) esto se debe a que el RK4 viene de un cuarto orden de
%las aproximaciones de la serie de Taylor. Cuando Euller en cambio solo
%es de primer orden.
```

## 2. Solucionar de forma numérica sistemas de ecuaciones diferenciales. Péndulo.

La ecuación del movimiento de un péndulo se escribe en forma de ecuación diferencial de la siguiente manera

$$\frac{d^2}{dt^2}\theta + \frac{g}{l} \sin \theta = 0$$

Las condiciones iniciales están dadas por

$$\theta(0) = \frac{\pi}{10} \text{ y } \frac{d}{dt}\theta(0) = 0.$$

- 1) A partir de la ecuación de segundo orden, define la variable  $y$  (vector columna) y la función  $f(t, y)$  de tal manera que la ecuación se transforme en dos ecuaciones de primer orden.

$$y' = f(t, y)$$

```

g = 9.81;
l = 1;
f = @(t,theta) [theta(2); -g/l*sin(theta(1))];

```

2) Resuelve el sistema de dos ecuaciones de primer orden usando el método RK4 en el intervalo tspan = [0, 10].

```

h = 0.1;
t0 = 0; tf = 10; theta_0 = [pi/10; 0];
[t, theta] = RK4(f, t0, theta_0, h, tf);

```

3) Obten el vector *theta* que contiene la parte de la solución que corresponde al ángulo  $\theta$ .

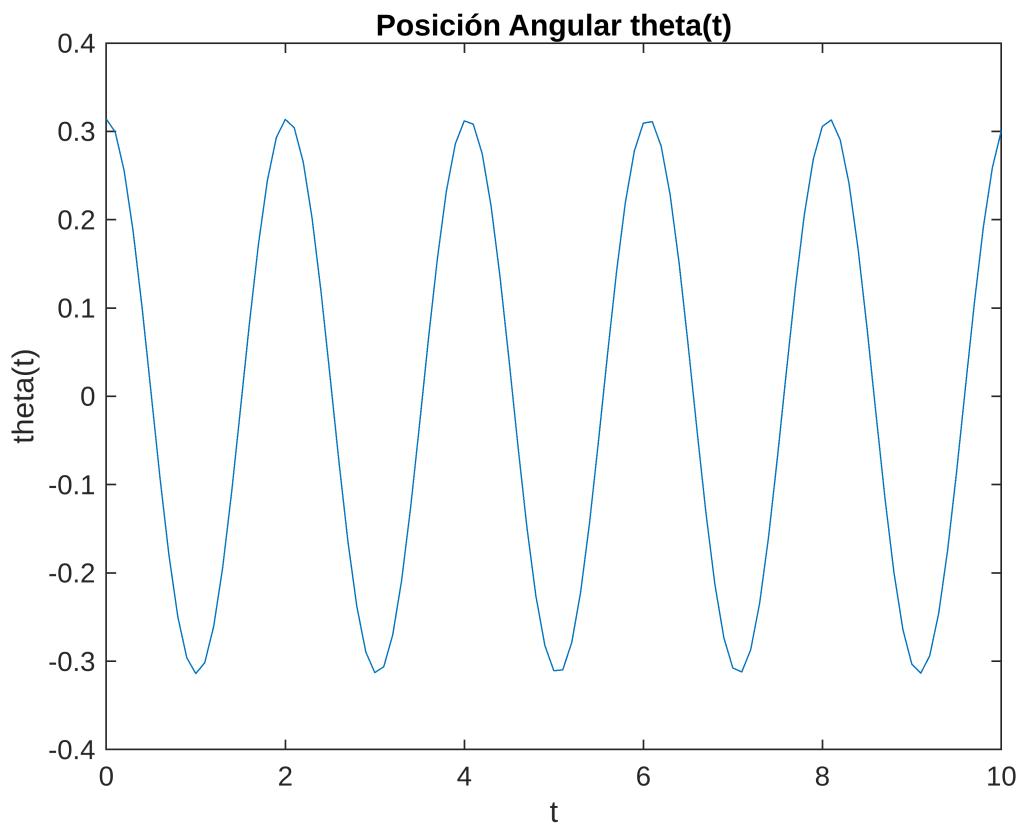
```
posAngular = theta(1, :);
```

4) Grafica, solamente,  $\theta(t)$ .

```

plot(t, posAngular)
title("Posición Angular theta(t)"); xlabel("t"); ylabel("theta(t)")

```



### 3. Utilizar de manera eficiente un lenguaje de programación matricial.

Calcula el periodo de oscilación T del péndulo de la pregunta anterior.

- 1) Usa la función *sign* de MATLAB para convertir el vector *theta* en un vector (que llamaremos *signo*) de unos y menos unos. Y Usa la función *diff* de MATLAB sobre el vector *signo* calculado para obtener el vector *cambio*.

```
signo = sign(posAngular);  
cambio = diff(signo);
```

- 2) ¿Qué significa que *cambio(i)==2*?

```
%Significa que completamos un periodo
```

- 3) Encuentra los valores del vector *tiempo t* donde *cambio==2*

```
t2 = t(cambio==2);
```

- 4) Calcula el promedio de la diferencia de los tiempos calculados en el inciso anterior.

```
prom = mean(diff(t2))
```

```
prom = 2
```

- 5) En una sola línea, a partir del vector *theta*, calcula el periodo T del péndulo.

```
T = mean(diff(t(diff(sign(posAngular))==2)))
```

```
T = 2
```

## 4. Entender el funcionamiento de métodos numéricos. BVP.

El balance de calor de estado estacionario de una barra se representa como

$$\frac{d^2}{dx^2}T - 0.15T = 0$$

Obten la solución  $T(x)$  para una barra de 10m con  $T(0) = 240$  y  $T(10) = 150$  (grados).

- 1) Pega aquí tu demostración de cómo convertir este problema en un sistema de ecuaciones lineales (2 puntos).

Mauricio Vorlego Chávez - 195106

$$\text{Sea } \tilde{y}(t) = g_{2(t)} y(t) + g_{1(t)} y_{(t)} + g_{0(t)} \quad (2)$$

Si sea la aproximación de la segunda derivada el sistema de diferencias contractas con un error  $h^2$ , tenemos que  $\tilde{y}$  se aproxima como

$$\tilde{y}(t) \approx \frac{y(t+1) - 2y(t) + y(t-1)}{h^2}, \quad \text{y también}$$

$$y(t) = \frac{y(t+1) - y(t-1)}{2h} \quad (3)$$

entonces escribimos (1) como, al sustituir (3).

$$g'(t) = g_2(t) \cdot \frac{y(t+1) - y(t-1)}{2h} + g_1(t) y(t) + g_0(t) \quad (4)$$

luego tendremos (2) y

$$\frac{y(t+1) - 2y(t) + y(t-1)}{h^2} = g_2(t) + g_1(t) \cdot y(t) + g_0(t) - \frac{g(t+1) - g(t-1)}{2h} \quad (5)$$

Despejando

$$-(1 - 0.5h g_0(t)) y(t+1) + (1 - 2 + h^2 g_1(t)) y(t) + (1 + 0.5h g_2(t)) g(t-1) = h^2 g(t) \quad (5)$$

Notamos que tienen un sistema de ecuaciones  $n-2$  ecuaciones y  $n-2$  incógnitas con  $i = 1, 2, 3, \dots, n-1$ . Notamos que el sistema de ecuaciones es particular pues si tomamos "Nodos" a través de la diagonal, notamos que los únicos diferentes distintos a cero, son aquellos próximos en una coordenada. Y podemos afirmar un sistema "triangular", en el que

$$\begin{aligned} \text{diagonal Principal: } & g_{(i)} = -(\gamma_2 + \gamma_{1(i)})h^2 = \gamma \\ \text{diagonal inferior: } & g_{(i-1)} = (\gamma_1 + \gamma_{2(i-1)}) \cdot h^2 = \beta \\ \text{diagonal Superior: } & g_{(i+1)} = (\gamma_1 + \gamma_{2(i+1)}) \cdot h^2 = \alpha \end{aligned}$$

Entonces tenemos algo de la forma:

$$\underbrace{\begin{bmatrix} \alpha & \gamma & 0 & 0 & 0 & 0 \\ \beta & \alpha & \gamma & 0 & 0 & 0 \\ 0 & \beta & \alpha & \gamma & 0 & 0 \\ 0 & 0 & \beta & \alpha & \gamma & 0 \\ 0 & 0 & 0 & \beta & \alpha & \gamma \\ 0 & 0 & 0 & 0 & \beta & \alpha \end{bmatrix}}_A \begin{bmatrix} y \\ y \\ y \\ y \\ y \\ y \end{bmatrix} = \begin{bmatrix} b \\ b \\ b \\ b \\ b \\ b \end{bmatrix}$$

$\Rightarrow Ay = b$  ó  $\sum \beta_i y_i + g_{(i)} y_{i+1} = b_{(i)}$

en el que la conocemos A entonces al despejar para  $b_i$ , tenemos que conocemos el valor final e inicial de  $y_i$ , los llamaremos  $y_a$  y  $y_b$ , respectivamente. Recordando  $i^* = 1, 2, 3, \dots, n$ . Y continuando en el despejio de las ec. 5, tenemos:

$$b(1) = h^2 * g_{(1)} - y_a * \left(1 + \frac{1}{2} \alpha + \gamma_2\right)$$

$$b(n) = h^2 * g_{(n)} - y_b * \left(1 - \frac{1}{2} \alpha + \gamma_2\right)$$

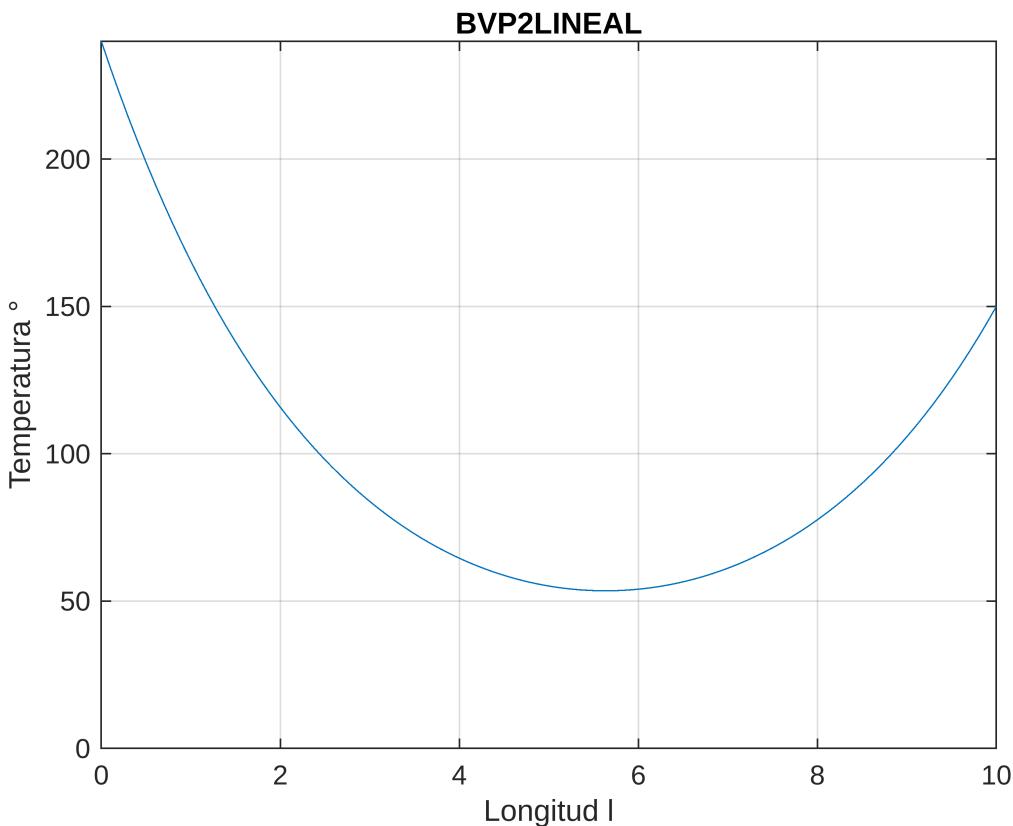
$$\text{Sea } j^* = 2, 3, 4, \dots, n-1$$

$$\Rightarrow b(j^*) = h^2 * g_{(j)}$$

y así sucesivamente, nuestro problema da valores en la frontera, y en un sistema tridiagonal de ecuaciones lineales

2) Resuelve el problema usando la función bvp2lineal y grafica la solución (2 puntos).

```
%function [t,y] = QuarPar_bvpplineal2(g0,g1,g2,a,b,ya,yb,n)
a = 0;b = 10; ya = 240;yb = 150;n=100;
g0 = @(t) 0+0*t;
g1 = @(t) .15+0*t;
g2 = @(t) 0+0*t;
[t, y] = bvpplineal2(g0,g1,g2,a,b,ya,yb,n);
plot(t,y); xlabel("Longitud l"); ylabel("Temperatura °"); title("BVP2LINEAL");
grid on; axis([a b 0 ya])
```



3) Resuelve el problema usando el método de shooting y grafica la solución (1 punto). Como valores iniciales tentativos de  $\frac{dy}{dx}$  puedes probar -60 y -120 antes de usar la función que encuentra raíces. También puedes tratar una pendiente inicial que estimes de la gráfica del inciso anterior.

```
M = [0,1;0.15,0];
fun = @(t,yz) M*yz;
h = 0.1;

target = 1;
g = @(z) shooting(fun, h, a, b, ya, z, target) - yb;

guess1 = -60;
PrimerShoot = g(guess1)
```

```

PrimerShoot = 1.8997e+03

guess2 = -120;
SegundoShoot = g(guess2)

SegundoShoot = -1.8234e+03

za = Biseccion(g,guess1, guess2)

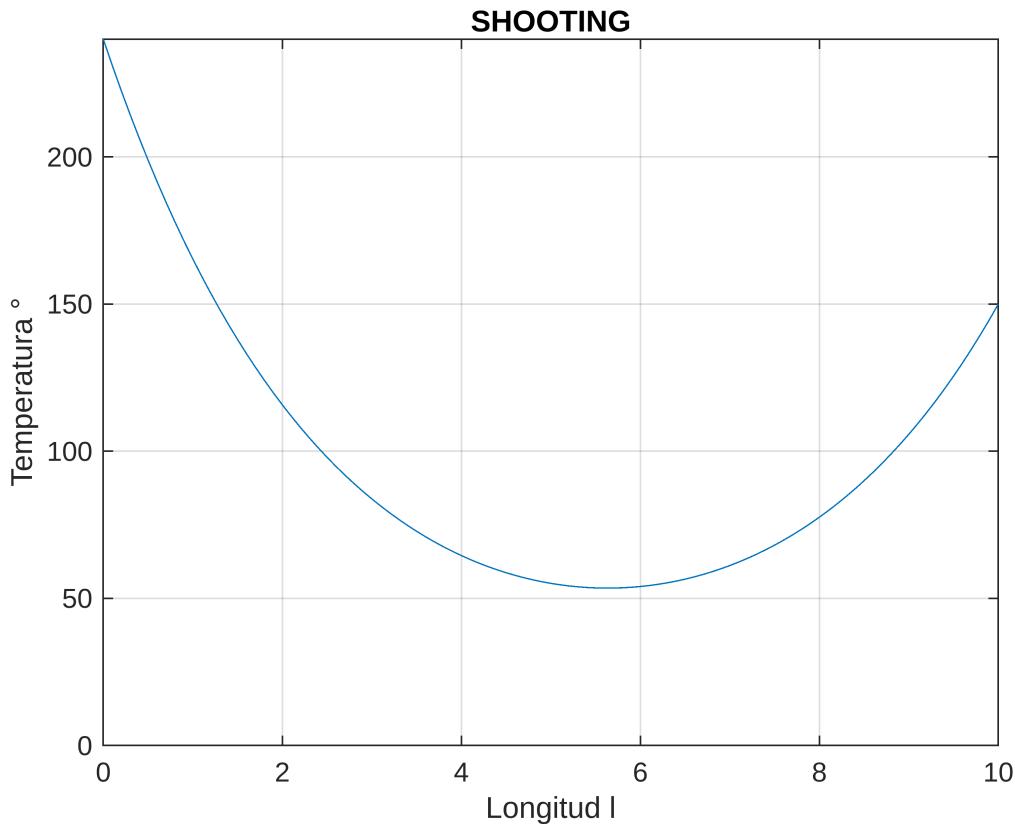
za = -90.6147

y0 = [ya;za];

[t,y] = RK4(fun, a, y0, h, b);

plot(t,y(1,:)); xlabel("Longitud l"); ylabel("Temperatura °");
title("SHOOTING"); grid on; axis([a b 0 ya])

```



## Métodos utilizados

Euler

```
function [t,y] = odeEuler(f, to, yo, h, tf)
```

```

t = t0:h:tf;
n = length(t);
m = length(y0);
y = zeros(m,n);
y(:,1) = y0;
for i=1:n-1
    y(:,i+1) = y(:,i) + f(t(i),y(:,i))*h;
end
end

```

## RK4

```

function [t,y] = RK4(f,t0,y0,h,tf)
t=t0:h:tf;
n=length(t);
m=length(y0);
y=zeros(m,n);
y(:,1)=y0;
for i=1:(n-1)
    k1=f(t(i),y(:,i));
    k2=f(t(i)+(1/2)*h,y(:,i)+(1/2)*k1*h);
    k3=f(t(i)+(1/2)*h,y(:,i)+(1/2)*k2*h);
    k4=f(t(i)+h,y(:,i)+k3*h);
    phi=(k1+2*k2+2*k3+k4)/6;
    y(:,i+1)=y(:,i)+phi*h;
end
end

```

## bvp2lineal

```

function [t,y] = bvp2lineal(g0,g1,g2,a,b,ya,yb,n)
t = linspace(a,b,n+2);
h = t(2)-t(1);
w0 = g0(t);
w1 = g1(t);
w2 = g2(t);
bv = zeros(n,1);
bv(1) = -(1 + w2(2)*h/2)*ya + w0(2)*h^2;
bv(n) = -(1 + w2(n+1)*h/2)*yb + w0(n+1)*h^2;
bv(2:n-1) = w0(3:n)*h^2;
diagonal = -(2+w1(2:n+1)*h^2);
subdiagonal = (1+w2(2:n+1)*h/2);
supdiagonal = (1-w2(2:n+1)*h/2);
res = QuarPar_Tridiag(subdiagonal,diagonal,supdiagonal,bv);
y = zeros(1,n+2);
y(1,1) = ya;
y(1,n+2) = yb;
y(1,2:n+1) = res;
end

```

## shooting

```
function Tl =shooting(f,h,x0,xf,t0,z0,target)
y0=[t0 ; z0];
[~,y]=RK4(f,x0,y0,h,xf);
Tl=y(target,end);
end
```

Función para encontrar raíces

```
function [xr,i]= Biseccion(f,xl,xu)
if sign(f(xl))*sign(f(xu)) >= 0
    error('f(xl)f(xu) < 0 no se satisface')
end
MAX = 55;
tolerancia = eps;
xr = (xl+xu)/2;
fx = f(xr);
i = 0;
while abs((xu-xl)/xu) > tolerancia && fx~=0 && i < MAX
    if sign(fx) ~= sign(f(xl))
        xu = xr;
    else
        xl = xr;
    end
    xr=(xl+xu)/2;
    fx=f(xr);
    i=i+1;
end
end
```