

**DESARROLLO**

Realizamos un cambio de variable.

La solución real es:

Realizamos la sustitución

**Raíces reales y distintas.**

Con condiciones iniciales.

Tenemos un sistema 2\*2

Reemplazamos en [1]

La solución real es:

**EL CÓDIGO DE MATLAB PROPUESTO.**

%% Rk4 vectorial

clc; clear all;close all; %Cerramos ventanas y limpiamos variables.

syms var\_u var\_y

format long

opcion=1;

ultimoDato=0;

while(opcion~=0)

disp('Digite: 1 si desea calcular la EDO por Método de runge kutta orden 4');

disp('Digite: 2 si desea calcular la EDO por Método de euler');

disp('Digite: 0 si desea salir del programa ');

opcion=input('');

if(ultimoDato~=1)

m=input('Digite la masa total del techo [m]: ');

E=input('Digite el módulo de elasticidad de las columna [E]: ');

I=input('Digite el momento de inercia de las columnas [I]: ');

hcolumna=input('Digite longitud de las columnas [hcol]: ');

h=input('Digite longitud del intervalo [h]: ');

a=input('Ingrese valor inicial de intervalo [a]: ');

b=input('Ingrese valor final de intervalo [b]: ');

u0=input('Digite condición inicial u(0): ');

u1=input('Digite condición inicial u''(0): ');

k=(E\*I)/h^2;

uprima=inline('var\_y',char(var\_u),char(var\_y));

yprima=inline(char((-k\*var\_u)/m),char(var\_u),char(var\_y));

n=(b-a)/h;

u=zeros(1,n);

y=zeros(1,n);

u(1)=u0; y(1)=u1;

ejeX=a:h:b;

end

switch opcion

case 1

%% método de runge kutta!

for i=2:n+1

k1(1,1)=h\*uprima(u(i-1),y(i-1));

k1(2,1)=h\*yprima(u(i-1),y(i-1));

k2(1,1)=h\*uprima(u(i-1)+k1(1,1)/2,y(i-1)+k1(2,1)/2);

k2(2,1)=h\*yprima(u(i-1)+k1(1,1)/2,y(i-1)+k1(2,1)/2);

k3(1,1)=h\*uprima(u(i-1)+k2(1,1)/2,y(i-1)+k2(2,1)/2);

k3(2,1)=h\*yprima(u(i-1)+k2(1,1)/2,y(i-1)+k2(2,1)/2);

k4(1,1)=h\*uprima(u(i-1)+k3(1,1),y(i-1)+k3(2,1));

k4(2,1)=h\*yprima(u(i-1)+k3(1,1),y(i-1)+k3(2,1));

u(i)=u(i-1)+(1/6)\*(k1(1,1)+2\*k2(1,1)+2\*k3(1,1)+k4(1,1));

y(i)=y(i-1)+(1/6)\*(k1(2,1)+2\*k2(2,1)+2\*k3(2,1)+k4(2,1));

end

%% graficamos

figure;

ax1 = subplot(2,1,1);

title('u vs y=u''');

plot(ax1,u,y,'--');xlabel('u');ylabel('y=u''');

grid on

ax2 = subplot(2,1,2);

title('u');

plot(ax2,ejeX,u,'--');xlabel('x');ylabel('u');

grid on

case 2

%% método de euler!

for i=2:n+1

k1(1,1)=h\*uprima(u(i-1),y(i-1));

k1(2,1)=h\*yprima(u(i-1),y(i-1));

u(i)=u(i-1)+h\*(k1(1,1));

y(i)=y(i-1)+h\*(k1(2,1));

end

figure;

ax1 = subplot(2,1,1);

title('u vs y=u''');

plot(ax1,u,y,'--');xlabel('u');ylabel('y=u''');

grid on

ax2 = subplot(2,1,2);

title('u');

plot(ax2,ejeX,u,'--');xlabel('x');ylabel('u');

grid on

end

ultimoDato=input('Digite 1: si desea usar los datos anteriores para otro método. ');

end

%

Pruebas con diferentes longitudes de intervalo (h) .

* Datos a ingresar:

Masa [m] =200kg

Módulo de elasticidad de las columna [E]: 0.01

Momento de inercia de las columnas [I]: 10

Longitud de las columnas [hcol]: 0.01

Longitud del intervalo [h]: 0.0001

Valor inicial de intervalo [a]: 1

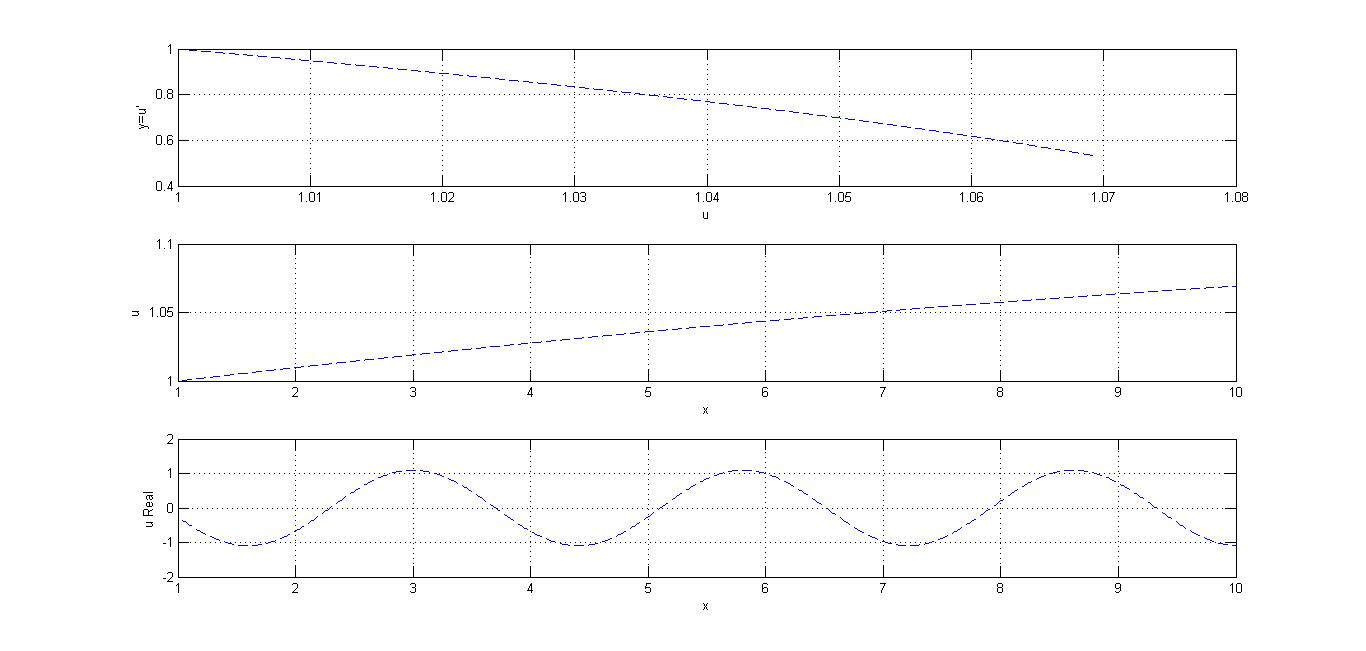
valor final de intervalo [b]: 10

condición inicial u(0): 1

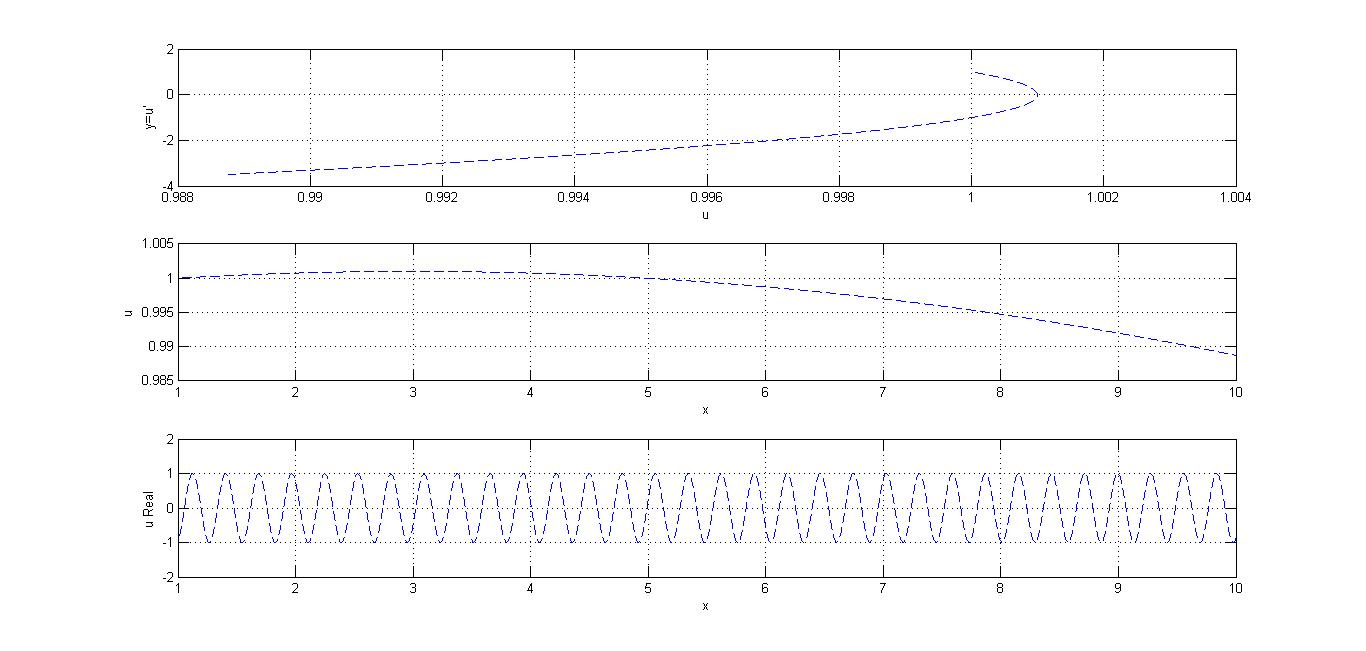
condición inicial u'(0): 1

Método de Euler

Con h= 0.01

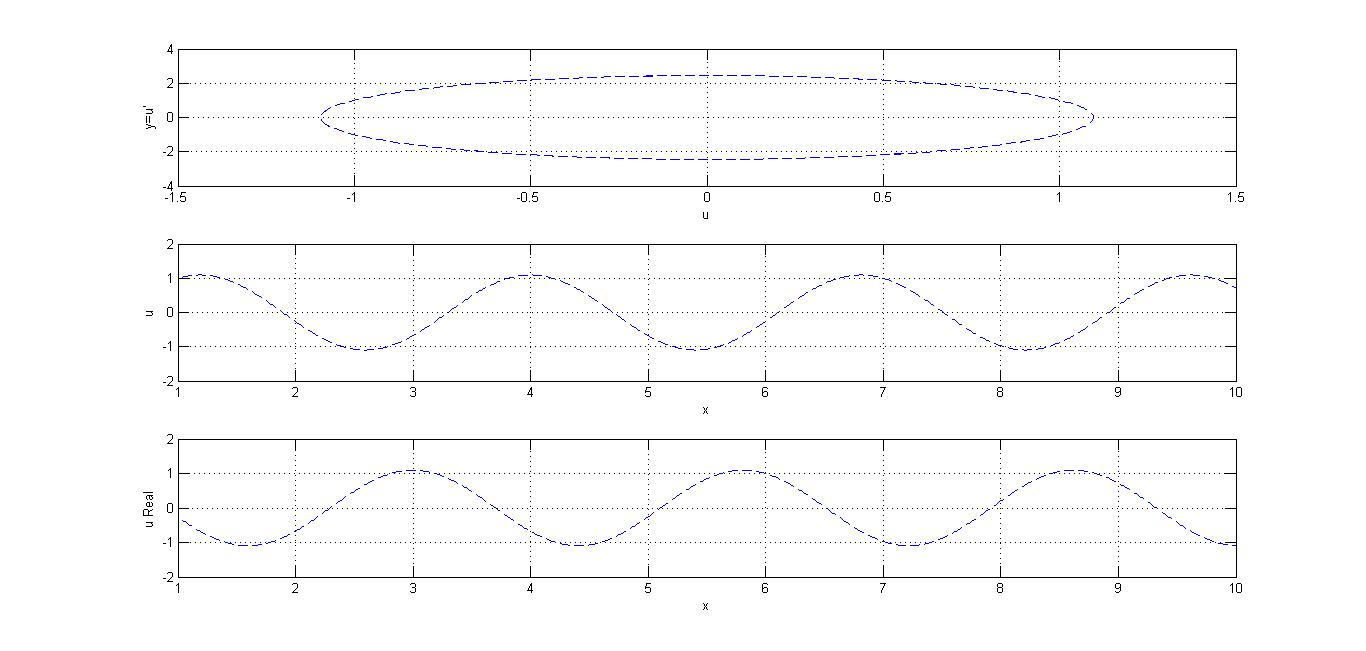


Con h= 0.001



Método de Runge Kutta

Con h= 0.01





**EL CÓDIGO DE MATLAB PROPUESTO.**

% metodo de solucion de ODE - Presa depredador

% metodo de euler

% Problema

% x1'(t)=k1\*x1(t)-k2\*x1(t)x2(t)

% x2'(t)=k3\*x1(t)x2(t)-k4\*x2(t)

% 0<=t<=200;

% y'= f(t,x,y)

% dado y(yto)=yo;

clc

clear all;

close all;

a=0;%valor del extremo inferior del intervalo.

b=200;%valor del extremo superior del intervalo.

N=1000000;% Valor arbitrario de subintervalos.

h=(b-a)/N;% tamaño de paso.

K1=3;K2=0.002;K3=0.0006;K4=0.5;%%Valores de K del problema.

fpresa='K1\*x-K2\*x\*y';%% x1(t)=x;

fdepredador='K3\*x\*y-K4\*y';%% x2(t)=y;

YPresa(1)=1000;

YDepredador(1)=500;

for i=1:N

x=YPresa(i);

y=YDepredador(i);

funcionpresa=eval(fpresa);%% se usa para eval para evaluar las funciones.

funcionDepredador=eval(fdepredador);

%% k1

k1Presa=h\*funcionpresa;

k1Depredador=h\*funcionDepredador;

%% Solucion

YPresa(i+1)=YPresa(i)+k1Presa;

YDepredador(i+1)=YDepredador(i)+k1Depredador;

end

t=a:h:b;%% intervalo de tiempo

figure(1)

subplot(2,1,1);

hold on

plot(t,YPresa,'r')

plot(t,YDepredador,'b')

title('Población de presa en rojo y Depredador en azul contra el tiempo');

subplot(2,1,2);

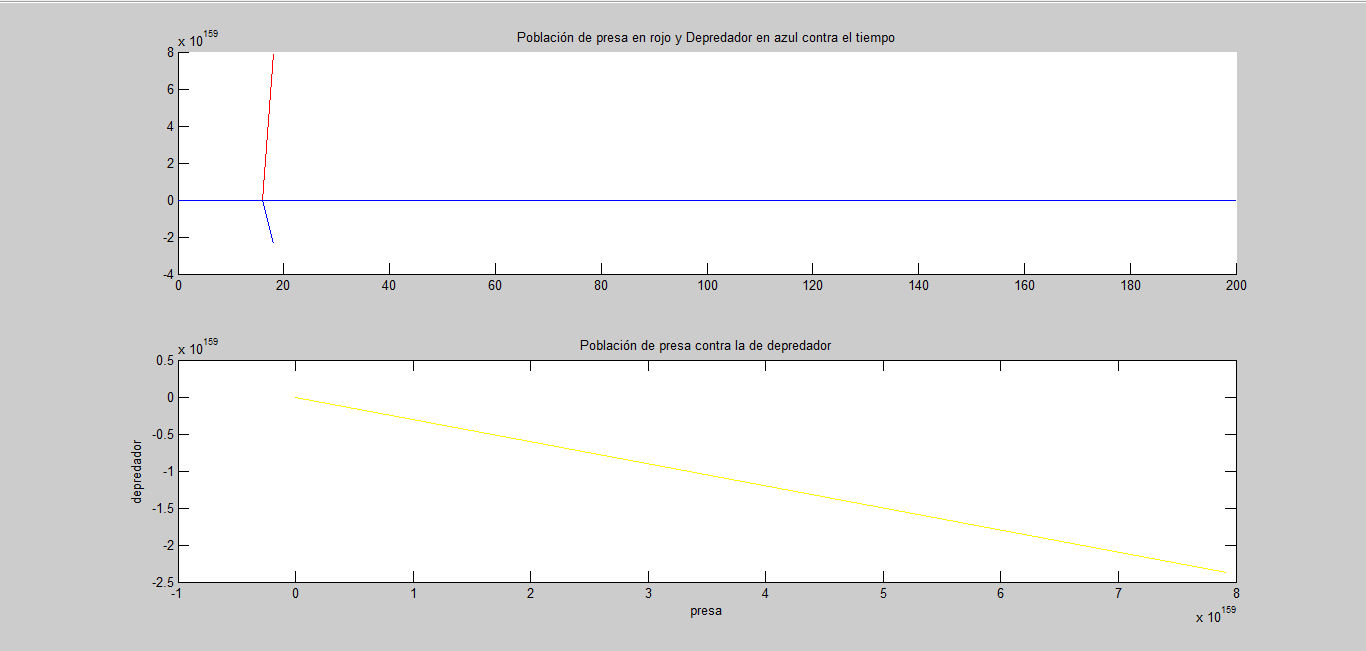
plot(YPresa,YDepredador,'y')

title('Población de presa contra la de depredador');

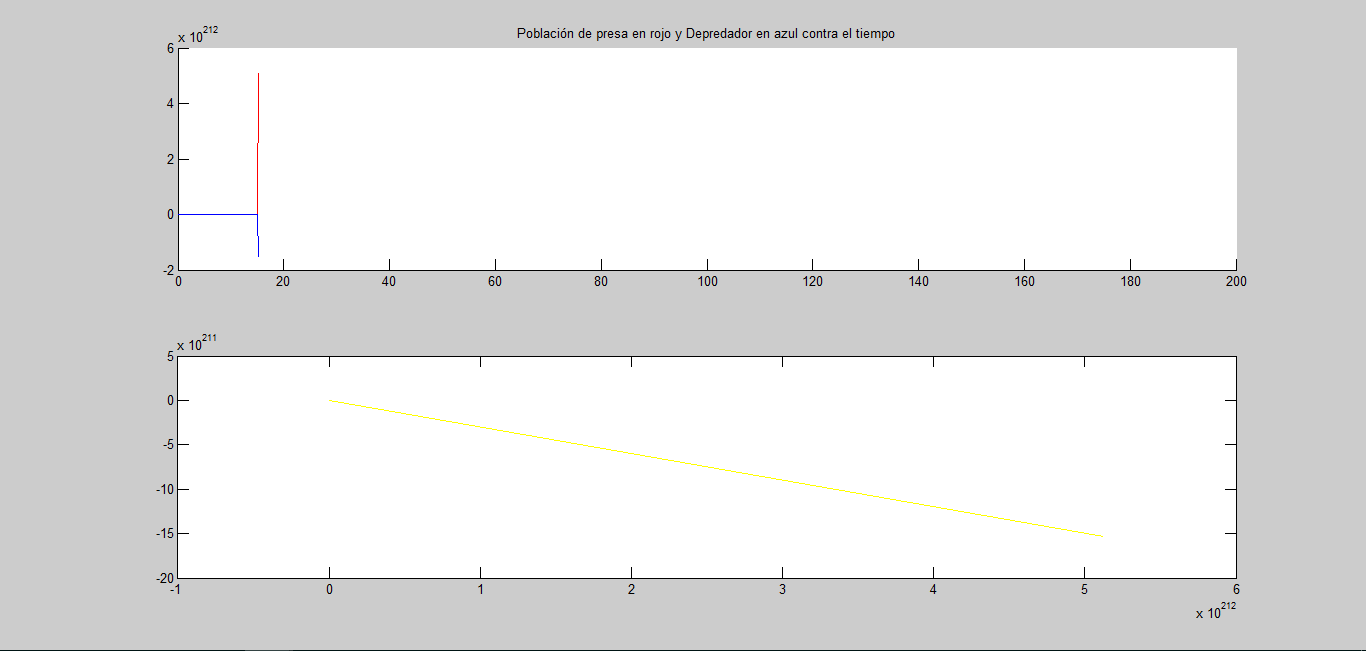
xlabel('presa');

ylabel('depredador');

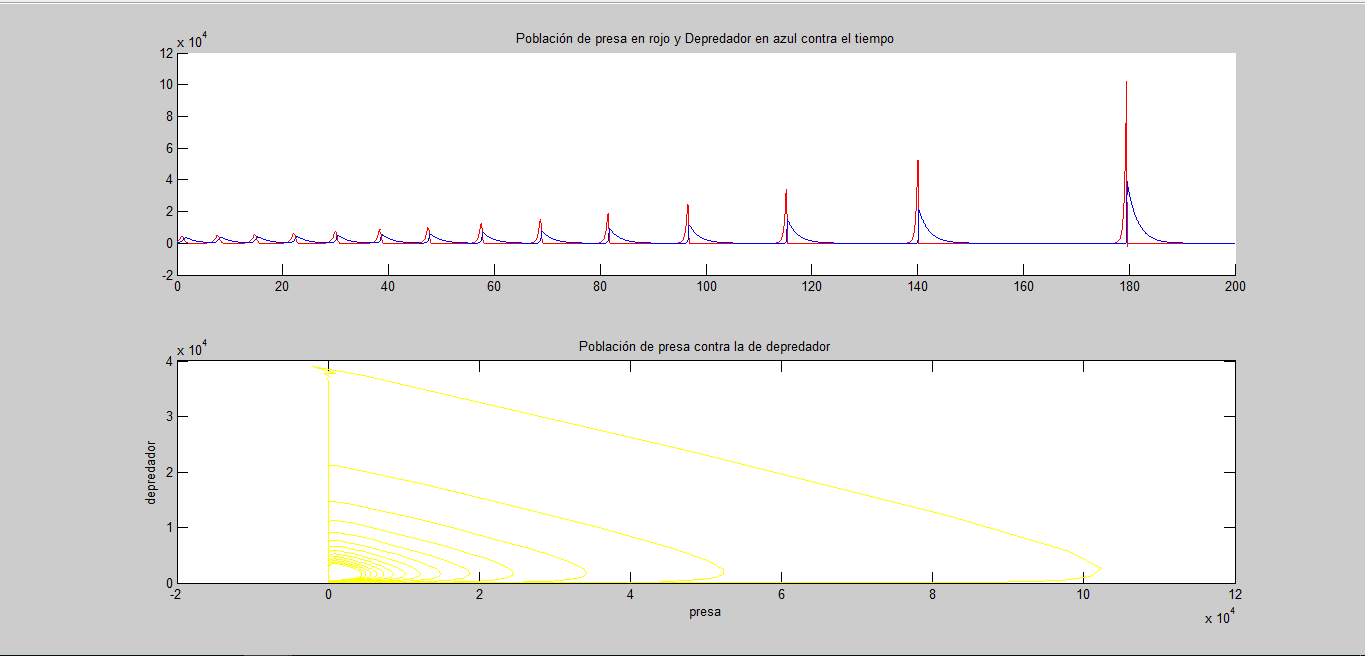
**(ensayos con diferentes *h*)**



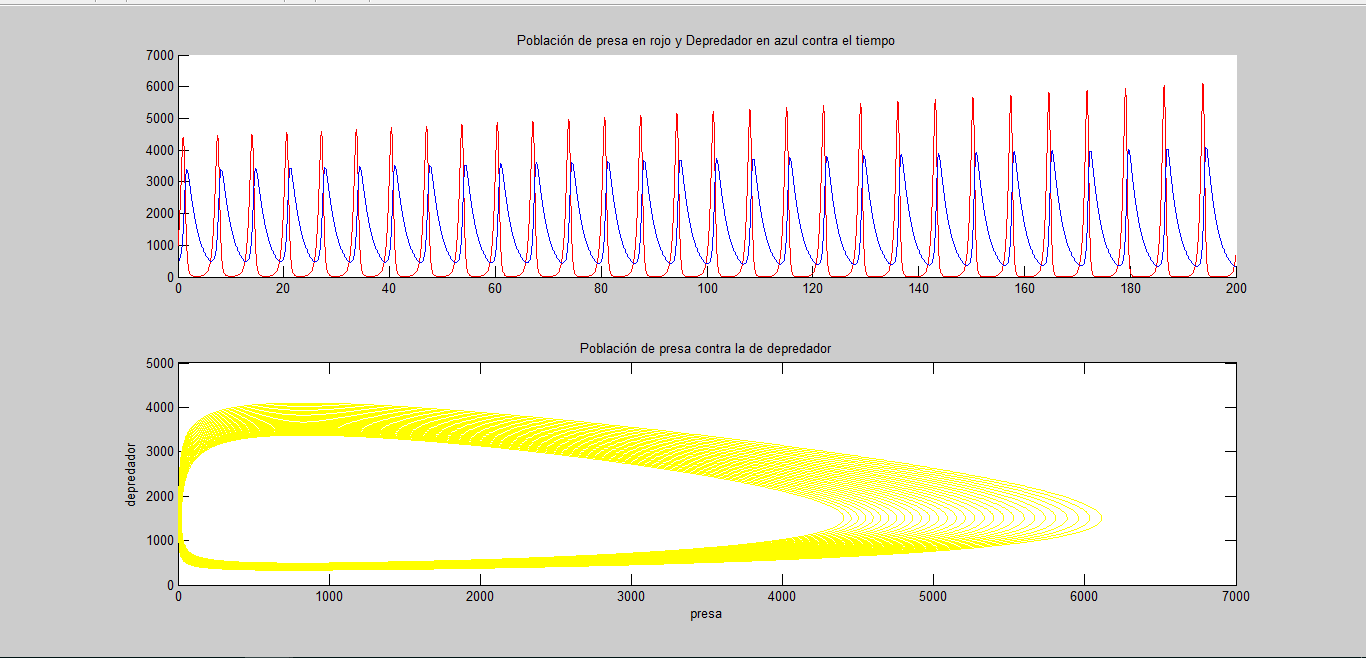
Puntos= 100 h=2;



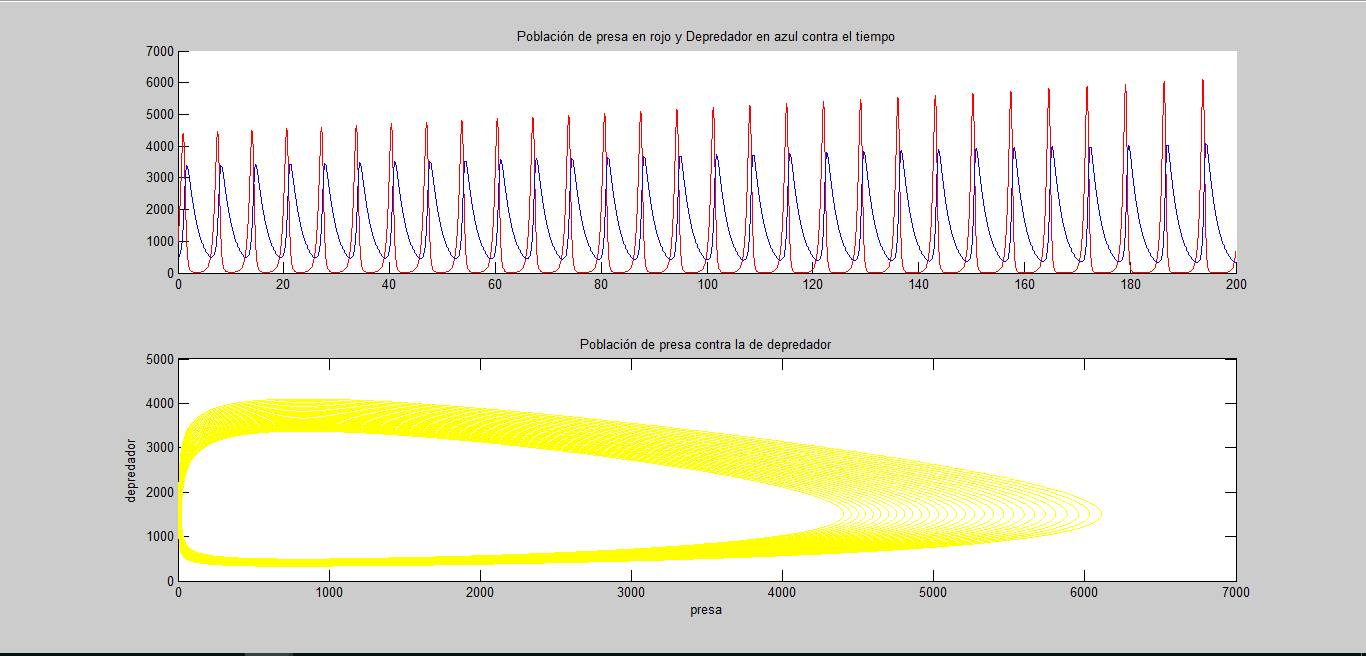
Puntos=1000 h=0.2;



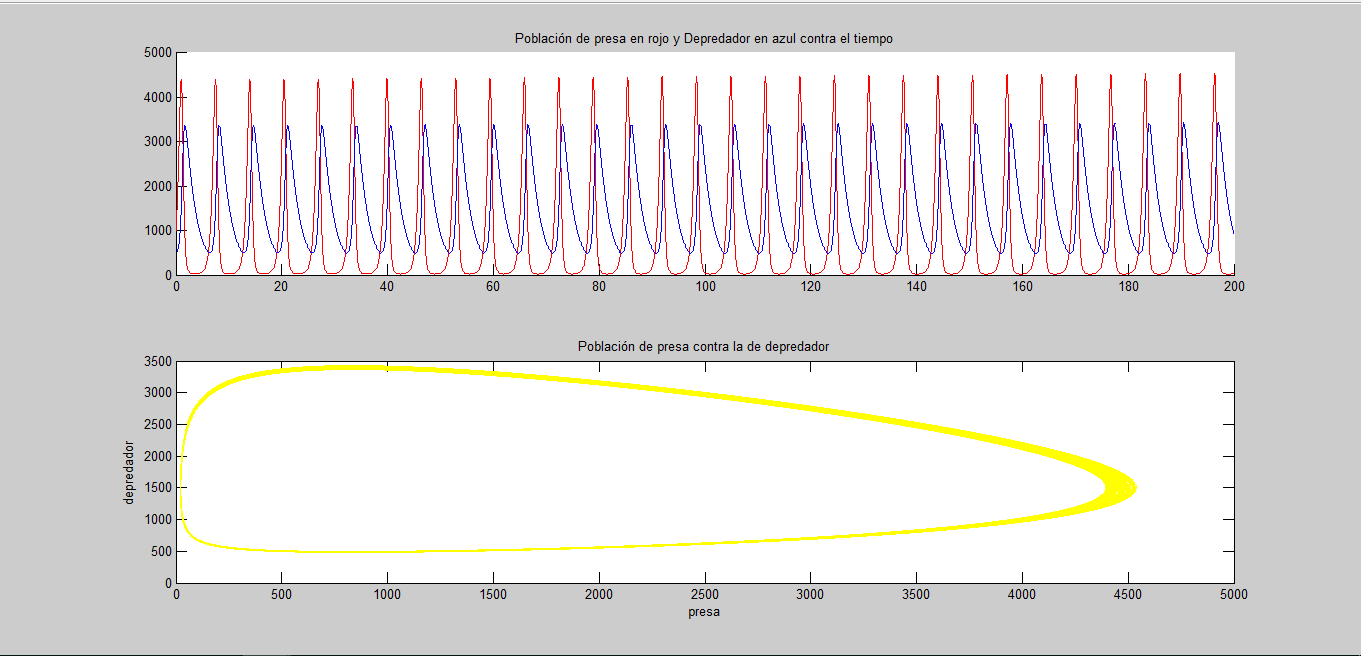
Puntos=10000 h=0.02;



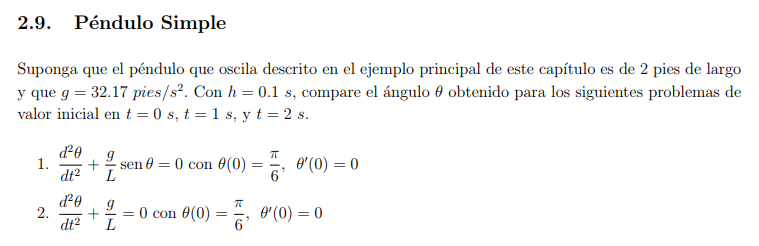
Puntos=1000000 h=0.002;



Puntos=1000000 h= 0.002;



Puntos=10000000 h= 0.0002;





Función 1

Función 2

**EL CÓDIGO DE MATLAB PROPUESTO.**

clc

clear all;

a=0;%valor del extremo inferior del intervalo.

b=2;%valor del extremo superior del intervalo.

h=0.1;%valor de h 0.1s

N=(b-a)/h;% tamaño de paso.

g=32.17;L=2;%%Valores de g & L del problema.

f1='v';%% x1(t)=x;

f2='-(g/L)\*sin(thetha)';

thethaf1(1)=pi/6;

thethaf2(1)=0;

for i=1:N

v=thethaf1(i);

thetha=thethaf2(i);

funcion1=eval(f1);

funcion2=eval(f2);

%% k1

k1f1=funcion1;

k1f2=funcion2;

%% k2

v=thethaf1(i)+(1/2)\*h\*k1f1;

thetha=thethaf2(i)+(1/2)\*h\*k1f2;

funcion1=eval(f1);

funcion2=eval(f2);

k2f1=funcion1;

k2f2=funcion2;

%% k3

v=thethaf1(i)+(1/2)\*h\*k2f1;

thetha=thethaf2(i)+(1/2)\*h\*k2f2;

funcion1=eval(f1);

funcion2=eval(f2);

k3f1=funcion1;

k3f2=funcion2;

%% k4

v=thethaf1(i)+h\*k3f1;

thetha=thethaf2(i)+h\*k3f2;

funcion1=eval(f1);

funcion2=eval(f2);

k4f1=funcion1;

k4f2=funcion2;

%% Solucion

thethaf1(i+1)=thethaf1(i)+(1/6)\*(k1f1+2\*(k2f1+k3f1)+k4f1)\*h;

thethaf2(i+1)=thethaf2(i)+(1/6)\*(k1f2+2\*(k2f2+k3f2)+k4f2)\*h;

end

t=a:h:b;%% intervalo de tiempo

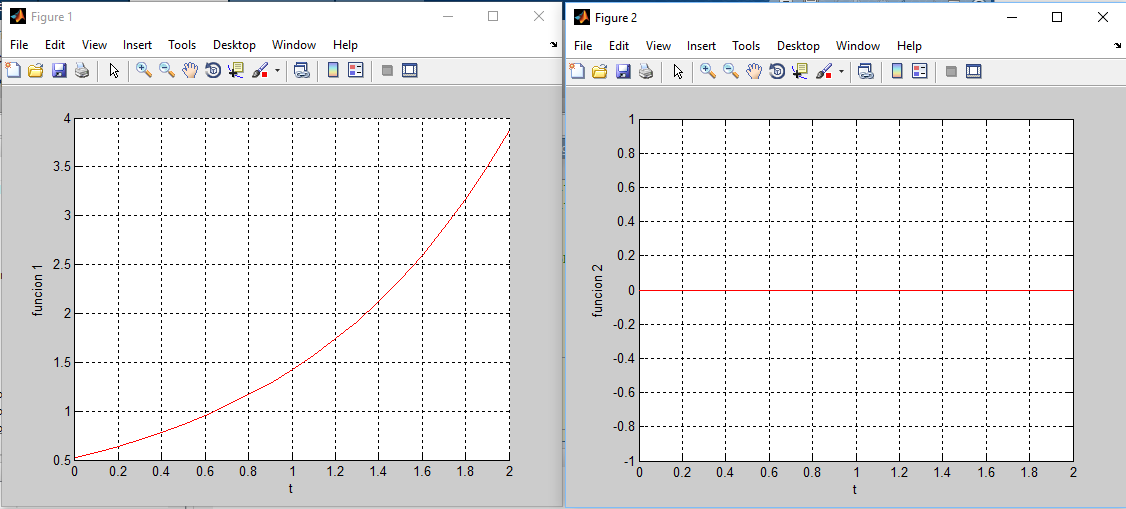
figure(1)

hold on

plot(t,thethaf1,'r')

xlabel('t');

ylabel('thetha');



Función 1

Función 2

clc

clear all;

a=0;%valor del extremo inferior del intervalo.

b=2;%valor del extremo superior del intervalo.

h=0.1;%valor de h 0.1s

N=(b-a)/h;% tamaño de paso.

g=32.17;L=2;%%Valores de g & L del problema.

f1='v';%% x1(t)=x;

f2='-(g/L)';

thethaf1(1)=pi/6;

thethaf2(1)=0;

for i=1:N

v=thethaf1(i);

thetha=thethaf2(i);

funcion1=eval(f1);

funcion2=eval(f2);

%% k1

k1f1=funcion1;

k1f2=funcion2;

%% k2

v=thethaf1(i)+(1/2)\*h\*k1f1;

thetha=thethaf2(i)+(1/2)\*h\*k1f2;

funcion1=eval(f1);

funcion2=eval(f2);

k2f1=funcion1;

k2f2=funcion2;

%% k3

v=thethaf1(i)+(1/2)\*h\*k2f1;

thetha=thethaf2(i)+(1/2)\*h\*k2f2;

funcion1=eval(f1);

funcion2=eval(f2);

k3f1=funcion1;

k3f2=funcion2;

%% k4

v=thethaf1(i)+h\*k3f1;

thetha=thethaf2(i)+h\*k3f2;

funcion1=eval(f1);

funcion2=eval(f2);

k4f1=funcion1;

k4f2=funcion2;

%% Solucion

thethaf1(i+1)=thethaf1(i)+(1/6)\*(k1f1+2\*(k2f1+k3f1)+k4f1)\*h;

thethaf2(i+1)=thethaf2(i)+(1/6)\*(k1f2+2\*(k2f2+k3f2)+k4f2)\*h;

end

t=a:h:b;%% intervalo de tiempo

figure(1)

hold on

plot(t,thethaf1,'r')

xlabel('t');

ylabel('funcion 1');

grid on

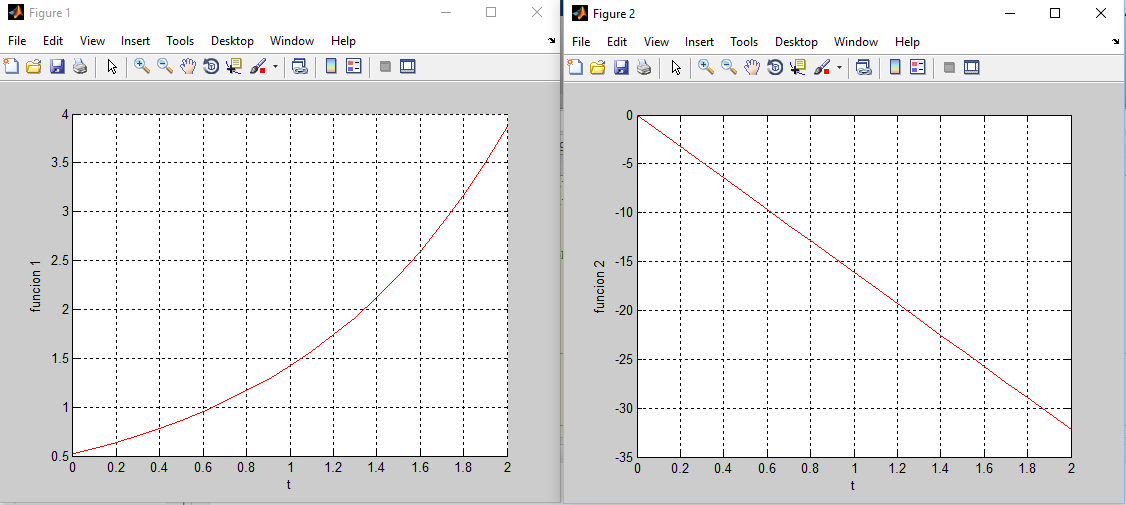
figure(2)

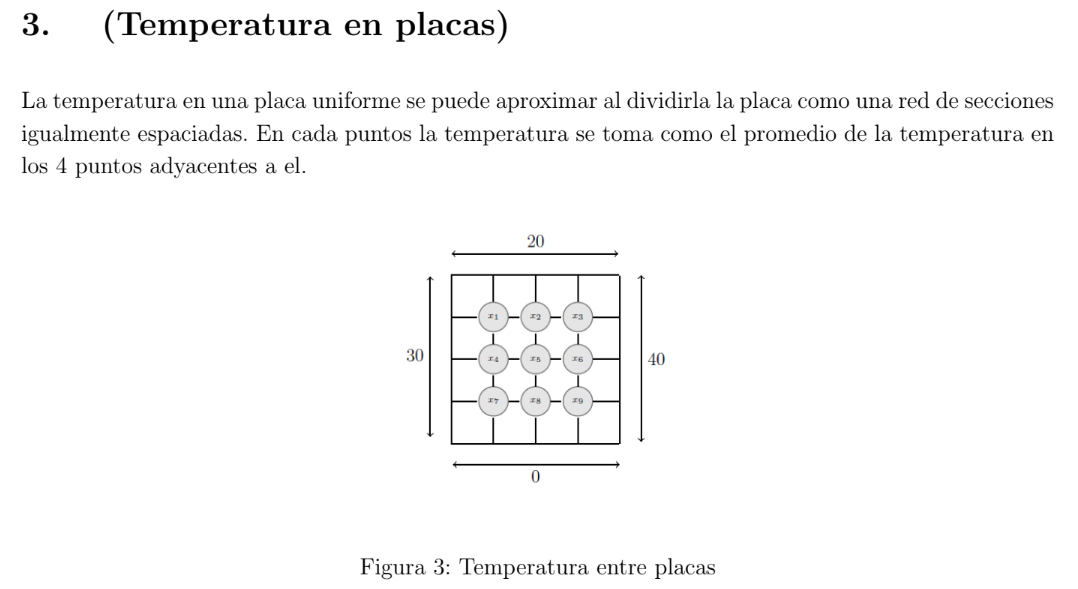
plot(t,thethaf2,'r')

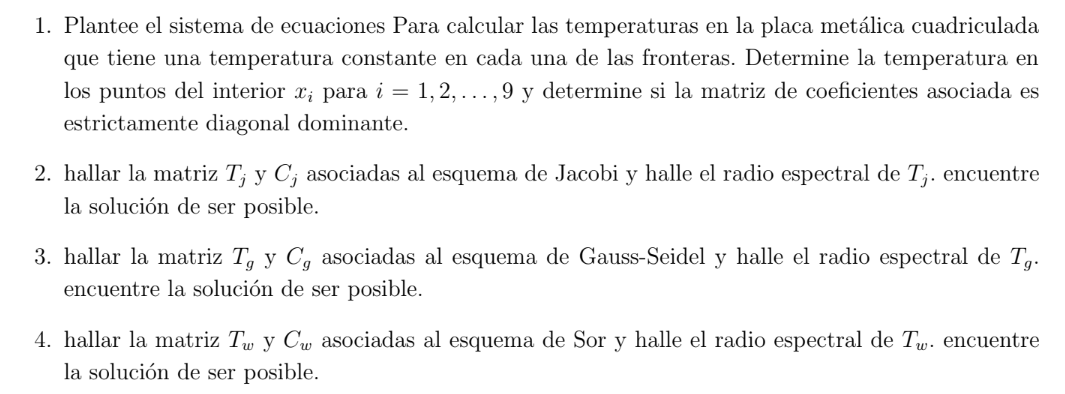
xlabel('t');

ylabel('funcion 2');

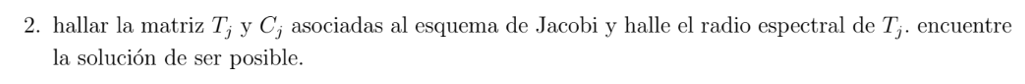
grid on







SOLUCION



La matriz de coeficientes es

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4 | -1 | 0 | -1 | 0 | 0 | 0 | 0 | 0 |
| -1 | 4 | -1 | 0 | -1 | 0 | 0 | 0 | 0 |
| 0 | -1 | 4 | 0 | 0 | -1 | 0 | 0 | 0 |
| -1 | 0 | 0 | 4 | -1 | 0 | -1 | 0 | 0 |
| 0 | -1 | 0 | -1 | 4 | -1 | 0 | -1 | 0 |
| 0 | 0 | -1 | 0 | -1 | 4 | 0 | 0 | -1 |
| 0 | 0 | 0 | -1 | 0 | 0 | 4 | -1 | 0 |
| 0 | 0 | 0 | 0 | -1 | 0 | -1 | 4 | -1 |
| 0 | 0 | 0 | 0 | 0 | -1 | 0 | -1 | 4 |

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

La Matriz

=

|  |
| --- |
| 30 |
| 0 |
| 40 |
| 30 |
| 0 |
| 40 |
| 50 |
| 20 |
| 60 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4 | -1 | 0 | -1 | 0 | 0 | 0 | 0 | 0 |  |
| -1 | 4 | -1 | 0 | -1 | 0 | 0 | 0 | 0 |  |
| 0 | -1 | 4 | 0 | 0 | -1 | 0 | 0 | 0 |  |
| -1 | 0 | 0 | 4 | -1 | 0 | -1 | 0 | 0 |  |
| 0 | -1 | 0 | -1 | 4 | -1 | 0 | -1 | 0 |  |
| 0 | 0 | -1 | 0 | -1 | 4 | 0 | 0 | -1 |  |
| 0 | 0 | 0 | -1 | 0 | 0 | 4 | -1 | 0 |  |
| 0 | 0 | 0 | 0 | -1 | 0 | -1 | 4 | -1 |  |
| 0 | 0 | 0 | 0 | 0 | -1 | 0 | -1 | 4 |  |

|  |
| --- |
| 30 |
| 0 |
| 40 |
| 30 |
| 0 |
| 40 |
| 50 |
| 20 |
| 60 |

=

La matriz es estrictamente dominante ya que

...

Y asi sucesivamente para todos los elementos de la diagonal

La matriz Tj es

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0.00 | 0.25 | 0.00 | 0.25 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| -0.25 | 0.00 | 0.25 | 0.00 | 0.25 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.25 | 0.00 | 0.00 | 0.00 | 0.25 | 0.00 | 0.00 | 0.00 |
| -0.25 | 0.00 | 0.00 | 0.00 | 0.25 | 0.00 | 0.25 | 0.00 | 0.00 |
| 0.00 | 0.25 | 0.00 | 0.25 | 0.00 | 0.25 | 0.00 | 0.25 | 0.00 |
| 0.00 | 0.00 | 0.25 | 0.00 | 0.25 | 0.00 | 0.00 | 0.00 | 0.25 |
| 0.00 | 0.00 | 0.00 | 0.25 | 0.00 | 0.00 | 0.00 | 0.25 | 0.00 |
| 0.00 | 0.00 | 0.00 | 0.00 | 0.25 | 0.00 | 0.25 | 0.00 | 0.25 |
| 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.25 | 0.00 | 0.25 | 0.00 |

|  |
| --- |
| 15 |
| 0 |
| 12.5 |
| -15 |
| 0 |
| 12.5 |
| 45 |
| 30 |
| 42.5 |

La matriz Cj es:

|  |
| --- |
| 17.14284 |
| 15.08926 |
| 20.71428 |
| 23.48212 |
| 22.49998 |
| 27.76785 |
| 24.28570 |
| 23.66070 |
| 27.85714 |

Temperatura de la placa para cada

**EL CÓDIGO DE MATLAB PROPUESTO.**

%% Metodo de jacobi para la temperatura en un placa

A=[[4,-1,0,-1,0,0,0,0,0],

[-1,4,-1,0,-1,0,0,0,0],

[0,-1,4,0,0,-1,0,0,0],

[-1,0,0,4,-1,0,-1,0,0],

[0,-1,0,-1,4,-1,0,-1,0],

[0,0,-1,0,-1,4,0,0,-1],

[0,0,0,-1,0,0,4,-1,0],

[0,0,0,0,-1,0,-1,4,-1],

[0,0,0,0,0,-1,0,-1,4]];

b=[30,

0,

40,

30,

0,

40,

50,

20,

60]'

x=[0 0 0 0 0 0 0 0 0]'

n=size(x,1);

normVal=Inf;

%%

% \* \_\*Tolerence for method\*\_

tol=1e-5; itr=0;

%% Algorithm: Jacobi Method

%%

while normVal>tol

xold=x;

for i=1:n

sigma=0;

for j=1:n

if j~=i

sigma=sigma+A(i,j)\*x(j);

end

end

x(i)=(1/A(i,i))\*(b(i)-sigma);

end

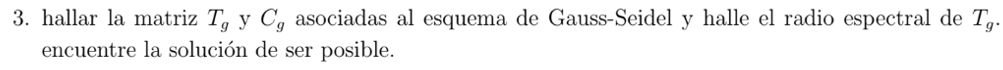
itr=itr+1;

normVal=abs(xold-x);

end

%%

fprintf('La solucion del sistema para la placa de temperatura es : \n%f\n%f\n%f\n%f\n%f\n%f\n%f\n%f\n%f\n en %d iteraciones',x,itr);



La solución del sistema con gauss seidel es

|  |
| --- |
| 17.142835 |
| 15.089263 |
| 20.714275 |
| 23.482121 |
| 22.499978 |
| 27.767846 |
| 24.285703 |
| 23.660703 |
| 27.857137 |

Temperatura de la placa para cada

**EL CÓDIGO DE MATLAB PROPUESTO.**

%% Metodo de gauss seidel para la temperatura en un placa

A=[[4,-1,0,-1,0,0,0,0,0],

[-1,4,-1,0,-1,0,0,0,0],

[0,-1,4,0,0,-1,0,0,0],

[-1,0,0,4,-1,0,-1,0,0],

[0,-1,0,-1,4,-1,0,-1,0],

[0,0,-1,0,-1,4,0,0,-1],

[0,0,0,-1,0,0,4,-1,0],

[0,0,0,0,-1,0,-1,4,-1],

[0,0,0,0,0,-1,0,-1,4]];

b=[30,

0,

40,

30,

0,

40,

50,

20,

60]'

x=[0 0 0 0 0 0 0 0 0]'

n=size(x,1);

normVal=Inf;

tol=1e-5; itr=0;

%%

while normVal>tol

xold=x;

for i=1:n

sigma=0;

for j=1:n

if j~=i

sigma=sigma+A(i,j)\*x(j);

end

end

x(i)=(1/A(i,i))\*(b(i)-sigma);

end

itr=itr+1;

normVal=abs(xold-x);

end

%%

fprintf('La solucion del sistema para la placa de temperatura es : \n%f\n%f\n%f\n%f\n%f\n%f\n%f\n%f\n%f\n en %d iteraciones',x,itr);