Objetivo:

Aproximar numéricamente la solución de una EDO.

CONTENIDO:

- ✓ Ecuaciones diferenciales ordinarias.
- ✓ Ecuaciones diferenciales algebraicas.
- ✓ Método de Euler.
- ✓ Método de Taylor.
- ✓ Método de runge Kutta.
- ✓ Ecuaciones diferenciales de segundo orden.
- ✓ Ecuaciones diferenciales de tercer orden.
- ✓ Ecuaciones diferenciales de cuarto orden.

Capitulo 3

CAPITULO III. ECUACIONES DIFERENCIALES LINEALES

En este capítulo se presentan diferentes métodos numéricos que aproximan soluciones de ecuaciones diferenciales con condiciones iniciales.

3.1 Ecuaciones diferenciales ordinarias

Definición 4. Una ecuación diferencial es la que contiene derivadas o diferenciales de una función incognita. (Espinoza, 2002).

3.2. El comando lsode de Octave

Octave tiene incorporado el comando lsode para resolver problemas de valor inicial. Los parámetros de entrada que necesita la función lsode son:

- Una función anónima.
- Un vector unidimensional con los valores iniciales u_0.
- Un vector que contenga las iteraciones t_n para calcular la aproximación.

La función lsode tiene como salida una matriz por defecto, las columnas contienen los valores y_n, que son los resultados aproximados del vector solución u sobre cada uno de los nodos t_n especificados en el tercer argumento.

[x, i_inicial, msg] = lsode(Nombre, x_0, t)

La ecuación diferencial a resolver es:

$$\frac{dx}{dt} = f(x,t)$$

con

$$x(t_0) = x_0$$

El jacobiano de $f(x)$ tiene la forma:

jac = j (x, t)

donde

jac es la matriz de derivadas parciales.

$$J = \frac{\partial f_i}{\partial x_j} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_N} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_N}{\partial x_1} & \frac{\partial f_N}{\partial x_2} & \cdots & \frac{\partial f_N}{\partial x_N} \end{bmatrix}$$

Ejemplo 3.1. Aproximar numéricamente el sistema Van Der Pol y visualizar su gráfica.

========== Incio del script ==========

```
funcion = @(y, t) [y(2); (1-y(1)^2)*y(2)-y(1)];
t = linspace (0, 30, 150); y = lsode (funcion, [3; 1], t);
fprintf(' y1       y2\n'); disp([y(:, 1) y(:, 2)]);
plot(y(:, 1), y(:, 2), 'k', 'linewidth', 1.5)
xlabel('Eje y1'); ylabel('Eje y2'); grid;
```

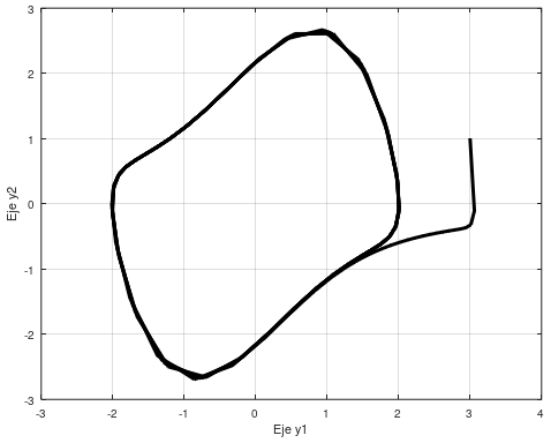
========== Fin del script ==========

Solución numérica del sistema de ecuaciones de Van Der Pol.

y_1	y_2
3.0000	1.0000
3.0606	-0.1092
3.0118	-0.3203
2.9415	-0.3677
2.8654	-0.3868
2.7861	-0.4014
2.7038	-0.4163
2.6183	-0.4326
2.5294	-0.4508
2.4367	-0.4714
2.3394	-0.4950
2.2371	-0.5222
2.1288	-0.5541
2.0136	-0.5919
1.8899	-0.6376

Visualización gráfica del sistema de ecuaciones de Van Der Pol. Ver figura 3.1.

FIGURA 3. 1



Grafica del sistema de Van Der Pol con la función lsode.

Ejemplo 3.2. Aproximar numéricamente el siguiente sistema de ecuaciones y visualizar su gráfica.

$$\begin{cases} x' = 77.27(y - x * y + x - 0.005 * x^2) \\ y' = (z - x * y - y)/77.27 \\ z' = 0.161 * (x - z) \end{cases}$$

========== Inicio de script ==========

function X=f(x, t)

```
 X=zeros(3,1);
 X(1)=77.27*(x(2)-x(1)*x(2)+x(1)-0.005*x(1)^2);
 X(2)=(x(3)-x(1)*x(2)-x(2))/77.27;
 X(3)=0.161*(x(1)-x(3));
endfunction
```

========== Fin de script ==========

En otro script se obtiene la solución del sistema aproximado de forma numérica.

========== Inicio de script ==========

```
x0 = [6;0.1; 6];
t = linspace (0, 300, 1500);
y = lsode('f', x0, t);
disp(y(1:15,:))
plot3(y(:, 1), y(:, 2), y(:, 3), 'k','linewid',1.5)
grid
```

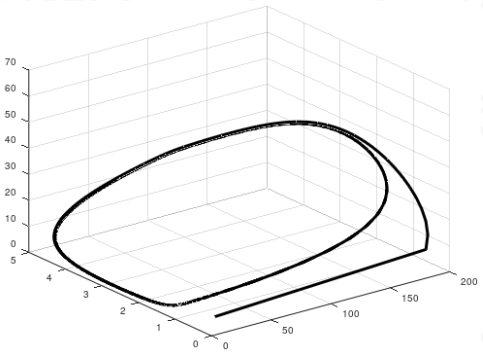
========== Fin de script ==========

Solución numérica del sistema de ecuaciones tridimensional.

x	y	z
6.0000e+00	1.0000e-01	6.0000e+00
1.8263e+02	8.6185e-02	1.0172e+01
1.8400e+02	8.0498e-02	1.5671e+01
1.8263e+02	8.8147e-02	2.0992e+01
1.7957e+02	1.0406e-01	2.6072e+01
1.7546e+02	1.2505e-01	3.0876e+01
1.7071e+02	1.4920e-01	3.5386e+01
1.6555e+02	1.7536e-01	3.9596e+01
1.6013e+02	2.0285e-01	4.3504e+01
1.5451e+02	2.3131e-01	4.7112e+01
1.4875e+02	2.6052e-01	5.0426e+01
1.4286e+02	2.9041e-01	5.3451e+01
1.3685e+02	3.2099e-01	5.6190e+01
1.3069e+02	3.5231e-01	5.8650e+01
1.2438e+02	3.8451e-01	6.0834e+01

Los resultados obtenidos se grafican en octave. (ver figura 3.2)

FIGURA 3. 2



Gráfica de un sistema tridimensional usando la función lsode.

Los comandos que aproximan soluciones actualmente en Octave son:

- Método de Runge-Kutta.
  La función ode45, resuelve sistemas de ecuaciones diferenciales ordinarias no rígidas, esta función usa el método Dormand-Prince de paso variable y orden alto, también ofrece un rendimiento mejor con tolerancias más pequeñas.

  La función ode23, resuelve un sistema de ecuaciones diferenciales ordinarias no rígidas, esta función aplica el método Bogacki-Shampine de tercer orden y acopla el tamaño de paso local para satisfacer una tolerancia especificada por el usuario.

  La función ode23s, al igual que las funciones ode45 y ode23 resuelve sistemas de ecuaciones diferenciales ordinarias rígidas usando el método modificado de Rosenbrock de segundo orden.

- Métodos lineales de varios pasos.
  La función ode15s, resuelve un conjunto de ecuaciones diferenciales ordinarias rígidas utilizando un paso variable.

  La función ode15i, resuelve un conjunto de ecuaciones diferenciales ordinarias totalmente implícitas usando el mismo método de paso variable y orden variable que ode15s.

[t, y] = ode45 (Función, Rango, Condición Inicial)

Función. Es una cadena que almacena el nombre de la función que se define en la EDO: $y' = f(t, y)$. La función debe tener dos entradas, el primero es el tiempo t y el segundo es un vector columna de variable y.

Rango. Son los límites de tiempo durante el cual se evaluará la EDO. Típicamente es un vector de dos elementos que tiene un inicio y un final ([t_init, t_final]).

Condición Inicial. Es el valor inicial del PVI.

Ejemplo 3.3. Aproximar numéricamente el sistema de ecuaciones de Van Der Pol con octave.

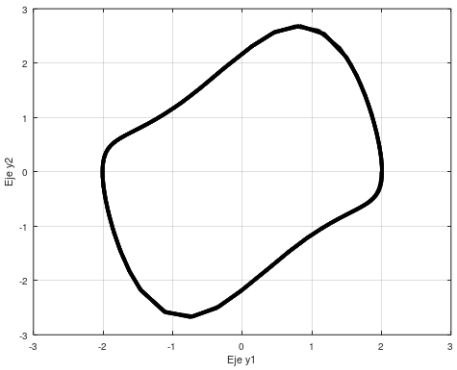
============== inicio de script ============

```
F= @(t, y) [y(2); (1 - y(1)^2) * y(2) - y(1)];
[t, y] = ode45 (F, [0, 20], [2, 0]);
disp([y(:, 1),y(:, 2)]);
plot(y(:, 1), y(:, 2), 'k', 'linewidth', 2)
xlabel('Eje y1'); ylabel('Eje y2'); grid;
```

============== fin de script ===============

Se visualiza la gráfica del sistema de Van Der Pol. (Ver figura 3.3).

FIGURA 3. 3



Grafica del sistema de ecuaciones de Van Der Pol usando la función ode45.

A continuación, se muestra con un ejemplo cómo se trabaja con la función ode23.

[t, y] = ode23 (Función, Rango, Condición Inicial)

Ejemplo 3.4. Aproximar numéricamente el sistema de ecuaciones de Van Der Pol.

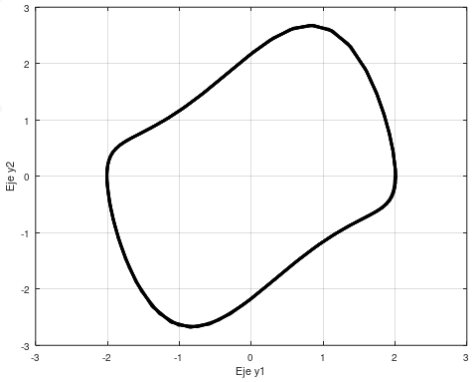
============ Inicio de script ==============

```
Fun = @(t, y) [y(2); (1 - y(1)^2) * y(2) - y(1)];
[t, y] = ode23 (Fun, [0, 20], [2, 0]);
disp([y(:,1),y(:,2)]);
plot(y(:,1),y(:,2),'k','linewidth',1.5)
xlabel('Eje y1');ylabel('Eje y2'); grid;
```

============== Fin de script ==============

Se visualiza la gráfica del sistema de ecuaciones de Van Der Pol en la figura 3.4.

FIGURA 3. 4



Grafica del sistema de ecuaciones de Van Der Pol usando la función ode23.

El ejemplo 3.5, muestra cómo se trabaja con la función ode23s.

Ejemplo 3.5. Aproximar numéricamente el sistema de ecuaciones rígida de Van Der Pol.

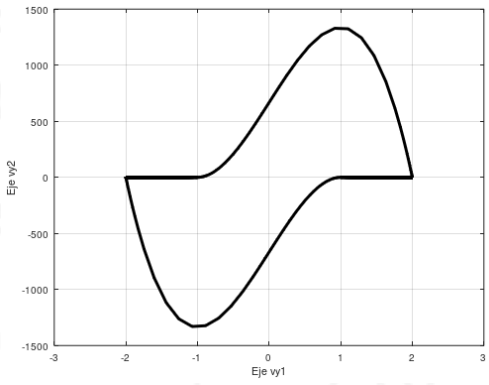
========== Inicio de script ==========

```
F = @(t, y) [y(2); 1000*(1-y(1)^2)*y(2)-y(1)];
Opt = odeset ('Mass', [1 0; 0 1], 'MaxStep', 1e-1);
[vt, vy]=ode23s(F, [0 2000], [2 0], Opt);
disp([vy(:, 1), vy(:, 2)]);
plot(vy(:, 1), vy(:, 2), 'k', 'linewidth', 1.5)
xlabel('Eje vy1'); ylabel('Eje vy2'); grid;
```

========== Fin de script ==========

La gráfica del sistema de ecuaciones de Van Der Pol se visualiza en la figura 3.5.

FIGURA 3. 5



Grafica del sistema de ecuaciones de Van Der Pol usando la función ode23s.

Ejemplo 3.6. Aproximar numéricamente el sistema bidimensional de Lotka-Volterra.

$$\begin{cases} u_1'(t) = au_1(t) - bu_2(t)u_1(t) \\ u_2'(t) = -cu_2(t) + du_1(t)u_2(t) \end{cases}$$

Con valor en los parámetros a = b = c = d = 1, se obtiene una aproximación numérica de soluciones de este sistema.

========== Inicio de script ==========

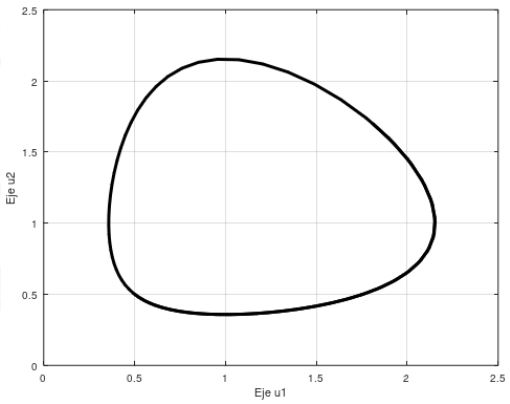
```
t=linspace(0, 10, 100); x0=[0.5; 0.5]; a=b=c=d=1;
dp=@(u, t)[a*u(1)-b*u(1)*u(2); -c*u(2)+d*u(1)*u(2)];
sol=lsode(dp, x0, t); disp([sol(:, 1), sol(:, 2)])
plot(sol(:, 1), sol(:, 2), 'k', 'linewidth', 1.5)
xlabel('Eje u1'); ylabel('Eje u2'); grid;
```

A continuación, se muestran las 10 primeras iteraciones de las soluciones aproximadas.

```
0.5000   0.5000
0.5265   0.4760
0.5558   0.4544
0.5879   0.4352
0.6229   0.4182
0.6611   0.4033
0.7027   0.3905
0.7477   0.3798
0.7964   0.3712
0.8489   0.3646
```

La gráfica de la ecuación de Lotka-Volterra se visualiza en la figura 3.6.

FIGURA 3. 6



Grafica del sistema de ecuaciones depredador-presa usando la función lsode.

3.3 Método de Euler

Euler utilizo el polinomio de Taylor de primer orden para deducir un método simple para aproximar la solución de una ecuación diferencial ordinaria.

$$\boldsymbol{w_{i+1} = w_i + hf(t_i, w_i) \qquad i = 0, 1, 2, \dots, N-1,}$$

Donde

h = tamaño de paso, con distribución uniforme en $[a, b]$.

$t_i = a + ih$, donde $i = 0, 1, 2, \dots, N$.

$w_i \approx y(t_i)$ representa el valor exacto de la solucion en t_i.

Ejemplo 3.7. Resuelva la siguiente EDO por el método analítico y graficar la solución.

$$\frac{dy}{dx} = \frac{x^2 - 1}{y^2}, \quad y(0) = 2$$

Solución analítica

Paso 1. Separar variables

$$y^2 dy = (x^2 - 1)dx$$

Paso 2. Integrar ambos lados

$$\frac{y^3}{3} = \frac{x^3}{3} - x + c$$

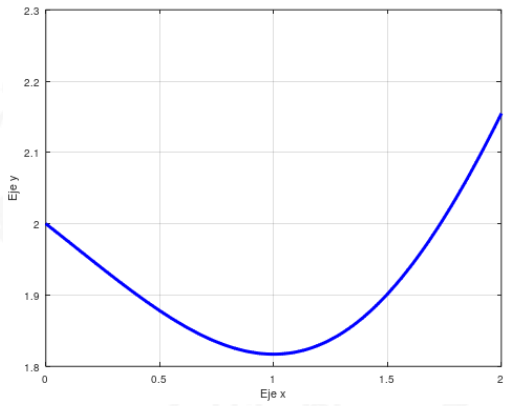
Paso 3. Despejar y

$$y = \sqrt[3]{x^3 - 3x + c}$$

Paso 4. Determinar el valor de c, usando las condiciones iniciales.

$$y = \sqrt[3]{x^3 - 3x + 8}$$

Gráfica de la solución del ejemplo 3.7.

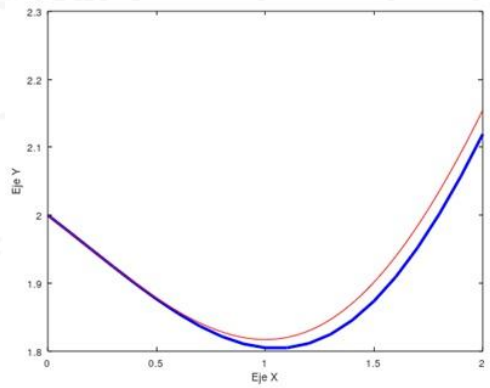


Resuelva el problema anterior, pero de forma numérica usando el método de Euler.

```
================ script ================
function R=Euler(a,b,ya,M)
% - Antes de correr el programa en octave deben
% cargar el siguiente paquete.
% pkg load symbolic
% syms x y t
% - [a, b]: dominio de la función.
% - ya: Valor inicial.
% - M: numéro de iteraciones.
f=@(x,y)(x*x-1)/(y*y);
h=(b-a)/M; T=zeros(1,M+1);
Y=zeros(1,M+1); T=a:h:b; Y(1)=ya;
for j=1:M
Y(j+1)=Y(j)+h*feval(f,T(j),Y(j));
end
R=[T' Y'];
plot(T,Y,'b','linewidth',1.5)
hold on
x=a:0.01:b; y=(x.^3-3*x+8).^(1/3);
plot(x,y,'r')
xlabel('Eje X'); ylabel('Eje Y');
endfunction
```

La grafica de color azul representa la solución aproximada y la gráfica de color rojo la solución exacta. Ver figura 3.8.

Figura 3.8.



Aproximación numérica de la ecuación $\frac{dy}{dx} = \frac{x^2-1}{y^2}$,

$y(0) = 2$, usando el método de Euler.

3.4 Método de Taylor de orden n

$W_0 = \alpha$

$W_{i+1} = W_i + hF^{(n)}(t_i, W_i)$

para cada $i = 0, 1, \dots, N-1$.

Donde

$$F^{(n)}(t_i, W_i) = f(t_i, W_i) + \frac{h}{2}f'(t_i, W_i) + \cdots + \frac{h^{n-1}}{n!}f^{(n-1)}(t_i, W_i)$$

Ejemplo 3.8. Aproximar numéricamente la solución de la siguiente ecuación diferencial con valor inicial.

$$y' = y - t^2 + 1\,, \quad 0 \le t \le 2\,, \quad y(0) = 0.5$$

Derivar la siguiente función $f\big(t, y(t)\big) = y(t) - t^2 + 1$ respecto a la variable t.

Para orden dos es el siguiente:

$$w_{i+1} = w_i + h\left[\left(1 + \frac{h}{2}\right)(w_i - t_i^2 + 1) - ht_i\right],$$

$$w_0 = 0.5.$$

$i = 0, 1, 2, \dots, N-1$.

Para orden cuatro:

$w_0 = 0.5$

$$w_{i+1} = w_i + h[\left(1 + \frac{h}{2} + \frac{h^2}{6} + \frac{h^3}{24}\right)(w_i - t_i^2) - \left(1 + \frac{h}{3} + \frac{h^2}{12} + \right)ht_i + 1 + \frac{h}{2} + \frac{h^2}{6} + \frac{h^3}{24}]$$

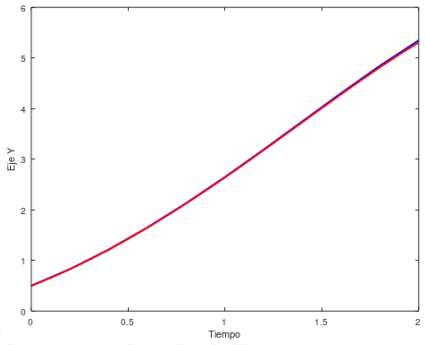
$i = 0, 1, \dots, N-1$.

Algoritmo de la serie de Taylor de orden dos.

```
==================== script ==============
function R=SerieTaylor2(a, b, ya, M)
% - [a, b]: dominio de la función.
% - ya: valor inicial.
% - M: numero de iteraciones.
f=input('ingrese f entre comillas, f(t, y)= ');
f=inline(f,'t','y'); h=(b-a)/M;
T=zeros(1,M+1); Y=zeros(1,M+1); T=a:h:b; Y(1)=ya;
for j=1:M
Y(j+1)=Y(j)+h*((1+h/2)*(Y(j)-T(j)^2+1)-h*T(j));
end
R=[T' Y'];
plot(T,Y,'b','linewidth',1.5)
xlabel('Tiempo'); ylabel('Eje Y');
hold on
t=linspace(0,2,100);
y=(t+1).^2-0.5*exp(t);
plot(t,y,'r','linewidth',1)
endfunction
```

La gráfica del ejemplo 3.8 se visualiza en la figura 3.9.

FIGURA 3. 9



La grafica rojo es solución real y la gráfica azul es la solución aproximada.

Burden & Faires. (2022), en su libro titulado "Análisis numérico" menciona diferentes metodos de aproximacion de soluciones de ecuaciones diferenciales ordinarias, a continuación se mencionan algunos de ellos.

3.5 Método de Runge Kutta

3.5.1 Método del punto medio

$W_0 = \alpha$

$W_{i+1} = W_i + hf(t_i + \frac{h}{2}, W_i + \frac{h}{2} f(t_i, W_i))$, para cada $i = 0, 1, \dots, N-1$.

3.5.2 Método modificado de Euler

$W_0 = \alpha$

$W_{i+1} = W_i + \frac{h}{2}[f(t_i, W_i) + f(t_{i+1}, W_i + hf(t_i, W_i))]$, para cada $i = 0, 1, \dots, N-1$.

3.5.3 Método de Heun

$W_0 = \alpha$

$W_{i+1} = W_i + \frac{h}{4}[f(t_i, W_i) + 3f\left(t_i + \frac{2}{3}h, W_i + \frac{2}{3}hf(t_i, W_i)\right)]$, para cada $i = 0, 1, \dots, N-1$.

3.5.4 Método de RK de orden cuatro.

$W_0 = \alpha$

$K_1 = hf(t_i, W_i)$

$K_2 = hf(t_i + \frac{h}{2}, W_i + \frac{K_1}{2})$

$K_3 = hf(t_i + \frac{h}{2}, W_i + \frac{K_2}{2})$

$K_4 = hf(t_i + h, W_i + K_3)$

$W_{i+1} = W_i + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4)$

Para cada $i = 0, 1, \dots, N-1$.

Ejemplo 3.9. Sea la ecuación diferencial $y' = 0.5 * (t - y)$ con $y(0) = 1$ en $[0, 3]$, usar Runge Kutta de orden cuatro para aproximar la ecuación de la ecuación diferencial.

========== script ==========

```
function R=rk4(a, b, ya, M)
% - [a, b]: dominio de la función.
% - ya: valor inicial.
% - M: número de iteraciones.
f=input('ingrese f entre comillas, f(t, y)= ');
```

```
f=inline(f, 't', 'y');
h=(b-a)/M; T=zeros(1, M+1); Y=zeros(1, M+1);
T=a:h:b; Y(1)=ya;
for j=1:M
K1=h*feval(f, T(j), Y(j));
K2=h*feval(f, T(j)+h/2, Y(j)+K1/2);
K3=h*feval(f, T(j)+h/2, Y(j)+K2/2);
K4=h*feval(f, T(j)+h, Y(j)+K3);
Y(j+1)=Y(j)+(K1+2*K2+2*K3+K4)/6;
end
R=[T' Y']; plot(T,Y,'b','linewidth',1.5);
xlabel('Tiempo'); ylabel('Eje Y')
endfunction
```

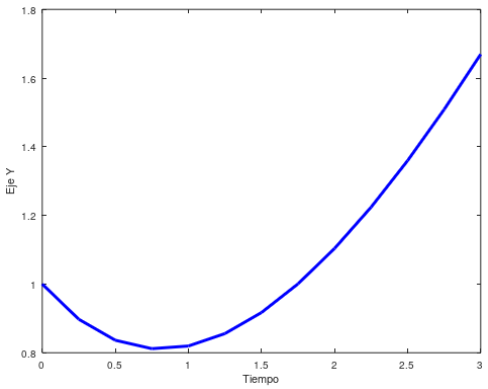
En la ventana de comandos de octave escribimos:

>> R=rk4(0,3,1,12) presione enter

ingrese f entre comillas, f(t, y) = '0.5*(t-y)'

La gráfica del ejemplo 3.9 se muestran en la figura 3.10.

FIGURA 3. 10



Aproximación numérica de la ecuación $y' = 0.5 * (t - y)$, usando el método de RK de orden 4.

3.6 Ecuación diferencial ordinaria de segundo orden

Mediante cambios de variables en una ecuación diferencial de orden n, se logra obtener un sistema de ecuaciones de primer orden como se detalla:

$$y''(x) + a_1 y'(x) + a_2 y(x) = R(x) \qquad (1)$$
$$y'(x_0) = y'_0$$
$$y(x_0) = y_0$$
Calcular: $y(x)$

Sea $v(x) = y'(x)$, reemplazando en (1) se obtiene la siguiente ecuación diferencial de primer orden con valor inicial.

$$y'(x) = v(x)$$
$$v'(x) = R(x) - a_1 v(x) - a_2 y(x)$$
$$y(x_0) = y_0$$
$$v(x_0) = y'_0$$

3.7 Ecuación diferencial ordinaria de tercer orden

$$y'''(x) + a_1 y''(x) + a_2 y'(x) + a_3 y(x) = R(x) \quad (2)$$

$$y''(x_0) = y''_0$$
$$y'(x_0) = y'_0$$
$$y(x_0) = y_0$$

Calcular: $y(x)$.

Sea

$$u(x) = y'(x), u'(x) = y''(x), v(x) = u'(x),$$

$$v'(x) = u''(x).$$

Reemplazando en (2) se obtiene la siguiente ecuación diferencial de primer orden con valores iniciales.

$$y'(x) = u(x)$$
$$u'(x) = v(x)$$
$$v'(x) = R(x) - a_1 v(x) - a_2 u(x) - a_3 y(x)$$

$$y(x_0) = y_0$$
$$u(x_0) = y'_0$$
$$v(x_0) = y''_0$$

3.8 Ecuación diferencial ordinaria de cuarto orden

$$y''''(x) + a_1 y'''(x) + a_2 y''(x) + a_3 y'(x) + a_4 y(x) = R(x) \ldots\ldots (3)$$

$$y'''(x_0) = y'''_0$$
$$y''(x_0) = y''_0$$
$$y'(x_0) = y'_0$$
$$y(x_0) = y_0$$

Calcular: $y(x)$.

Sea

$u(x) = y'(x), u'(x) = y''(x), v(x) = u'(x), v'(x) = u''(x), w(x) = v'(x), w'(x) = v''(x).$

reemplazando y despejando de la ecuación (3) se obtienen el siguiente sistema de ecuaciones diferenciales de primer orden con valores iniciales.

$$y'(x) = u(x)$$
$$u'(x) = v(x)$$
$$v'(x) = w(x)$$
$$w'(x) = R(x) - a_1 w(x) - a_2 v(x) - a_3 u(x) - a_4 y(x)$$

$$y(x_0) = y_0$$
$$u(x_0) = y'_0$$
$$v(x_0) = y''_0$$
$$w(x_0) = y'''_0$$

Ejemplo 3.10. Dada la ecuación diferencial

$$\text{PVI} \begin{cases} y'' + sen(x)y' + x^2 y = 3/x \\ y_0 = 1, \ \ y'_0 = -2. \ x_0 = 2. \end{cases}$$

Hallar el valor de $y(4)$, para $n = 4$.

Solución

========== script ==========

```
% - [a, b]: dominio de la función.
% - ya, yb: valor inicial.
% - M: numero de iteraciones.
function R=EulerS(a, b, ya, yb, M)
h=(b-a)/M; T=zeros(1, M+1);
Y1=zeros(1, M+1); Y2=zeros(1, M+1);
T=a:h:b; Y1(1)=ya; Y2(1)=yb;
f1=@(v)v; % función anónima uno
f2=@(t, y, v) 3/t-sin(t)*v-t^2*y; % función anónima dos.
for j=1:M
 Y1(j+1)=Y1(j)+h*feval(f1, Y2(j));
 Y2(j+1)=Y2(j)+h*feval(f2, T(j), Y1(j),Y2(j));
end
```

R=[T' Y1'];

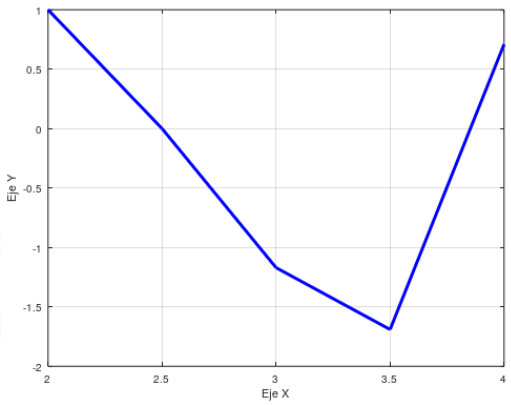
plot(T, Y1, 'b', 'linewidth', 1.5)

xlabel('Eje X'); ylabel('Eje Y');

endfunction

La gráfica del ejemplo 3.10 se visualiza en la figura 3.11

FIGURA 3. 11



Gráfica de la solución numérica de

$y'' + sen(x)y' + x^2y = \frac{3}{x}$, desarrollado con el método de Euler.

Ejemplo 3.11. Dada la ecuación diferencial

PVI $\begin{cases} y''' + 3xy'' + \cos(x)\, y' - xy = x^3 \\ y_0 = 1,\ \ y'_0 = -1,\ \ y''_0 = 3 \ \ , x_0 = 2. \end{cases}$

Hallar el valor de $y(4)$, para $n = 8$.

Solución

========== script ==========

function R=EulerS3(a, b, ya, yb, yc, M)

h=(b-a)/M; T=zeros(1,M+1);

Y1=zeros(1,M+1); Y2=zeros(1,M+1); Y3=zeros(1,M+1);

T=a:h:b; % variable independiente x=t;

Y1(1)=ya; Y2(1)=yb; Y3(1)=yc;

f1=@(u) u; % Primera ec. del sist.

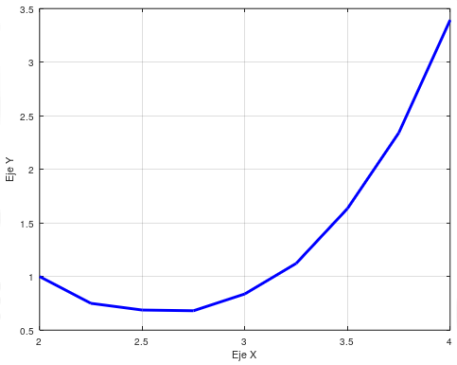
f2=@(v) v; % Segunda ec. del sist.

f3=@(t,y,u,v) t^3+t*y-cos(t)*u-3*t*v; % Tercera ec. del sist.

```
for j=1:M
Y1(j+1)=Y1(j)+h*feval(f1,Y2(j));      Y2(j+1)=Y2(j)+h*feval(f2,Y3(j));
Y3(j+1)=Y3(j)+h*feval(f3,T(j),Y1(j),Y2(j),Y3(j));
end
R=[T' Y1'];
plot(T,Y1,'b','linewidth',1.5)
xlabel('Eje X'); ylabel('Eje Y');
endfunction
```

La gráfica del ejemplo 3.11 se muestra en la figura 3.12.

FIGURA 3. 12



Gráfica de la solución numérica de $y''' + 3xy'' + \cos(x)\, y' - xy = x^3$, usando el método de Euler.

Taller N° 05 EDO con octave

1. Con el método de Euler aproxime las soluciones de los siguientes problemas de valor inicial.

a. $y' = te^{3t} - 2y$, $0 \le t \le 1$, $y(0) = 0$, con $h = 0.5$

b. $y' = 1 + (t - y)^2$, $2 \le t \le 3$, $y(2) = 1$, con $h = 0.5$

2. Con el método de Taylor de orden dos aproxime las soluciones de los siguientes problemas de valor inicial.

a. $y' = te^{3t} - 2y$, $0 \le t \le 1$, $y(0) = 0$, con $h = 0.5$

b. $y' = 1 + (t - y)^2$, $2 \le t \le 3$, $y(2) = 1$, con $h = 0.5$

3. Aplique Euler modificado para aproximar numéricamente las soluciones del ejercicio 1.

4. Desarrolle el ejercicio 1 usando el método de Heun.

5. Desarrolle el ejercicio 1 usando el método del punto medio.

6. Desarrolle el ejercicio 1 usando RK4.

Objetivo:

Aproximación numérica de soluciones en sistemas de ecuaciones diferenciales no lineales.

CONTENIDO:

- ✓ Preliminares.
- ✓ Sistema no lineal de Lorenz en $\mathbb{R}^4$.
- ✓ Visualización gráfica de las trayectorias de fase.

Capitulo 4

CAPITULO IV. ECUACIONES DIFERENCIALES NO LINEALES.

4.1 Modelo de proyección de Velezmoro.

Velezmoro et al, (2019), propusieron un modelo matemático que permite graficar objetos 4D en 3D siguiendo el mismo método para visualizar objetos 3D en un espacio bidimensional. Los modelos matemáticos se construyen utilizando un proyector de hologramas 3D.

4.1.1 Modelo de proyección 4D

Considere un cubo de dos unidades de longitud cuyo centro es el origen de coordenadas del espacio tridimensional. Cuatro ejes de coordenadas, x, y, z y w están ubicados en el cubo. (ver figura 4.1).