1) MÉTODO DE EULER

```
close all
clear all
clc
```

Tenemos:

```
dydx = @(x,y) -2*x^3+12*x^2-20*x+8.5;
```

Sujeta a la solución analítica:

```
yx = @(x) (1/-2)*x.^4+4*x.^3-10*x.^2+8.5*x+1;
```

Los datos son:

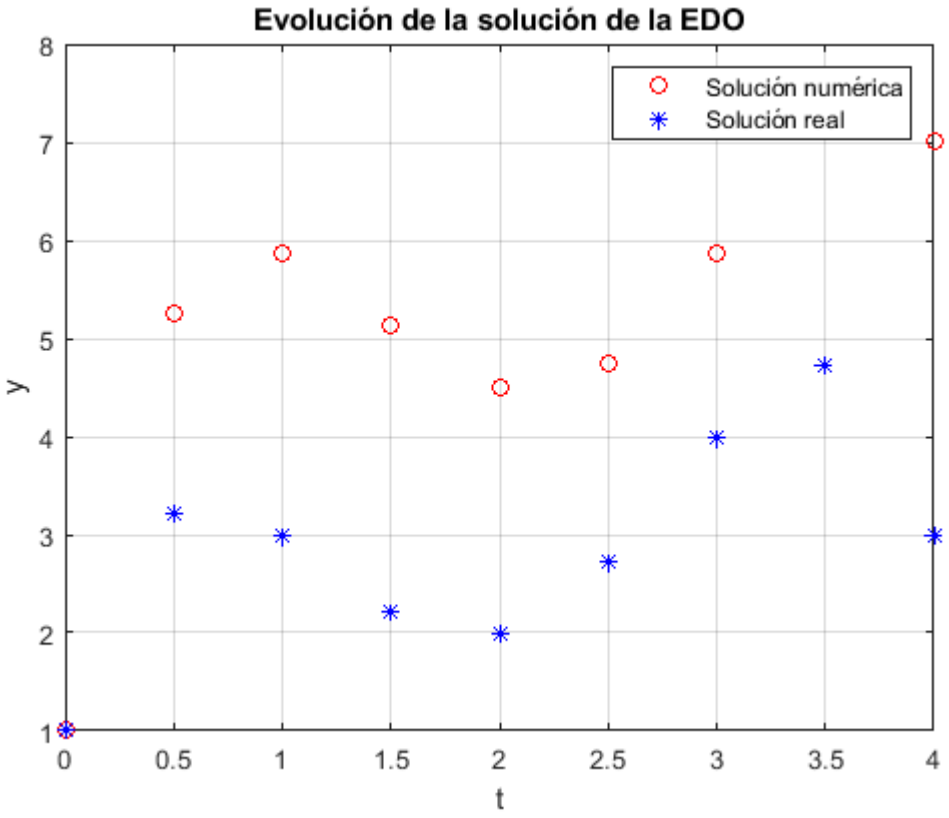
```
x0 = 0; xf = 4; h = 0.5; n = (xf-x0)/h;
y0 = 1;
```

El método de Taylor de orden 1 es el método de euler, usamos este caso particular:

```
[x, y, yreal] = TaylorEDO(dydx,yx,x0,y0,h,n);
```

Ploteamos lo obtenido y lo comparamos con la solución real:

```
figure(1)
plot(x,y,'ro',x,yreal,'b*')
title('Evolución de la solución de la EDO')
xlabel('t')
ylabel('y')
legend('Solución numérica','Solución real')
grid on
```

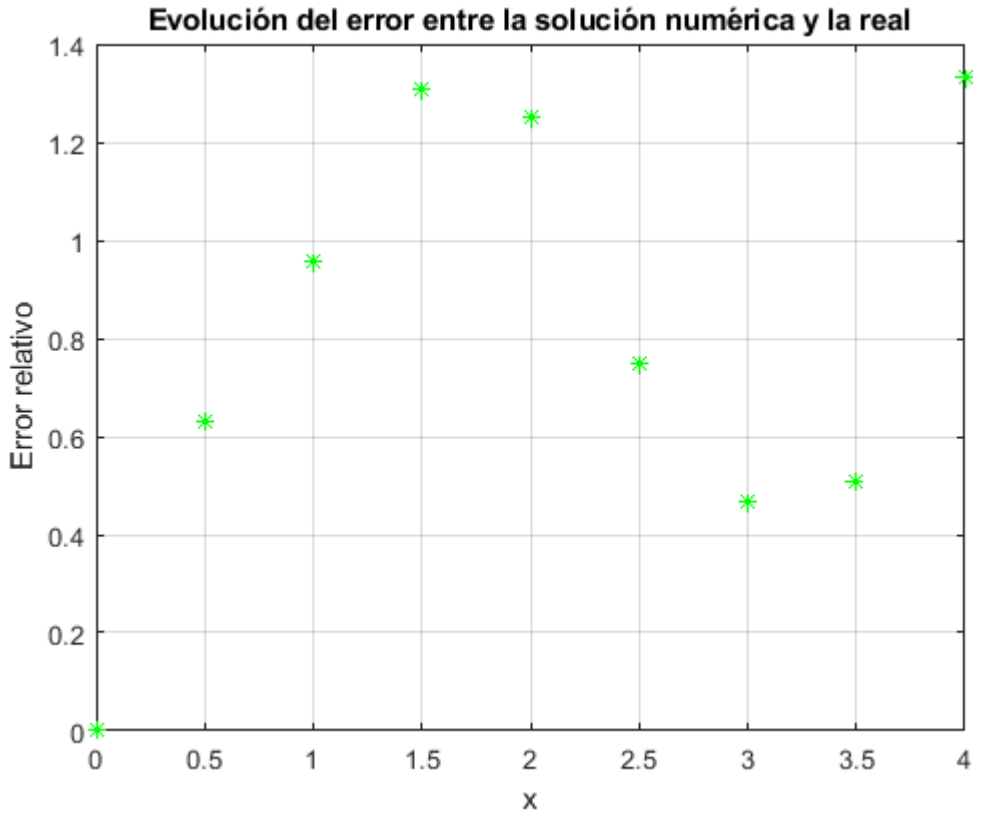


Introducimos un error relativo:

```
Err = abs((yreal-y)./yreal);
figure (2)
```

Ploteamos este error:

```
plot(x,Err,'g*')
title('Evolución del error entre la solución numérica y la real')
grid on
xlabel('x')
ylabel('Error relativo')
```



En resumen, se tiene la siguiente comparación:

```
Resumen = table(x',yreal',y',Err','VariableNames',{'x','y(x) (Real)','y(x) (Numérico)','Error r
```

ans = 9×4 table

	x	y(x) (Real)	y(x) (Numérico)	Error relativo
1	0	1.0000	1.0000	0
2	0.5000	3.2188	5.2500	0.6311
3	1.0000	3.0000	5.8750	0.9583
4	1.5000	2.2188	5.1250	1.3099
5	2.0000	2.0000	4.5000	1.2500
6	2.5000	2.7188	4.7500	0.7471
7	3.0000	4.0000	5.8750	0.4688
8	3.5000	4.7188	7.1250	0.5099
9	4.0000	3.0000	7.0000	1.3333

2) TAYLOR DE ORDEN 4 MATRICIAL

```
close all
clear all
clc
```

Como se demostró, nuestra matriz derivada tiene los siguientes elementos:

```
d11 = @(x,w1,w2) w2;
d21 = @(x,w1,w2) x^2*sin(x)+x*w1;
d31 = @(x,w1,w2) w1+x*w2+2*x*sin(x)+x^2*cos(x);
d41 = @(x,w1,w2) 2*w2+x^3*sin(x)+x^2*w1+2*x*sin(x)+x^2*cos(x);
d12 = @(x,w1,w2) d21(x,w1,w2);
d22 = @(x,w1,w2) d31(x,w1,w2);
d32 = @(x,w1,w2) d41(x,w1,w2);
d42 = @(x,w1,w2) 4*x*w1+4*x^2*sin(x)+x^3*cos(x)+w2+2*sin(x)+4*x*cos(x);
```

Es decir:

```
d = @(x,w1,w2) [d11(x,w1,w2) d12(x,w1,w2);...
    d21(x,w1,w2) d22(x,w1,w2);...
    d31(x,w1,w2) d32(x,w1,w2);...
    d41(x,w1,w2) d42(x,w1,w2)];
```

Datos:

```
x0 = 0; xf = 1; w10 = 1; w20 = 0; h = 0.1;
```

Aplicamos el método y guardamos lo obtenido:

```
[x,w1,w2] = TaylorMatricialOrden4(h,x0,xf,w10,w20,d);
```

SOLUCIÓN ANALÍTICA:

Hacemos el desarrollo de la EDO según la teoría de serie de potencias y truncamos la serie

para tener un resultado finito. Habiéndolo calculado en papel, tenemos:

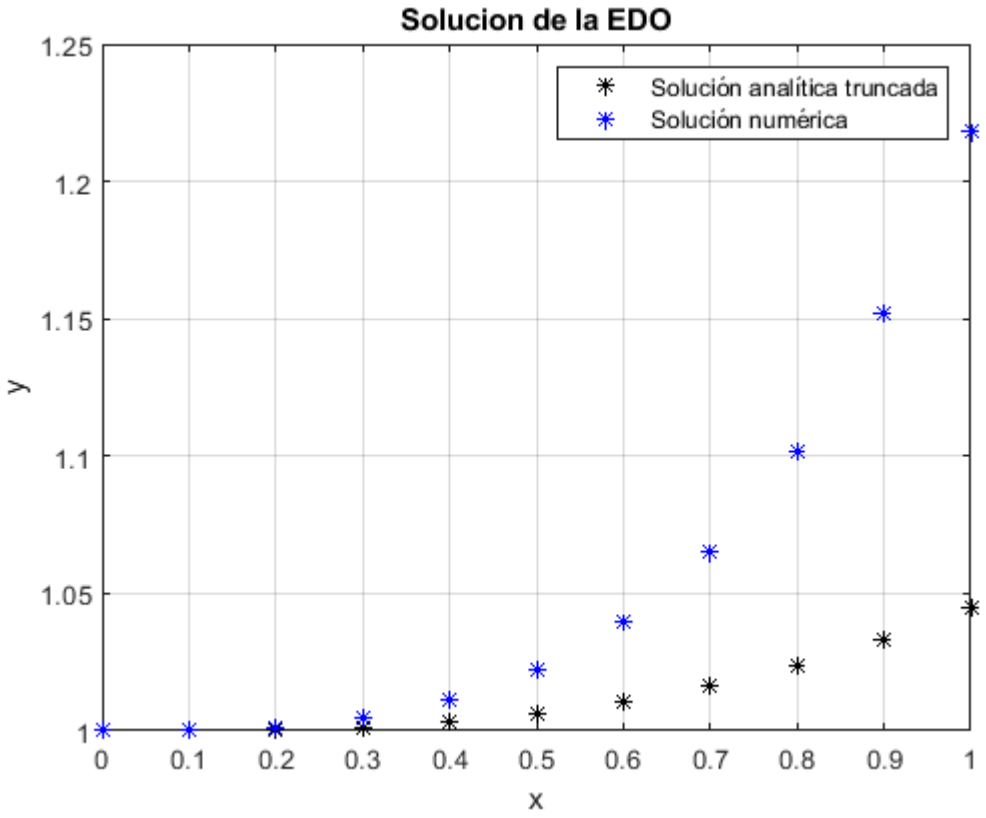
```
a3 = 1/20;
a4 = (-a3/30);
a5 = ((-1/6)-a4)/42;
a6 = (-a5)/56;
```

La solución analítica sería:

```
yanal = @(x) 1 + a3*x.^3+a4*x.^4+a5*x.^5+a6*x.^6;
figure (3)
```

Ploteamos comparando tanto la solución analítica junto con la numérica:

```
plot(x,yanal(x),'k*',x,w1,'b*')
title('Solucion de la EDO')
legend('Solución analítica truncada', 'Solución numérica')
grid on
xlabel('x')
ylabel('y')
```



Vemos que se comportan de manera semejante. Ahora bien, tenemos el error:

```
Err = abs((yanal(x)-w1)./yanal(x));
Resumen = table(x',yanal(x)',w1',Err','VariableNames',{'x','y(x) (Analítico)','y(x) (Numérico)'
```

ans = 11×4 table

	x	y(x) (Analítico)	y(x) (Numérico)	Error relativo
1	0	1.0000	1.0000	0
2	0.1000	1.0000	1.0002	0.0001
3	0.2000	1.0004	1.0013	0.0009
4	0.3000	1.0013	1.0046	0.0033
5	0.4000	1.0031	1.0111	0.0080
6	0.5000	1.0060	1.0223	0.0162
7	0.6000	1.0103	1.0398	0.0292
8	0.7000	1.0161	1.0656	0.0487
9	0.8000	1.0236	1.1020	0.0765
10	0.9000	1.0331	1.1518	0.1149
11	1.0000	1.0445	1.2183	0.1665

3) USO DE LA TEORÍA DE BUTCHER

SOLUCIÓN POR MI MÉTODO RK

```
close all
clear all
clc
```

El sistema resultante es:

```
dw = @(x,w1,w2) [w2;-2*x*w1];
```

Sujeto a:

```
x0 = 0; xf = 5; w10 = 1; w20 = 0;
```

Proponemos el numero de etapas y un factor de normalizacion para los pesos:

```
n = 3;
m = 6;
```

Creamos nuestra tabla de butcher:

```
[A,b,c] = TablaButcher(n,m)
```

```
A = 3×3
     0     0     0
     2     0     0
     3     3     0
b = 1×3
    0.4615    0.3077    0.2308
c = 3×1
     0
     2
     6
```

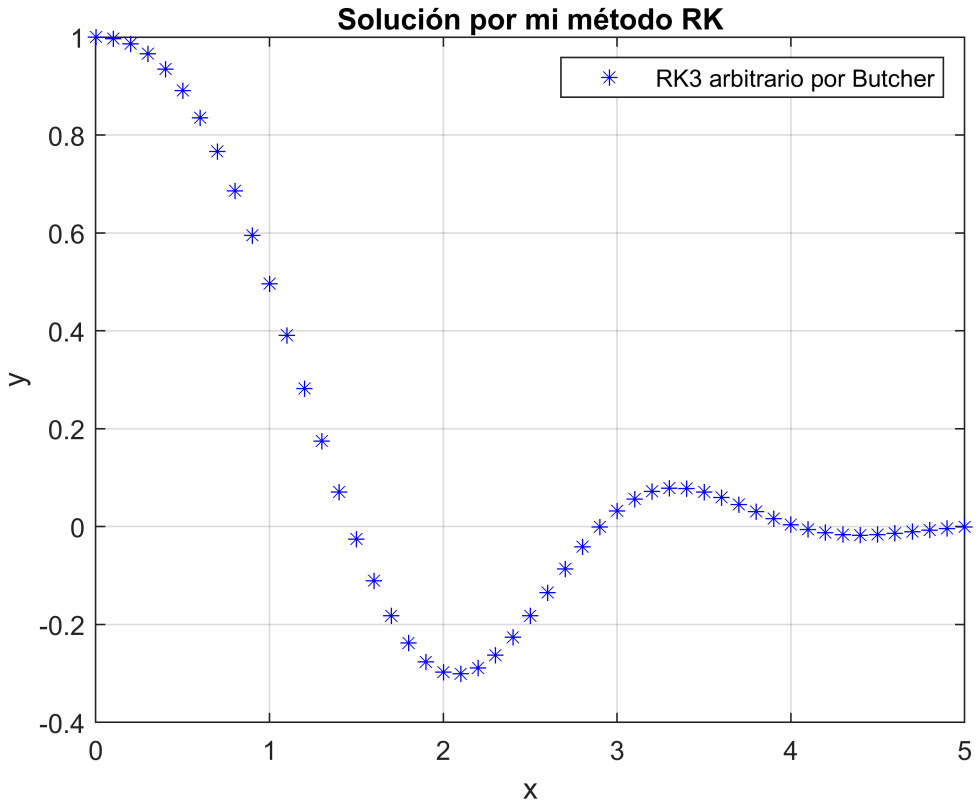
```
xj = x0; w1j = w10; w2j = w20; h = 0.1;
```

Aplicamos nuestro método RK creado:

```
[x,wj] = RKButcher(dw,xj,xf,w1j,w2j,h,n,A,b,c);
figure (4)
```

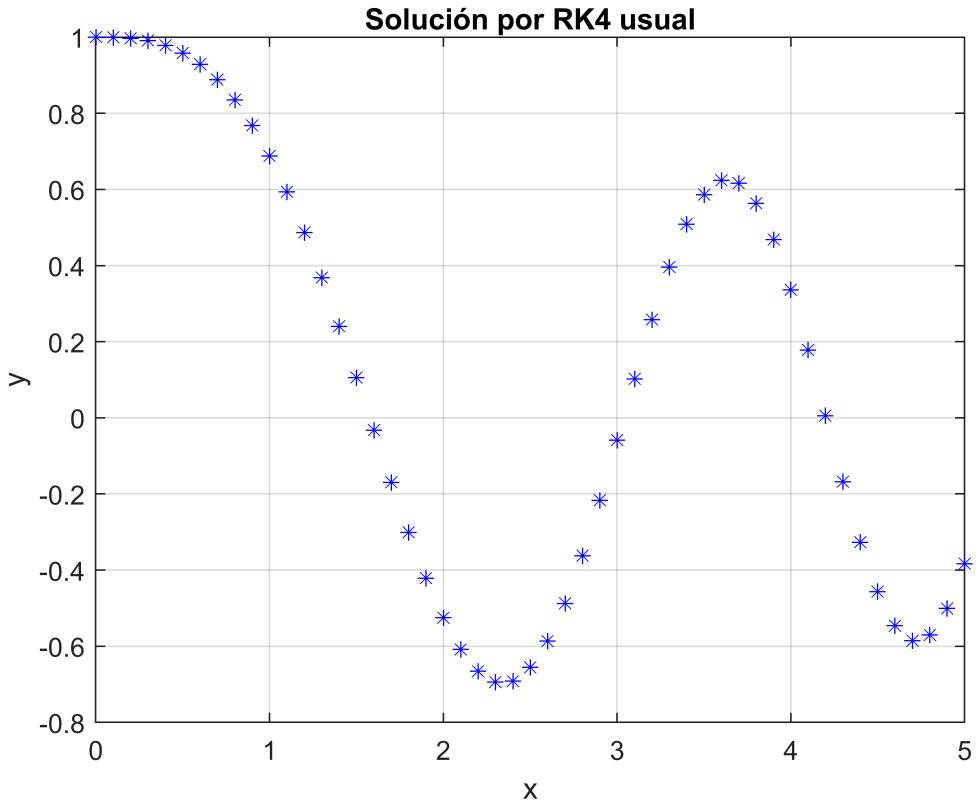
Ploteamos la solución obtenida:

```
plot(x,wj(1,:),'b*')
grid on
xlabel('x')
ylabel('y')
title('Solución por mi método RK')
legend('RK3 arbitrario por Butcher')
```



SOLUCIÓN POR RK4 USUAL:

```
z = @(x,w) [w(2);-2*x*w(1)];
z0 = [w10;w20];
[x,z] = RK4(z,x0,xf,z0,h);
figure(5)
plot(x,z(1,:),'b*')
grid on
xlabel('x')
ylabel('y')
title('Solución por RK4 usual')
```



SOLUCIÓN ANALÍTICA:

Resolvemos la EDO usando el método de serie de potencias y truncamos la serie para hacerla finita.

Tomamos el factor:

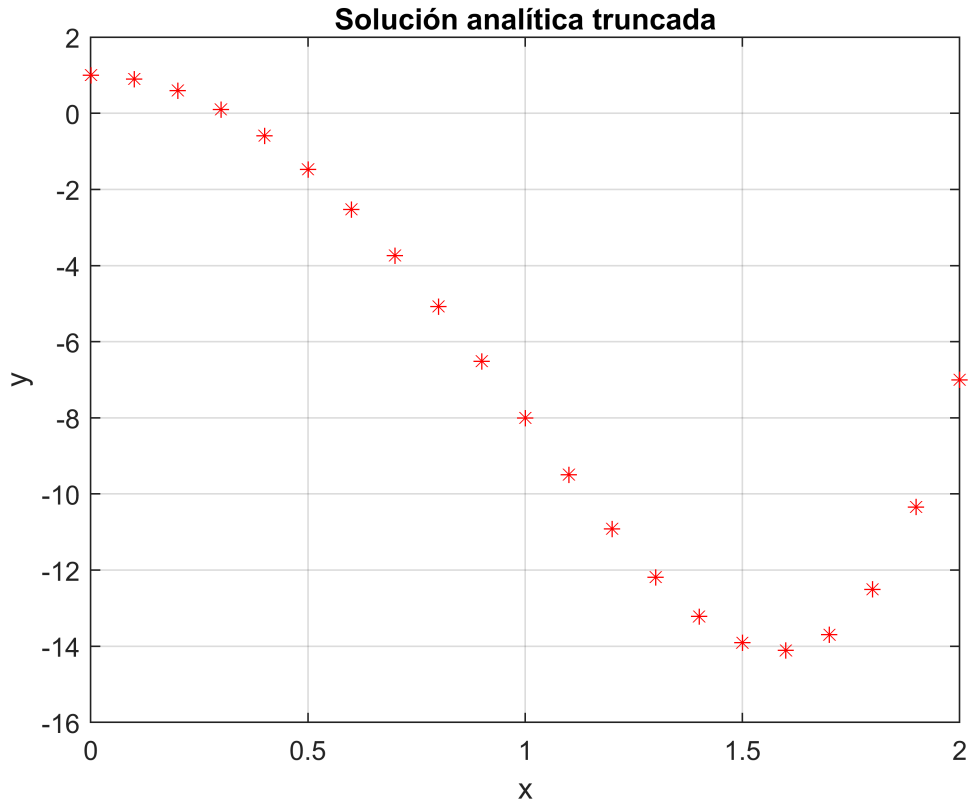
```
a5 = 1;
```

La función solución truncada es, entonces:

```
yanal = @(x) 1+a5*(-10*x.^2+x.^5);
```

Ploteamos en una vecindad cerca del cero, para mejor visualización de la aproximación:

```
plot([0:h:2],yanal(0:h:2),'r*')
grid on
xlabel('x')
ylabel('y')
title('Solución analítica truncada')
```



Vemos que se comporta de manera similar.

```
Err = abs((z(1,:)-wj(1,:))./z(1,:));
```

En resumen, el error es:

```
Resumen = table(x',z(1,:)',wj(1,:)',Err','VariableNames',{'x','y(x) (RK4)','y(x) (Mi RK3)','Err
```

Resumen = 51×4 table

	x	y(x) (RK4)	y(x) (Mi RK3)	Error relativo
1	0	1.0000	1.0000	0
2	0.1000	0.9997	0.9972	0.0024
3	0.2000	0.9973	0.9865	0.0109
4	0.3000	0.9910	0.9661	0.0251
5	0.4000	0.9788	0.9347	0.0450
6	0.5000	0.9587	0.8913	0.0703
7	0.6000	0.9290	0.8352	0.1010
8	0.7000	0.8883	0.7667	0.1369
9	0.8000	0.8351	0.6863	0.1782
10	0.9000	0.7686	0.5954	0.2253
11	1.0000	0.6883	0.4961	0.2793

	x	y(x) (RK4)	y(x) (Mi RK3)	Error relativo
12	1.1000	0.5943	0.3907	0.3425
13	1.2000	0.4873	0.2825	0.4202
14	1.3000	0.3686	0.1747	0.5260
15	1.4000	0.2404	0.0710	0.7047
16	1.5000	0.1056	-0.0252	1.2385
17	1.6000	-0.0324	-0.1105	2.4121
18	1.7000	-0.1693	-0.1822	0.0762
19	1.8000	-0.3004	-0.2380	0.2078
20	1.9000	-0.4206	-0.2765	0.3426
21	2.0000	-0.5249	-0.2974	0.4335
22	2.1000	-0.6083	-0.3010	0.5051
23	2.2000	-0.6662	-0.2889	0.5664
24	2.3000	-0.6948	-0.2630	0.6214
25	2.4000	-0.6916	-0.2265	0.6725
26	2.5000	-0.6554	-0.1825	0.7215
27	2.6000	-0.5865	-0.1346	0.7704
28	2.7000	-0.4873	-0.0863	0.8228
29	2.8000	-0.3619	-0.0407	0.8875
30	2.9000	-0.2164	-0.0005	0.9978
31	3.0000	-0.0585	0.0324	1.5541
32	3.1000	0.1028	0.0568	0.4477
33	3.2000	0.2578	0.0722	0.7199
34	3.3000	0.3963	0.0790	0.8006
35	3.4000	0.5088	0.0782	0.8463
36	3.5000	0.5868	0.0712	0.8787
37	3.6000	0.6240	0.0598	0.9042
38	3.7000	0.6165	0.0457	0.9259
39	3.8000	0.5637	0.0308	0.9454
40	3.9000	0.4683	0.0166	0.9646
41	4.0000	0.3367	0.0042	0.9876
42	4.1000	0.1784	-0.0056	1.0313
43	4.2000	0.0056	-0.0124	3.1966
44	4.3000	-0.1676	-0.0161	0.9036
45	4.4000	-0.3264	-0.0173	0.9471

	x	y(x) (RK4)	y(x) (Mi RK3)	Error relativo
46	4.5000	-0.4566	-0.0163	0.9644
47	4.6000	-0.5461	-0.0138	0.9748
48	4.7000	-0.5857	-0.0104	0.9822
49	4.8000	-0.5706	-0.0068	0.9880
50	4.9000	-0.5012	-0.0035	0.9930
51	5.0000	-0.3832	-0.0007	0.9982

```
function [x, y, yreal] = TaylorEDO(dnf,yx,x0,y0,h,n)
    ntaylor = length(dnf(x0,y0));
    staylor = y0;
    x(1) = x0;
    y(1) = y0;
    vj = dnf(x0,y0);
    for i = 1:n
    if i>=2
        staylor = y(i);
        vj = dnf(x(i),y(i));
    end
    for j = 1:ntaylor
        y(i+1) = staylor + (((h^j)*vj(j))/factorial(j));
        staylor = y(i+1);
    end
        x(i+1) = x(i)+h;
    end
    yreal = yx(x);
end
function [x,w1,w2] = TaylorMatricialOrden4(h,x0,xf,w10,w20,d)
    d0 = d(x0,w10,w20); [nr,~] = size(d0);
    for i = 1:nr
        E(i,i) = (h^i)/factorial(i);
    end
    dtay = E*d0; n = (xf-x0)/h;
    x(1,1) = x0; w1(1,1) = w10; w2(1,1) = w20;
    for i = 1:n
        w1(1,i+1) = w1(1,i) + sum(dtay(:,1));
        w2(1,i+1) = w2(1,i) + sum(dtay(:,2));
        x(1,i+1) = x(1,i) + h;
        dtay = E*d(x(1,i+1),w1(1,i+1),w2(1,i+1));
    end
end
function [A,b,c] = TablaButcher(n,m)
    A = zeros(n);
    c = zeros(n,1);
    rand = randi([1,m],1,n);
    b = rand./sum(rand);
    for i = 2:n
        A(i,1:i-1) = randi([1,m],1,i-1);
    end
```

```
    for i = 1:n
        c(i) = sum(A(i,:));
    end
end
function [x,wj] = RKButcher(dw,xj,xf,w1j,w2j,h,n,A,b,c)
    K = zeros(2,n);
    np = (xf-xj)/h;
    wj(:,1) = [w1j;w2j];
    x(1,1) = xj;
    for j = 1:np
        for i = 1:n
            if i == 1
                K(:,i) = dw(x(1,j),w1j,w2j);
            else
                K(:,i) = dw(x(1,j)+c(i)*h,w1j+h*sum(K(1,:).*A(i,:)),w2j+h*sum(K(2,:).*A(i,:)));
            end
        end
        w1j = w1j+h*sum(K(1,:).*b);
        w2j = w2j+h*sum(K(2,:).*b);
        wj(:,j+1) = [w1j;w2j];
        x(1,j+1) = x(1,j) + h;
    end
end
function [x,z] = RK4(df,t0,tf,z0,h)
    x = [t0:h:tf]; n = (tf-t0)/h;
    z(:,1) = z0;
    for i = 1:n
        k1 = df(x(i),z(:,i));
        k2 = df(x(i)+(h/2),z(:,i)+(h/2)*k1);
        k3 = df(x(i)+(h/2),z(:,i)+(h/2)*k2);
        k4 = df(x(i)+h,z(:,i)+h*k3);
        z(:,i+1) = z(:,i)+(h/6)*(k1+2*k2+2*k3+k4);
    end
end
```