6.1. RESOLUCIÓN NUMÉRICA DE PROBLEMAS DE VALORES INICIALES

Muchos problemas de ingeniería se pueden formular en términos problemas de valores iniciales para ecuaciones diferenciales ordinarias. Por ejemplo, la segunda ley de Newton en mecánica clásica, ecuaciones de redes eléctricas en Electrotecnia, problemas sobre dinámica de poblaciones en ecología.

En general, no siempre es posible resolver de manera analítica el problema de valores iniciales (Problema de Cauchy)

$$\begin{aligned} y'(x) &= f(x, y(x)) \quad , \quad x \in [x_0, x_0 + a] \\ y(x_0) &= y_0 \end{aligned} \tag{1}$$

Por tanto debemos encontrar procedimientos numéricos para obtener funciones que se aproximen suficientemente a la solución del problema de Cauchy propuesto.

Esperamos obtener valores aproximados $y_1, y_2 \ldots\ldots y_n$ de la función y(x) correspondientes a un conjunto finito de valores $x_1, x_2 \ldots\ldots x_n$ en un intervalo $[x_0, x_0 + a]$.

Uniendo mediante segmentos rectilíneos los puntos (x_0, y_0), (x_1, y_1)..... (x_n, y_n) se obtiene una poligonal que es aproximación a la solución y = y(x) en el intervalo $[x_0, x_0 + a]$

Sin embargo, esta aproximación, no resulta conveniente en la práctica, pues privilegia los puntos x_k en los que la función aproximante no es diferenciable en general (presenta angulosidades). Podemos mejorar la aproximación, a la vez que se hace más suave sustituyendo la curva poligonal por otra más regular, obtenida mediante interpolación polinomica.

6.2 .Integración numérica por serie de Taylor

Vamos a considerar que los puntos x_n, n = 0, 1,2….. están igualmente espaciados $x_n = x_0 + n\,h$, h es la amplitud del intervalo entre dos pasos consecutivos. Supuesta la función suficientemente regular (esta condición suele verificarse en los problemas de Ingeniería), se satisface el teorema de existencia y unicidad de la solución y de acuerdo con el teorema de Taylor se tiene

$$y\ (x) = y(x_0) + (x - x_0)\ y'(x_0) + \frac{(x-x_0)^2}{2!}\ y''(x_0) + \cdots .... \tag{2}$$

Las derivadas en este desarrollo no se conocen explícitamente ya que la solución no es conocida. Sin embargo, si f es suficientemente diferenciable, pueden obtenerse tomando la derivada total de (1) con respecto a x. De esta forma se tiene

$$y' = f(x, y)$$

$$y'' = f_x + f_y\, y' = f_x + f_y\, f$$

$$y''' = f'' = f_{xx} + 2\, f_{xy}\, f + f_{yy}\, f^2 + f_x\, f_y + f_{y^2}\, f$$

Por tanto, podemos expresar cualquier derivada de *y* en términos de f(x, *y)* y sus derivadas parciales. Sin embargo a menos que f(x, y) sea una función muy simple, las derivadas totales de orden superior son difíciles de calcular. Por razones prácticas, debemos limitar el número de términos del desarrollo (2), esta restricción conduce a una restricción en el valor de x para el cual (2) es una aproximación razonable. Si suponemos que la serie truncada (2) es una buena aproximación para un paso de longitud *h*, es decir, $x - x_0 = h$, podemos evaluar y en $x = x_0 + h$; a continuación calcular y', y'' ...en $x_0 + h$ y luego usar la expresión (2). Si continuamos de esta manera, obtendremos un conjunto discreto de valores y_n que son aproximaciones de la verdadera solución en los puntos $x_n = x_0 + n\,h$, (n = 0, 1,2….). Vamos a denominar el valor de la solución exacta en un punto x_n por $y(x_n)$ y el valor de una solución aproximada por y_n.

Para formalizar el procedimiento, introducimos el operador

$$T_k(x, y) = f(x, y) + \frac{h}{2} f'(x, y) + \cdots \ldots \ldots + \frac{h^{k-1}}{k!} f^{k-1}(x, y) \quad , k = 1,2 \ldots .. \qquad (3)$$

En la expresión (3) se ha utilizado un paso de longitud h, y f^k denota la k-ésima derivada total de f(x, y(x)) con respecto a x. Vamos ahora a enunciar el algoritmo

6.2.1.Algoritmo de Taylor de orden k. Se trata de calcular en un intervalo [a, b] una solución aproximada de la ecuación diferencial

$$y' = f(x, y) \quad , \qquad y(a) = y_0$$

1. Se escoge un paso $h = \frac{(b-a)}{N}$. A continuación se toma

$x_0 = a \quad , \quad x_n = a + n\,h \quad , \quad y(x_n) = y(a + n\,h) \quad , \quad x_n = b$

2. Obtenemos aproximaciones y_n a $y(x_n)$ a partir de la fórmula

$$y_{n+1} = y_n + h\,T_k\,(x_n, y_n) \qquad n = 0, 1, \ldots\ldots N-1$$

Donde $T_k\,(x, y)$ viene dada por la expresión (3). El error local del algoritmo de Taylor de orden k es

$$E = \frac{h^{k+1}\ f^k\ (\alpha, y(\alpha))}{(k+1)!} = \frac{h^{k+1}}{(k+1)!} y^{k+1}(\alpha) \qquad x_n < \alpha < x_n + h$$

Para k = 1 en el algoritmo de Taylor obtenemos el método de Euler y el error local viene dado por:

$$y_{n+1} = y_n + h\,f(x_n, y_n) \qquad , E = \frac{h^2}{2!}\ y''(\alpha) \tag{4}$$

6.3. Estimación del error y convergencia del método de Euler

Para resolver la ecuación diferencial $y' = f(x, y)$, $y(x_0) = y_0$ por el método de Euler, escogemos un tamaño de paso constante h, y aplicamos la fórmula

$$y_{n+1} = y_n + h\,f(x_n, y_n)$$

Donde $x_n = x_0 + n\,h$. A la verdadera solución de la ecuación diferencial en el punto x_n la denominamos $y(x_n)$, y la solución obtenida aplicando la formula (4) y_n. Queremos estimar la magnitud del error de discretización o de truncamiento definido por

$$e_n = y\,(x_n) - y_n \tag{5}$$

Observamos que, si y_0 es una solución exacta, como suponemos, entonces e_0=0 .Suponiendo que las derivadas que necesitamos existen, podemos desarrollar $y(x_{n+1})$ con respecto $x = x_n$, usando el teorema de Taylor

$$y(x_{n+1}) = y(x_n) + h\ y'(x_n) + \frac{h^2}{2}\ y''(\alpha) \qquad x_n \leq \alpha \leq x_{n+1} \tag{6}$$

A la cantidad $\frac{h^2}{2}\ y''(\alpha)$ se la denomina error local por truncamiento, es decir es el error cometido en el paso de x_n , a x_{n+1}, suponiendo que y , y ' son conocidas exactamente en x = x_n. En una computadora habrá también un error al calcular y_{n+1} utilizando la expresión (4), debido al redondeo. No se tendrán en cuenta los errores de redondeo

Si restamos la expresión (4) de la expresión (6) y usamos la expresión (5), obtenemos

$$e_{n+1} = e_n + h[\, f(x_n ,\ y(x_n)) - f(x_n ,y_n)] + \frac{h^2}{2}\ y''(\alpha) \tag{7}$$

Ahora bien, por el teorema del valor medio del cálculo diferencial, tenemos

$$f(x_n , y(x_n)) - f(x_n\ , y_n) = f_y\,(x_n ,\ \bar{y}_n)\ (y(x_n) - y_n) = f_y\ (x_n , \bar{y}_n)\ e_n$$

Donde $\bar{y}_n$ está entre y_n e $y(x_n)$. Por tanto, la expresión (7) se transforma en

$$e_{n+1} = e_n + h\ f_y\,(x_n ,\ \bar{y}_n)\ e_n + \frac{h^2}{2}\ y''(\alpha) \tag{8}$$

Vamos a suponer ahora que sobre el intervalo considerado, se tiene

$$\left|f_y\,(x,y)\right| < L \quad , \ |y''(x)| < M$$

Donde L e M son constantes positivas prefijadas. Tomando valores absolutos en la expresión (8) se tiene

$$|e_{n+1}| \leq |e_n| + h\,L\,|e_n| + \frac{h^2}{2}\,M = (1 + h\,L)|e_n| + \frac{h^2}{2}\,M$$

Teorema 1. Sea y_n la solución aproximada de la ecuación diferencial (1) obtenida mediante el método de Euler. Si la solución exacta y(x) de la ecuación (1) tiene una segunda derivada continua en el intervalo $[x_0, x_0 + h]$ y si sobre este intervalo se satisfacen las desigualdades

$$|f_y(x, y)| < L \quad , \; |y''(x)| < M$$

Para constantes positivas dadas L e M el error $e_n = y(x_n) - y_n$ del método de Euler en un punto $x_n = x_0 + n\,h$ estará acotado de la forma siguiente

$$|e_n| \leq \frac{h\,M}{2\,L}\left(e^{(x_n - x_0)L} - 1\right) \tag{9}$$

Por tanto, el error es de la forma O (h), por tanto, el error tiende acero cuando h→0 como C h para alguna constante C, si $x_n = x$ se mantiene fijo. La característica principal de este error es establecer la convergencia del método más que proporcionar una estimación del error real

6.4. Método de Runge-Kutta

Los métodos anteriores se llaman método de un paso porque la aproximación en x_{n+1} es una función del valor aproximado en x_n un paso hacia atrás. Vamos a ver otros métodos de un paso que son en general más exactos que el método de Euler, como son los métodos de Runge -Kutta. La idea del método es promediar en cada paso, con pesos adecuados, los valores f(x, y) en los extremos y también en puntos interiores del intervalo $[x_n, x_{n+1}]$ de anchura h, en que se está trabajando. El desarrollo de esta idea da lugar a toda una familia de métodos de Runge –Kutta

Los métodos de Runge -Kutta tienen la exactitud del esquema de la serie de Taylor sin necesidad del cálculo de derivadas superiores. El método de R-K de primer orden con n =1 es, el método de Euler. Dentro de los métodos de R dos etapas y orden dos es particularmente conocido: El método de Euler mejorado. El método de R-K por excelencia es un método de cuatro etapas de cuarto orden

El método de Euler no es muy útil en las aplicaciones prácticas porque requiere un tamaño de paso muy pequeño para una precisión razonable. El algoritmo de Taylor de orden superior tiene el inconveniente de que necesitamos obtener derivadas totales de un orden alto de *y(x)*. Los métodos de Runge-Kutta tratan de obtener mayor precisión, y al mismo tiempo evitan la necesidad de derivadas de orden superior, calculando la función *f(x, y)* en puntos seleccionados de cada subintervalo. Vamos a deducir el más simple de los métodos de Runge-Kutta. Se busca una fórmula de la siguiente forma

$$y_{n+1} = y_n + a\,k_1 + b\,k_2 \tag{10}$$

Donde

$$k_1 = h\,f(x_n, y_n)$$

$$k_2 = h\,f(x_n + c\,h, y_n + d\,k_1)$$

a, *b*, *c* , *d* son constantes que deben calcularse de modo que la expresión (10) esté de acuerdo con el algoritmo de Taylor . Desarrollando $y(x_{n+1})$ en serie de Taylor hasta el orden h^3, obtenemos

$$y(x_{n+1}) = y(x_n) + h\,y'(x_n) + \frac{h^2}{2}y''(x_n) + \frac{h^3}{3!}y'''(x_n) + \cdots \ldots \ldots \ldots =$$

$$= y(x_n) + h\,f(x_n, y_n) + \frac{h^2}{2}(f_x + f\,f_y)_n +$$

$$\frac{h^3}{6}(f_{xx} + 2f\,f_{xy} + f_{yy}\,f^2 + f_x\,f_y + f_{y^2}f)_n + 0(h^4) \tag{11}$$

Donde hemos utilizado los siguientes desarrollos

$$y' = f(x, y)$$

$$y'' = f' = f_x + f_y\,y' = f_x + f_y\,y' = f_x + f_y\,f$$

$$y''' = f'' = f_{xx} + 2\, f_{xy}\, f + f_{yy}\, f^2 + f_x\, f_y + f_{y^2}\, f$$

El subíndice *n* significa las funciones se evalúan en (x_n , y_n). Por otra parte, usando el desarrollo de Taylor para funciones de dos variables, se tiene que

$\frac{k_2}{h}$ =f$(x_n + c\, h\, , y_n + d\, k_1) = f(\, x_n, y_n\,) + c\, h\, f_x$ + d $k_1\, f_y + \frac{c^2 h^2}{2} f_{xx} + c\, h\, d\, k_1\, f_{xy} + \frac{d^2\ k_1^2}{2}\, f_{yy} +$ $0(h^3)$

Todas las derivadas están evaluadas en $\{x_n\, , y_n\}$. Si sustituimos la expresión de k_2 en (10) y tenemos en cuenta que $k_1 = h\, f\, (x_n\, , y_n\,)$, se obtiene

$$y_{n+1} = y_n + (a+b)h\, f + \ b\ h^2\, (\, c\, f_x + d\, f\, f_y) + \ b\, h^3 \left(\frac{c^2}{2}\ f_{xx} + c\, df\, f_{xy} + \frac{d^2}{2}\ f^2 f_{yy}\ \right) + 0(h^4) \qquad (12)$$

Comparando las formulas (12) y (11), y haciendo coincidir potencias correspondientes de *h* y h^2 se debe verificar

$$a + b = 1$$

$$b\ c = b\, d = \frac{1}{2} \qquad (13)$$

En la ecuación (13), tenemos cuatro incógnitas, y tres ecuaciones por tanto tenemos un grado de libertad en la solucion de la ecuación

Hay muchas soluciones de la ecuación (13), de las cuales quizá la más simple es

$$a = b = 1/2 \qquad c = d = 1$$

De donde obtenemos

$$y_{n+1} = y_n + \frac{1}{2}(k_1 + \ k_2\ ) = y_n + \frac{h}{2}\,[f(x_n, y_n) + f\big(x_n\ + h\, , y_n + h\ f(x_n\, , y_n)\big)] \qquad (14)$$

A este método también se le conoce con el nombre de método de Euler mejorado

6.4.1. Algoritmo. Método de Runge-Kutta de orden 2. Sea la ecuación

$$y' = f(x, y) \quad , y(x_0) = y_0$$

Se trata de ir calculando aproximaciones y_n a $y(x_0 + n\ h)$, h fijo y n = 0, 1,2…. usando la fórmula (14)

Vamos a representar geométricamente el algoritmo anterior (Fig. 1). El método de Euler da un incremento $P_1 P_0 = h\ f(x_n, y_n)$ a y_n ; $P_2 P_0$ = h f(x_n +h , $y_n + h\ f(x_n, y_n)$) . Tomando el promedio de estos incrementos obtenemos la fórmula (14).

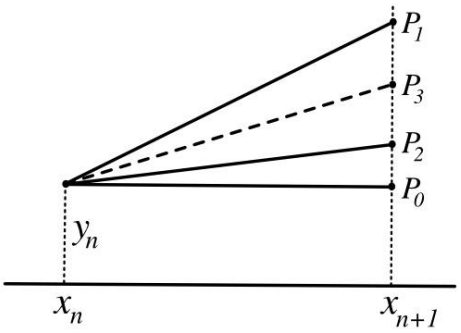


Fig.1

El error local en la formula (14) es de la forma

$$y(x_{n+1}) - y_{n+1} = \frac{h^3}{12}\left(f_{xx} + 2\,f\,f_{xy} + f_{yy}\,f^2 - 2\,f_x\,f_y - 2f_{y^2}f\right) + 0(h^4)$$

El error local es de orden h ^{3} mientras que en el método de Euler es de orden h^2. Esperamos que se pueda usar un tamaño más grande de paso con la fórmula (14).El precio a pagar por ello es que debemos evaluar la función f(x, y) dos veces para cada paso de integración.

6.4.2. Algoritmo de Runge-Kutta de orden 4. Sea la ecuación $y' = f(x, y), y(x_0) = y_0$, vamos a generar aproximaciones y_n a $y(x_0 + n\,h)$ para h fijo y n = 0, 1,2… ..., usando la fórmula

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2\,k_2 + 2\,k_3 + k_4) \qquad (15)$$

Donde

$$k_1 = f(x_n\,,\, y_n\,)$$

$$k_2 = h\,f(x_n + \frac{h}{2}\,,\, y_n + \frac{1}{2}\,k_1)$$

$$k_3 = h\,f(x_n + \frac{h}{2}\,,\, y_n + \frac{1}{2}\,k_2)$$

$$k_4 = h\,f(x_n + h\,,\, y_n + k_3)$$

El error local de truncamiento en este algoritmo es $O(h^5)$. Esta fórmula (15) es muy usada en la práctica.

Ejercicios resueltos

1. Resolver por el método de Euler el problema de valores iniciales $y' = y\,, y(0) = 1$

Vamos a utilizar la formula

$$y_{n+1}=y_n + h\,f(x_n, y_n\,) \qquad , E = \frac{h^2}{2!}\,y''(\alpha)$$

Con tamaño de paso h=0.01 y seis cifras decimales

$$y(0.01) \approx y_1 = 1 + 0.01 = 1.01$$

$$y(0.02) \approx y_2 = 1.01 + 0.01(1.01) = 1.0201$$

$$y(0.03) \approx y_3 = 1.0201 + 0.01(1.0201) = 1.030301$$

$$y(0.04) \approx y_4 = 1.030301 + 0.01(1.030301) = 1.040606$$

La solución exacta de la ecuación diferencial es $y = e^x$, por tanto el valor correcto en x =0.04 es 1.04081.

Para mayor precisión con el método de Euler, vamos a tomar un paso más pequeño h=0.005

$$y(0.05) \approx y_1 = 1.0050$$

$$y(0.010) \approx y_2 = 1.0100$$

$$y(0.015) \approx y_3 = 1.0151$$

$$y(0.020) \approx y_4 = 1.0202$$

$$y(0.025) \approx y_5 = 1.0253$$

$$y(0.030) \approx y_6 = 1.0304$$

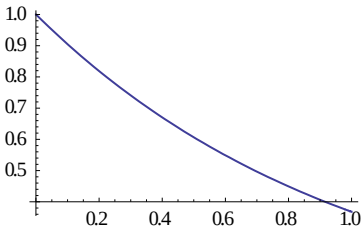
$$y(0.035) \approx y_7 = 1.0356$$

$$y(0.040) \approx y_8 = 1.0356$$

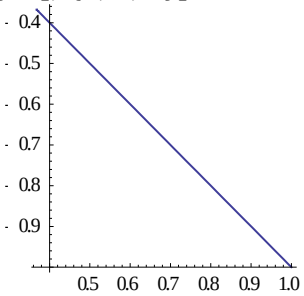
2. Resolver mediante el método de Euler el problema de valor inicial abajo indicado, tomando un paso h=0.1

$$y'[x] = -y[x] \ , y(0) = 1 \ , intervalo \ [0, 1]$$

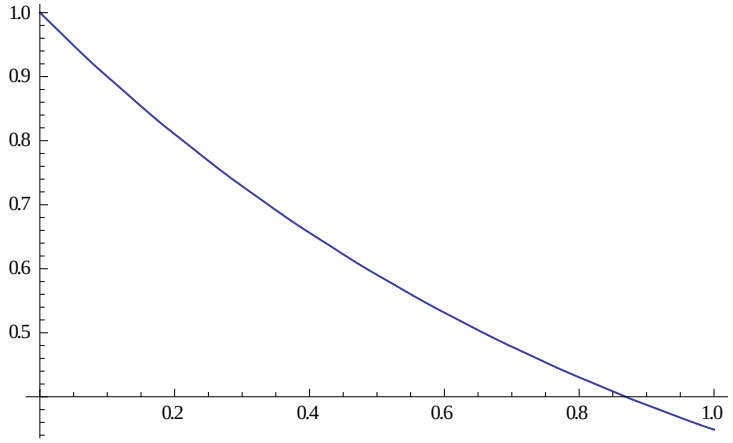
```
s=NDSolve[{y'[x]==- y[x] ,y[0]==1},y,{x,0,1}]
Plot [Evaluate[y[x]/.s], {x, 0, 1}]
```




```
ParametricPlot [Evaluate [{y[x], y'[x]}/.s], {x, 0, 1}]
```




```
{y [1],y'[1]}/.s
 solución =NDSolve [{y'[t]==-y[t], y[0]==1},y[t],{t,0,1},
 Method→"ExplicitEuler", "StartingStepSize"→0.1]
 Plot [Evaluate [y[t]/.solucion],{t,0,1}]
```

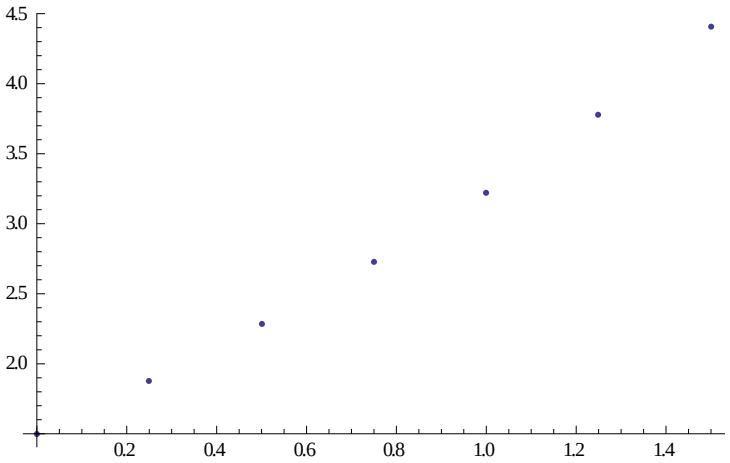



```
st=Table [Evaluate[solucion],{t,0,1,0.1}]
{{{y[0.]→1.}},{{y[0.1]→0.9}},{{y[0.2]→0.81}},{{y[0.3]→0.729}},{{y[0.4]→0.6561}},{{y[0.5]→0.
59049}},{{y[0.6]→0.531441}},{{y[0.7]→0.478297}},{{y[0.8]→0.430467}},{{y[0.9]→0.38742}},{
{y[1.]→0.348678}}}
```

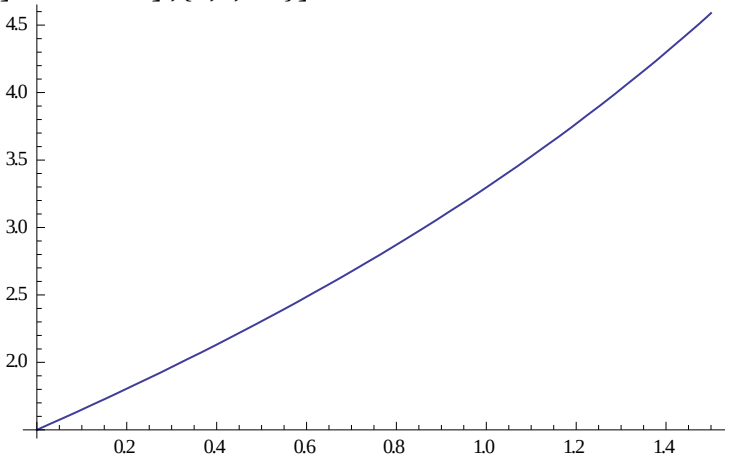
3. Resolver mediante el método de Euler el problema de valor inicial abajo indicado, tomando un paso de h=0.25

$$y' = y - t \ , y(0) = 1.5$$

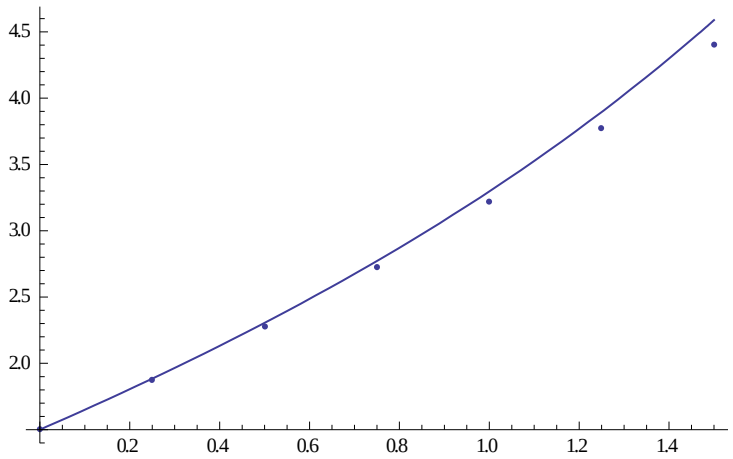
```
 euler[t0_,tn_,y0_,N_]:=Module[{i,h},h=(tn-t0)/N;T=Table[t0+(i-
1)*h,{i,1,N+1}];Y=Table[y0,{i,1,N+1}];
  For[i=1,i≤N,i++,
   Y[[i+1]]=Y[[i]]+h *f[T[[i]],Y[[i]]];];
  Return [Transpose[{T,Y}]];]

f[t_,y_]:=y-t;
 solucion=euler[0,1.5,1.5,6]
{{0.,1.5},{0.25,1.875},{0.5,2.28125},{0.75,2.72656},{1.,3.2207},{1.25,3.77588},{1.5,4.40735}}
a=ListPlot[solucion]
```




```
solucion1= NDSolve [{y'[x]== y[x]-x, y[0]==1.5},y[x],{x,0,1.5},
  Method→"ExplicitEuler", "StartingStepSize"→0.1]

{{y[x]→InterpolatingFunction[{{0.,1.5}},<>][x]}}
b=Plot [Evaluate [y[x]/.solucion1] ,{x,0,1.5}]
```



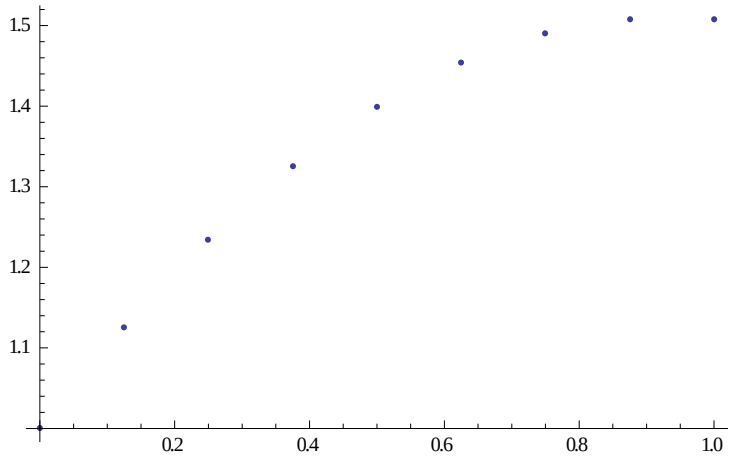

```
Show[ a,b ]
```



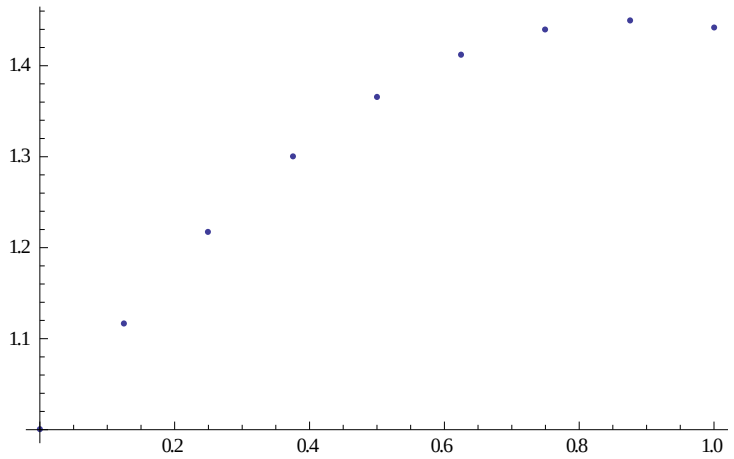
4. Resolver mediante el método de Taylor de orden dos el problema de valor inicial abajo indicado tomando un paso igual a 0.1 en el intervalo [0,1]

$$y' = 1 - t\sqrt[3]{y}\ , y(0) = 1$$

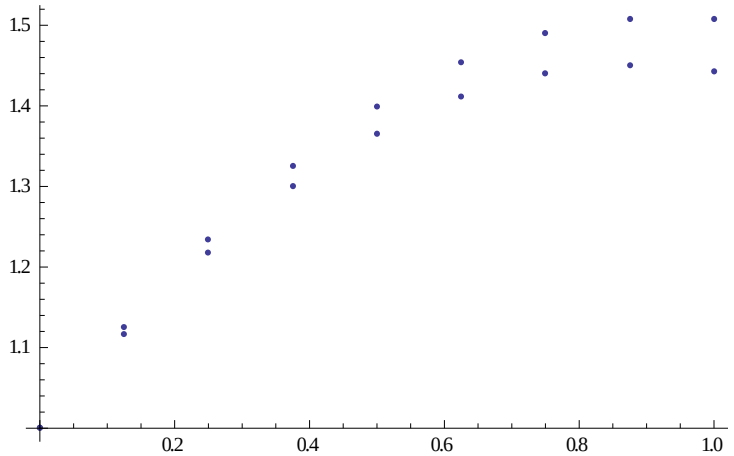
```
taylor2[t0_,tn_,y0_,N_]:= Module[{i,h},h=(tn-t0)/N;T=Table[t0+(i-
1)*h,{i,1,N+1}];Y=Table[y0,{i,1,N+1}];
  For[i=1,i≤N,i++,
   Y[[i+1]]=Y[[i]]+h *f[T[[i]],Y[[i]]]+ h^2/2* d2y[T[[i]],Y[[i]]]];
  Return[Transpose[{T,Y}]];]

f[t_,y_]:=1-t*y^(1/3);
d2y[t_,y_]:=-  y^(1/3)-(t/3)*y ^(-(2/3))* (1-t*y^(1/3));
 solucioneuler=N[euler[0 ,1 ,1 ,8] ]
{{0.,1.},{0.125,1.125},{0.25,1.23375},{0.375,1.32523},{0.5,1.39874},{0.625,1.45385},{0.75,1.49
034},{0.875,1.50826},{1.,1.50783}}
soluciontaylor2=N[taylor2[0 ,1,1,8]]
{{0.,1.},{0.125,1.11719},{0.25,1.2176},{0.375,1.30047},{0.5,1.3653},{0.625,1.41182},{0.75,1.44
003},{0.875,1.45011},{1.,1.44245}}
a=ListPlot[solucioneuler]
```




```
b=ListPlot[soluciontaylor2]
```

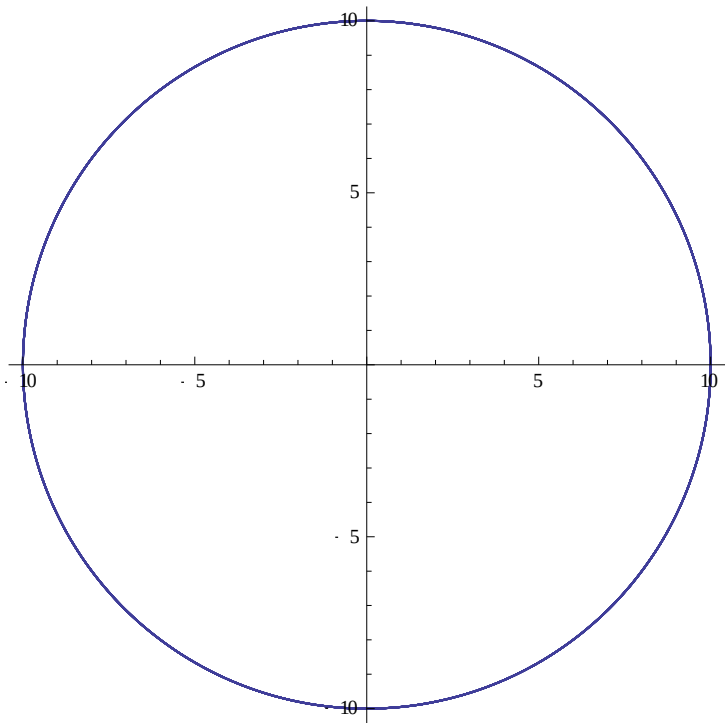


Show[a,b]

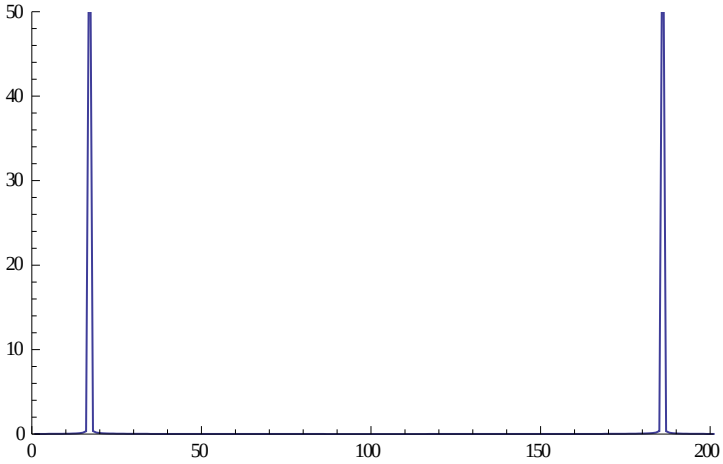


5. Calcular la solucion del oscilador armónico en el intervalo o [0,100] mediante el algoritmo de Runge-Kutta de 4 pasos utilizando h=0.5

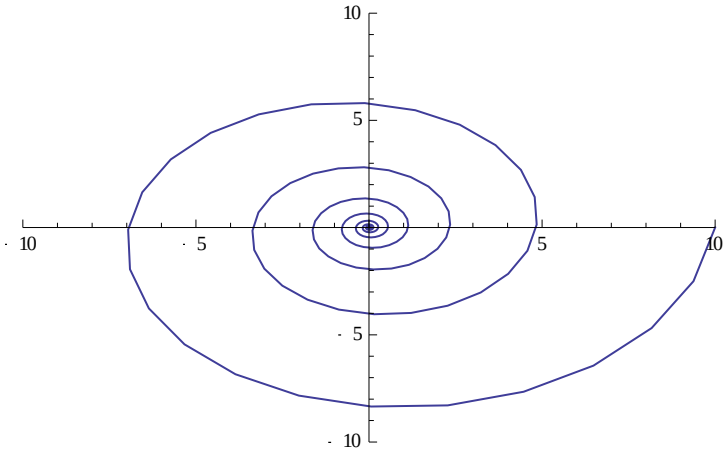
```
oscilador={y'[t] ==x[t]
   ,x'[t]+y[t]==0};
solucion=NDSolve[Join[oscilador,{x[0]==0 ,
                        y[0]==10 }],{x,y },{t,0,100},
                        MaxSteps→10000];

trayectorias={x[t],y[t]}/.solucion⟦1⟧] ;


g=ParametricPlot [Evaluate [trayectorias],{t,0,100}]
```




```
a=Table[trayectorias , {t,0,100,0.5}];
b=Fourier[a];
c=Map[First,b];
d=ListPlot[Abs[c],Joined→True,
                PlotRange→{{0,Length[a]},{0,50}}]
```

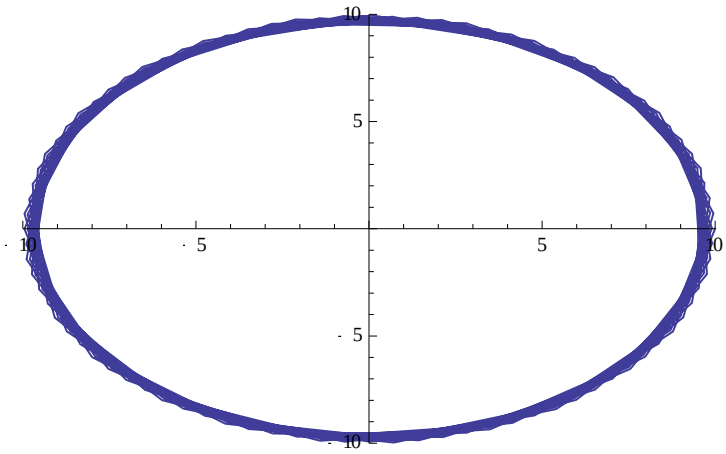



```
runge[t0_,tn_,y10_,y20_,N_]:=Module[{i,h},h=(tn-t0)/N;
  T=Table [t0+(i-1)*h ,{i,1,N+1}];
  Y1=Table [y10,{i,1,N+1}];
  Y2=Table [y20,{i,1,N+1}];
  For[i=1,i≤N+1,i++,
   k11=Y1[[i]]+h/2*f1[T[[i]],Y1[[i]],Y2[[i]]];
   k12=Y2[[i]]+h/2*f2[T[[i]],Y1[[i]],Y2[[i]]];
   Y1[[i+1]]=Y1[[i]]+h/2*f1[T[[i]]+h/2,k11,k12];
   Y2[[i+1]]=Y2[[i]]+ h/2*f2[T[[i]]+h/2,k11,k12];];
  Return[Transpose[{T,Y1,Y2}]];]
f1[t_,y1_,y2_]=y2;
f2[t_,y1_,y2_]=-y1;
solucion=runge[0,100,10,0,200];
data1=Table[{solucion[[i,2]],solucion[[i,3]]},{i,1,200}];
ListPlot[data1,Joined→True ,PlotRange→{{-10,10},{-10,10}}]
```



Observamos que en estas condiciones el algoritmo de Runge introduce una difusión numérica en este problema y la difusión numérica se amortigua , lo cual no ocurre en la solución analítica .Por tanto en estas condiciones es necesario utilizar un método numérico de mayor orden para resolver este problema.

```
   Runge- kutta4
 rk4 [t0_, tn_,y10_,y20_,N_]:=Module[{i,h},h=(tn-t0)/N;
  T=Table [t0+(i-1)*h ,{i,1,N+1}];
  Y1=Table [y10,{i,1,N+1}];
  Y2=Table [y20,{i,1,N+1}];
  For [i=1,i≤N+1,i++,
   k11=h*f1[T[[i]],Y1[[i]],Y2[[i]]];
   k12=h*f2[T[[i]],Y1[[i]],Y2[[i]]];
   k21=h*f1[T[[i]]+h/2,Y1[[i]]+k11/2,Y2[[i]]+k12/2];
   k22=h*f2[T[[i]]+h/2,Y1[[i]]+k11/2,Y2[[i]]+k12/2];
   k31=h*f1[T[[i]]+h/2,Y1[[i]]+k21/2,Y2[[i]]+k22/2];
   k32=h*f2[T[[i]]+h/2,Y1[[i]]+k21/2,Y2[[i]]+k22/2];

   k41=h*f1[T[[i]]+h,Y1[[i]]+k31,Y2[[i]]+k32];

   k42=h*f2[T[[i]]+h,Y1[[i]]+k31,Y2[[i]]+k32];
   Y1[[i+1]]=Y1[[i]]+1/6*(k11+2*k21+2*k31+k41);
   Y2[[i+1]]=Y2[[i]]+1/6*(k12+2*k22+2*k32+k42);];
  Return [Transpose[{T,Y1,Y2}]];]
f1 [t_,y1_,y2_]=y2;
f2 [t_,y1_,y2_]=-y1;
solucion=rk4[0,100,10,0,200];
data1=Table[{solucion[[i,2]],solucion[[i,3]]},{i,1,200}];
ListPlot[data1,Joined→True ,PlotRange→{{-10,10},{-10,10}}]
```


10
5
-10
-5
5
10
-5
-10