MATLAB

Procedimientos numéricos

Solución numérica de ecuaciones diferenciales (I)

Las leyes que gobiernan los fenómenos de la naturaleza se expresan habitualmente en forma de ecuaciones diferenciales. Las ecuaciones del movimiento de los cuerpos (la segunda ley de Newton) es una ecuación diferencial de segundo orden, como lo es la ecuación que describe los sistemas oscilantes, la propagación de las ondas, la transmisión del calor, la difusión, el movimiento de partículas subatómicas, etc.

Pocas ecuaciones diferenciales tienen una solución analítica sencilla, la mayor parte de las veces es necesario realizar aproximaciones, estudiar el comportamiento del sistema bajo ciertas condiciones. Así, en un sistema tan simple como un péndulo, la amplitud de la oscilación ha de ser pequeña y el rozamiento ha de ser despreciable, para obtener una solución sencilla que describa aproximadamente su movimiento periódico.

Se estudia el procedimiento de Runge-Kutta que se aplica de forma directa a una ecuación diferencial de primer orden, pero veremos como se extiende a un sistema de ecuaciones de primer orden, a un ecuación diferencial de segundo orden y a un sistema de ecuaciones diferenciales de segundo orden.

El procedimiento de Runge-Kutta se puede programar fácilmente en los ordenadores y además, se emplea mucho en la práctica, debido a la su exactitud relativamente elevada de la solución aproximada de la ecuación diferencial. La justificación del procedimiento de Runge-Kutta no es sencilla, el lector interesado puede consultar algún libro de métodos numéricos de análisis.

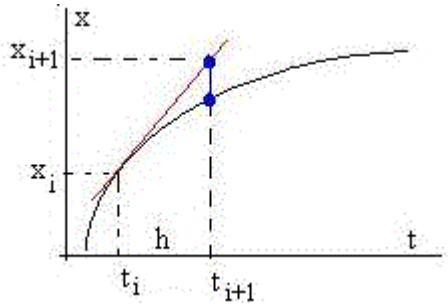
Método de Euler

Vamos aresolver la ecuación diferencial de primer orden

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

con con la condición inicial de que en el instante t_0 la posición es x_0

La primera derivada nos permite conocer la posición x_{i+1} en el instante t_{i+1}, a partir de la posición x_i en el instante t_i de acuerdo a la fórmula siguiente. La línea de color rojo es la tangente a la curva en el instante t_i



$x_{i+1}=x_i+f(t_i,x_i)h$

El procedimiento de Euler produce un error que se acumula a cada paso *h* de integración, que es el segmento en color azul que une los dos puntos en la figura.

Escribimos una función denominada *euler*, a la que le pasaremos:

- la función *f*(*t*,*x*),
- la condición inicial de que en el instante t_0 la posición es x_0,
- el instante final t_f
- el número de pasos de integración *n*

y nos devolverá un vector *t* y su correspondiente vector *x*.

```
function [t,x] =euler(f,t0,tf,x0,n)
      h=(tf-t0)/n;
      t=t0:h:tf;
      x=zeros(n+1,1); %reserva memoria para n+1 elementos del vector x
      x(1)=x0;
      for i=1:n
          x(i+1)=x(i)+f(t(i),x(i))*h;
      end
end
```

Supongamos que queremos integrar la ecuación diferencial

$$\frac{dx}{dt}=\cos t$$

con las condición inicial *t*=0, *x*=0.

$$x-0=\int_0^t \cos t\cdot dt$$

$$x=\sin t$$

Tomamos un intervalo *h*=π/6, y construimos la siguiente tabla

t	$\frac{dx}{dt}=\cos t$	***x*(Euler)**	***x*=sin *t***
0	1	0	0
π/6	0.866	0.523	0.5
π/3	0.5	0.977	0.866
π/2	0	1.239	1
2π/3	-0.5	1.239	0.866
5π/6	-0.866	0.977	0.5
π		0.523	0

Para aplicar el método de Euler precisamos de un paso *h* pequeño, incluso así los errores se van acumulando y al cabo de cierto tiempo la diferencia entre el valor exacto y el calculado es grande.

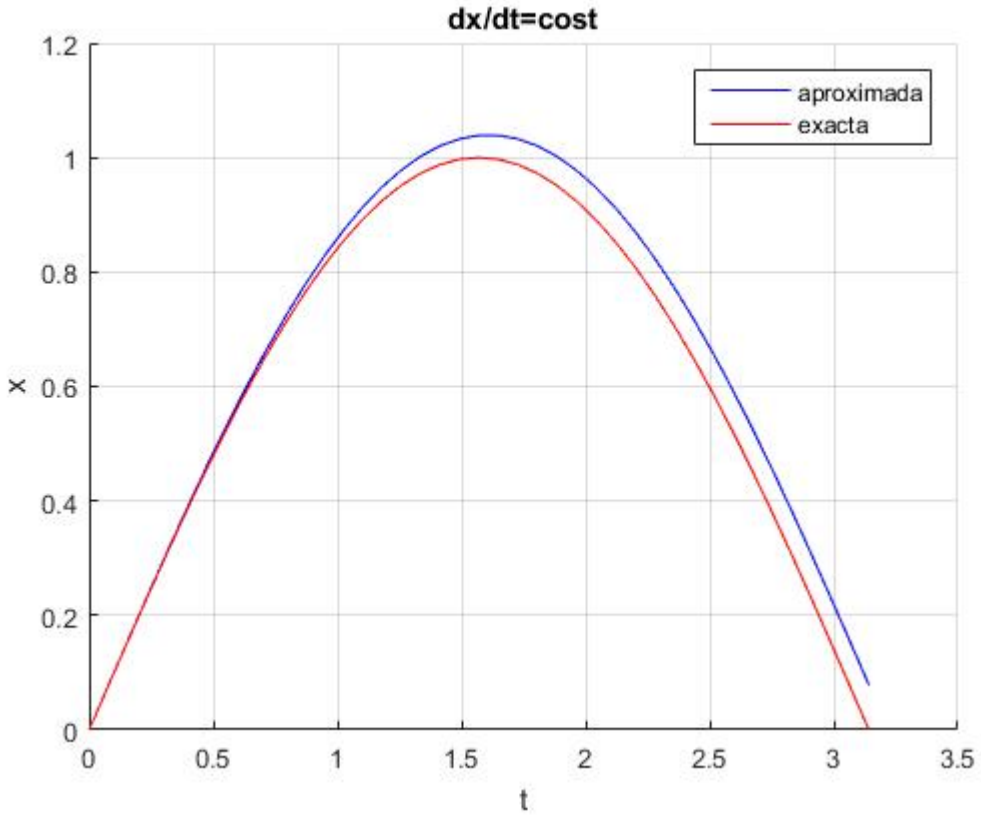
Escribimos un script en el que definiremos la función $f(t,x)$, las condiciones iniciales y llamaremos a la función *euler*. Finalmente, representaremos gráficamente la solución exacta y la obtenida aplicando el método de Euler

```
tf=input('tiempo final, tf: ');
n=input('número de pasos, n: ');
f=@(t,x) cos(t);
%condiciones iniciales
t0=0;
x0=0;
[t,x]=euler(f,t0,tf,x0,n);

hold on
plot(t,x,'b')
y=sin(t);
plot(t,y,'r')
xlabel('t')
ylabel('x')
grid on
legend('aproximada','exacta')
title('dx/dt=cost')
hold off
```

En la ventana de comandos corremos el script

```
tiempo final, tf: pi
número de pasos, n: 40
```



Apreciamos diferencias entre la solución exacta y la obtenida mediante integración numérica por el método de Euler

Método de Runge-Kutta

En esta sección vamos a estudiar la aplicación del método de Runge-Kutta a:

- Una ecuación diferencial de primer orden
- Un sistema de dos ecuaciones diferenciales de primer orden
- Una ecuación difrencial de segundo orden
- Un sistema de dos ecuaciones diferenciales de segundo orden

Ecuación diferencial de primer orden

Sea una ecuación diferencial de primer orden, con la condición inicial de que en el instante t_0 el valor inicial de x es x_0

Se elige una anchura de paso h y se calculan cuatro números k_1, k_2, k_3, k_4 de acuerdo con el procedimiento esquematizado en la tabla adjunta. Según el procedimiento ordinario de Runge-Kutta, a partir del valor de x en el instante t se determina el valor de x en el instante $t+h$ mediante la fórmula que figura en la última fila de dicha tabla.

$\frac{dx}{dt}=f(t,x)$
$k_1=h\cdot f(t,x)$ $k_2=h\cdot f\left(t+\frac{1}{2}h,x+\frac{1}{2}k_1\right)$ $k_3=h\cdot f\left(t+\frac{1}{2}h,x+\frac{1}{2}k_2\right)$ $k_4=h\cdot f\left(t+h,x+k_3\right)$
$x(t+h)=x(t)+\frac{1}{6}(k_1+2k_2+2k_3+k_4)$

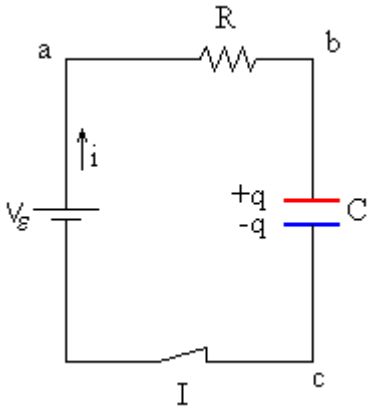
Definimos la función *rk_1* que resuelve la ecuación diferencial de primer orden, cuando le pasamos:

- la función $f(t,x)$,
- la condición inicial de que en el instante $t_0$el valor inicial es x_0,
- el instante final t_f
- el número de pasos de integración n comprendidos entre el instante inical t_0 y final t_f.

y nos devolverá un vector t y su correspondiente vector x.

```
function [t,x] =rk_1(f,t0,tf,x0,n)
    h=(tf-t0)/n;
    t=t0:h:tf;
    x=zeros(n+1,1); %reserva memoria para n elementos del vector x
    x(1)=x0;
    for i=1:n
        k1=h*f(t(i),x(i));
        k2=h*f(t(i)+h/2,x(i)+k1/2);
        k3=h*f(t(i)+h/2,x(i)+k2/2);
        k4=h*f(t(i)+h,x(i)+k3);
        x(i+1)=x(i)+(k1+2*k2+2*k3+k4)/6;
     end
end
```

Considérese el circuito en serie de la figura. Inicialmente el condensador está descargado. Si se cierra el interruptor I la carga empieza a fluir produciendo corriente en el circuito, el condensador se empieza a cargar. Una vez que el condensador adquiere la carga máxima, la corriente cesa en el circuito.



$$R\frac{dq}{dt}=V_\varepsilon-\frac{q}{C}$$

$$\int_0^q \frac{dq}{CV_\varepsilon-q}=\frac{1}{RC}\int_0^t dt \qquad q=CV_\varepsilon\left(1-\exp\left(\frac{-t}{RC}\right)\right)$$

Escribimos un script para que realice las siguientes tareas:

1. Establezca, mediante comandos *input*:
   - La resistencia *R* del circuito
   - La capacidad *C* del condensador
   - El tiempo final, *tf*
   - el número de pasos, *n*.
2. Fije las condiciones iniciales, en el instante inicial *t*=0, el condensador está descargado *x*=0.
3. Defina la función *f*(*t*,*x*),
4. Llame al procedimiento numérico *rk*_1
5. Mediante el comando *plot* realice una representación gráfica de la solución numérica
6. Realice una representación gráfica de la solución exacta

Ejemplo: *R*=2.0, *C*=0.8, y *tf*=10.

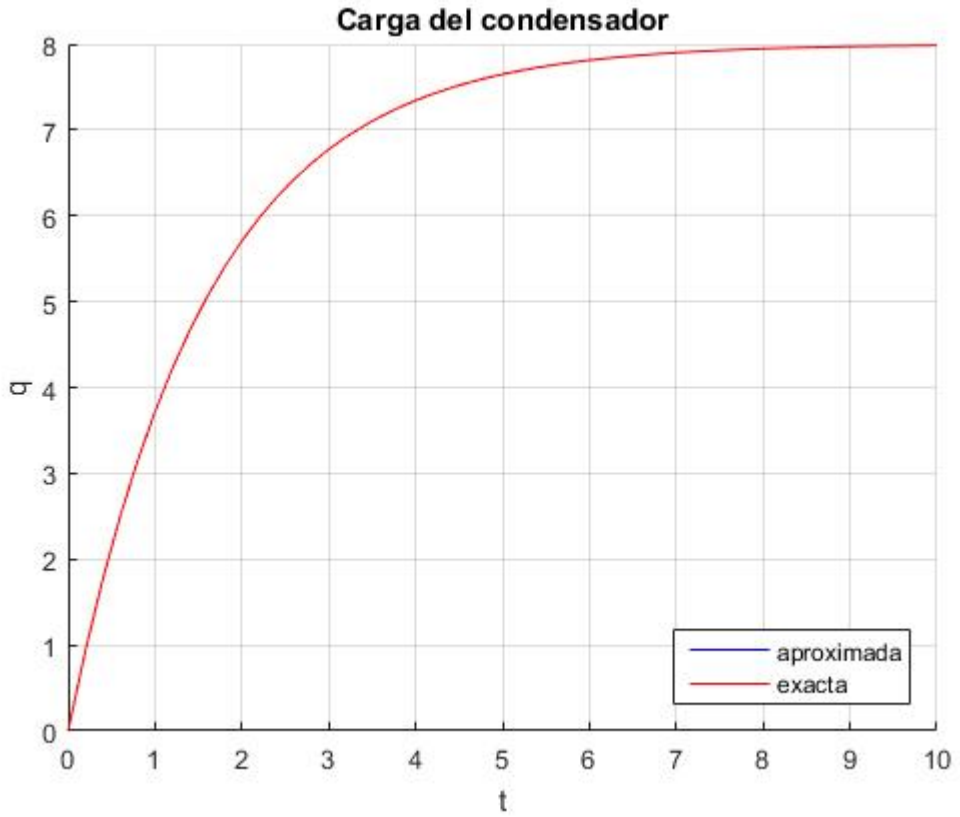
```
V0=10;
R=input('Resistencia R: ');
C=input('Capacidad C: ');
tf=input('tiempo final, tf: ');
n=input('número de pasos, n: ');

f=@(t,x) V0/R-x/(R*C);
%condiciones iniciales
t0=0; x0=0;
[t,x]=rk_1(f,t0,tf,x0,n);

hold on
plot(t,x,'b')
y=C*V0*(1-exp(-t/(R*C)));
plot(t,y,'r')
xlabel('t')
ylabel('q')
grid on
legend('aproximada','exacta','Location','Southeast')
title('Carga del condensador')
hold off
```

En la ventana de comandos corremos el script

```
Resistencia R: 2
Capacidad C: 0.8
tiempo final, tf: 10
número de pasos, n: 50
```



No se aprecia diferencia entre la solución exacta y la numérica, aplicando el procedimiento de Runge_Kutta.

Sistema de dos ecuaciones diferenciales de primer orden

El procedimiento de Runge-Kutta es igualmente efectivo en la resolución de un sistema de dos ecuaciones diferenciales de primer orden.

$$\frac{dx}{dt}=f(t,x,y) \qquad \frac{dy}{dt}=g(t,x,y)$$

El procedimiento de aplicación del método de Runge-Kutta a cada una de las ecuaciones diferenciales, con las condición inicial siguiente, en el instante t_0

- el valor inicial de x es x_0
- el valor inicial de y es y_0

se esquematiza en la tabla adjunta. Como vemos además de los cuatro números k_1, k_2, k_3, k_4 para la primera ecuación diferencial precisamos otros cuatro números l_1, l_2, l_3, l_4 para la segunda ecuación diferencial. A partir del valor de x en el instante t, se determina el valor de x en el instante $t+h$, y a partir del valor de y en el instante t se determina el valor de y en el instante $t+h$ mediante las fórmulas de la última fila de la tabla.

$\frac{dx}{dt}=f(t,x,y)$	$\frac{dy}{dt}=g(t,x,y)$
$k_1=h\cdot f(t,x,y)$	$l_1=h\cdot g(t,x,y)$
$k_2=h\cdot f\left(t+\frac{1}{2}h,x+\frac{1}{2}k_1,y+\frac{1}{2}l_1,\right)$	$l_2=h\cdot g\left(t+\frac{1}{2}h,x+\frac{1}{2}k_1,y+\frac{1}{2}l_1\right)$
$k_3=h\cdot f\left(t+\frac{1}{2}h,x+\frac{1}{2}k_2,y+\frac{1}{2}l_2\right)$	$l_3=h\cdot g\left(t+\frac{1}{2}h,x+\frac{1}{2}k_2,y+\frac{1}{2}l_2\right)$
$k_4=h\cdot f\left(t+h,x+k_3,y+l_3\right)$	$l_4=h\cdot g\left(t+h,x+k_3,y+l_3\right)$

$x(t+h)=x(t)+\frac{1}{6}(k_1+2k_2+2k_3+k_4)$	$y(t+h)=y(t)+\frac{1}{6}(l_1+2l_2+2l_3+l_4)$

Definimos la función *rk_2_1* que resuelve el sistema de dos ecuaciones diferenciales de primer orden, cuando le pasamos:

- las funciones *f* (*t*,*x*,*y*) y *g*(*t*,*x*,*y*)
- las condiciones iniciales (x_0,y_0) en el instante t_0
- el número *n* de pasos de integración entre t_0 y el tiempo final t_f

Nos devuelve los vectores *x* e *y* para cada instante que se guarda en el vector *t* comprendido entre el instante inicial t_0 y el final t_f.

```
function [t,x,y] =rk_2_1(f,g,t0,tf,x0,y0,n)
      h=(tf-t0)/n;
      t=t0:h:tf;
%reserva memoria para n+1 element(i)os del vect(i)or x(i)
      x=zeros(n+1,1);
      y=zeros(n+1,1);
      x(1)=x0; y(1)=y0;

      for i=1:n
        k1=h*f(t(i),x(i),y(i));
        l1=h*g(t(i),x(i),y(i));
        k2=h*f(t(i)+h/2,x(i)+k1/2,y(i)+l1/2);
        l2=h*g(t(i)+h/2,x(i)+k1/2,y(i)+l1/2);
        k3=h*f(t(i)+h/2,x(i)+k2/2,y(i)+l2/2);
        l3=h*g(t(i)+h/2,x(i)+k2/2,y(i)+l2/2);
        k4=h*f(t(i)+h,x(i)+k3,y(i)+l3);
        l4=h*g(t(i)+h,x(i)+k3,y(i)+l3);

        x(i+1)=x(i)+(k1+2*k2+2*k3+k4)/6;
        y(i+1)=y(i)+(l1+2*l2+2*l3+l4)/6;
      end
end
```

Consideremos una serie radioactiva de tres elementos A-->B-->C en la que, una sustancia radiactiva A se desintegra y se transforma en otra sustancia radiactiva B, que a su vez se desintegra y se transforma en una sustancia C estable. Las ecuaciones diferenciales que gobiernan el proceso y sus soluciones analíticas son, respectivamente,

$$\frac{dx}{dt}=-ax \qquad x=x_0\exp(-at)$$
$$\frac{dy}{dt}=ax-by \qquad y=\frac{a}{b-a}x_0\left(\exp(-at)-\exp(-bt)\right)$$

La solución analítica que aparece a la derecha, se ha obtenido con las condiciones iniciales *t*=0, *x*=x_0 e *y*=0. La segunda solución se obtiene siempre que *a* sea distinto de *b.* En el caso de que *a* sea igual a *b*, la solución analítica para *y* es

$$y=x_0a\exp(-at)$$

La interpretación del sistema de ecuaciones diferenciales no es complicada. En la unidad de tiempo, desaparecen *ax* núcleos de la sustancia A al desintegrarse (primera ecuación). En la unidad de tiempo, se producen *ax* núcleos de la sustancia B y a su vez desaparecen *bx* núcleos de la sustancia B, que al desintegrarse se transforman en núcleos de la sustancia C estable (segunda ecuación).

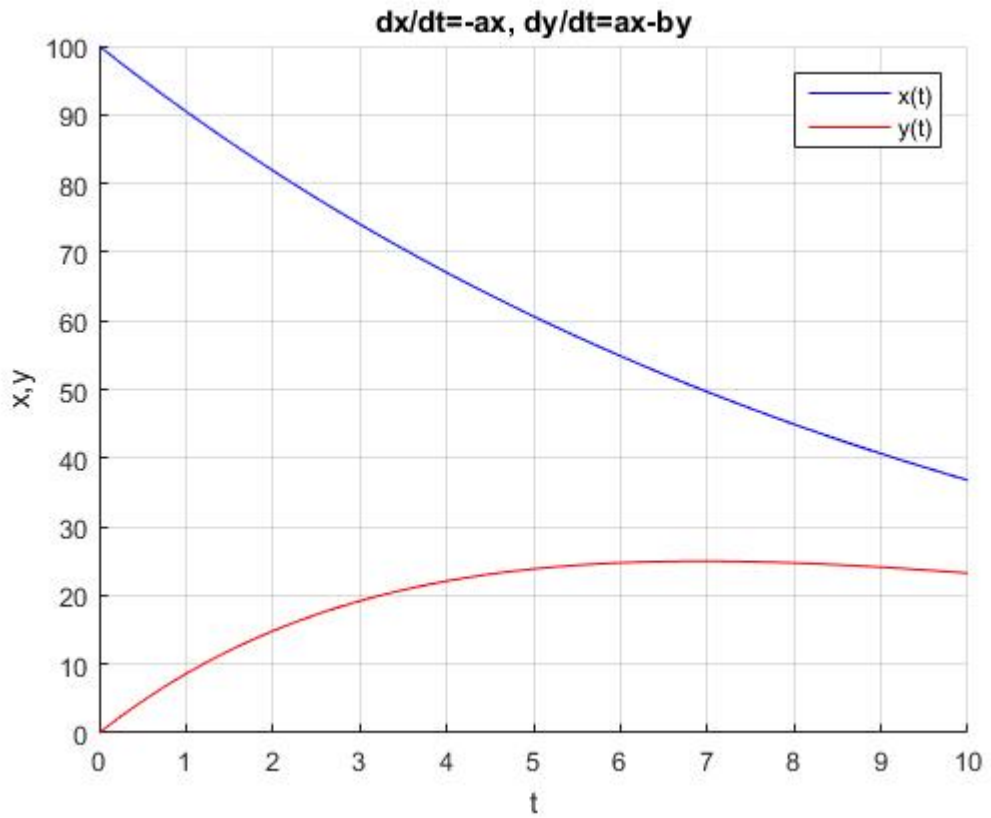
Escribimos un script en el que definiremos las funciones *f*(*t*,*x*,y), *g*(*t*,*x*,y), las condiciones iniciales y llamaremos a la función *rk_2_1*

```
a=input('parámetro a: ');
b=input('parámetro b: ');
x0=input('valor inicial de x: ');
y0=input('valor inicial de y: ');
tf=input('tiempo final, tf: ');
n=input('número de pasos, n: ');
f=@(t,x,y) -a*x;
g=@(t,x,y) a*x-b*y;
%condiciones iniciales
t0=0;

[t,x,y]=rk_2_1(f,g,t0,tf,x0,y0,n);
hold on
plot(t,x,'b')
plot(t,y,'r')
xlabel('t')
ylabel('x,y')
grid on
legend('x(t)','y(t)')
title('dx/dt=-ax, dy/dt=ax-by')
hold off
```

En la ventana de comandos corremos el script

```
parámetro a: 0.1
parámetro b: .2
valor inicial de x: 100
valor inicial de y: 0
tiempo final, tf: 10
número de pasos, n: 40
```



Ecuación diferencial de segundo orden

Existen muchas situaciones en las que es necesario resolver una ecuación diferencial de segundo orden.

$$\frac{d^2x}{dt^2}=f(t,x,v)$$

con las condiciones iniciales

$$x(t_0)=x_0 \qquad \left(\frac{dx}{dt}\right)_{t_0}=v_0$$

Una ecuación diferencial de segundo orden es equivalente a un sistema de dos ecuaciones diferenciales de primer orden, por lo que aplicaremos el mismo esquema.

$\frac{dx}{dt}=v$	$\frac{dv}{dt}=f(t,x,v)$
$k_1=hv$ $k_2=h\left(v+\frac{1}{2}l_1\right)$ $k_3=h\left(v+\frac{1}{2}l_2\right)$ $k_4=h\left(v+l_3\right)$	$l_1=h\cdot f(t,x,v)$ $l_2=h\cdot f\left(t+\frac{1}{2}h,x+\frac{1}{2}k_1,v+\frac{1}{2}l_1\right)$ $l_3=h\cdot f\left(t+\frac{1}{2}h,x+\frac{1}{2}k_2,v+\frac{1}{2}l_2\right)$ $l_4=h\cdot f\left(t+h,x+k_3,v+l_3\right)$
$x(t+h)=x(t)+\frac{1}{6}(k_1+2k_2+2k_3+k_4)$	$v(t+h)=v(t)+\frac{1}{6}(l_1+2l_2+2l_3+l_4)$

Definimos la función *rk_2* que resuelve la ecuación diferencial de segundo orden, cuando le pasamos:

- la función *f* (*t,x,v*)
- las condiciones iniciales: posición inicial x_0 y velocidad inicial v_0 en el instante t_0
- el número *n* de pasos de integración entre t_0 y el tiempo final t_f

Nos devuelve los vectores de las posiciones *x* y las velocidades *v* para cada instante que se guarda en el vector *t* comprendido entre el instante inicial t_0 y el final t_f.

```
function [t,x,v] =rk_2(f,t0,tf,x0,v0,n)
      h=(tf-t0)/n;
      t=t0:h:tf;
%reserva memoria para n+1 element(i)os del vect(i)or x(i)
      x=zeros(n+1,1);
      v=zeros(n+1,1);
      x(1)=x0; v(1)=v0;

      for i=1:n
        k1=h*v(i);
        l1=h*f(t(i),x(i),v(i));
        k2=h*(v(i)+l1/2);
        l2=h*f(t(i)+h/2,x(i)+k1/2,v(i)+l1/2);
        k3=h*(v(i)+l2/2);
        l3=h*f(t(i)+h/2,x(i)+k2/2,v(i)+l2/2);
        k4=h*(v(i)+l3);
        l4=h*f(t(i)+h,x(i)+k3,v(i)+l3);

        x(i+1)=x(i)+(k1+2*k2+2*k3+k4)/6;
        v(i+1)=v(i)+(l1+2*l2+2*l3+l4)/6;
      end
end
```

La ecuación diferencial que describe un oscilador armónico amortiguado y su solución para unas condiciones iniciales fijadas es

$$\frac{d^2x}{dt^2}+2\gamma\frac{dx}{dt}+\omega_0^2x=0$$

$$\omega=\sqrt{\omega_0^2-\gamma^2}$$

$$x=\exp(-\gamma t)\left(A\sin(\omega t)+B\cos(\omega t)\right)$$

$$v=\frac{dx}{dt}=-\gamma\exp(-\gamma t)\left(A\sin(\omega t)+B\cos(\omega t)\right)+\omega\exp(-\gamma t)\left(A\cos(\omega t)-B\sin(\omega t)\right)$$

Condiciones iniciales: en el instante t=0, la posición inicial es x_0 y la velocidad inicial v_0.

$$t=0\quad\begin{cases}x_0=B\\ v_0=-\gamma B+\omega A\end{cases}$$

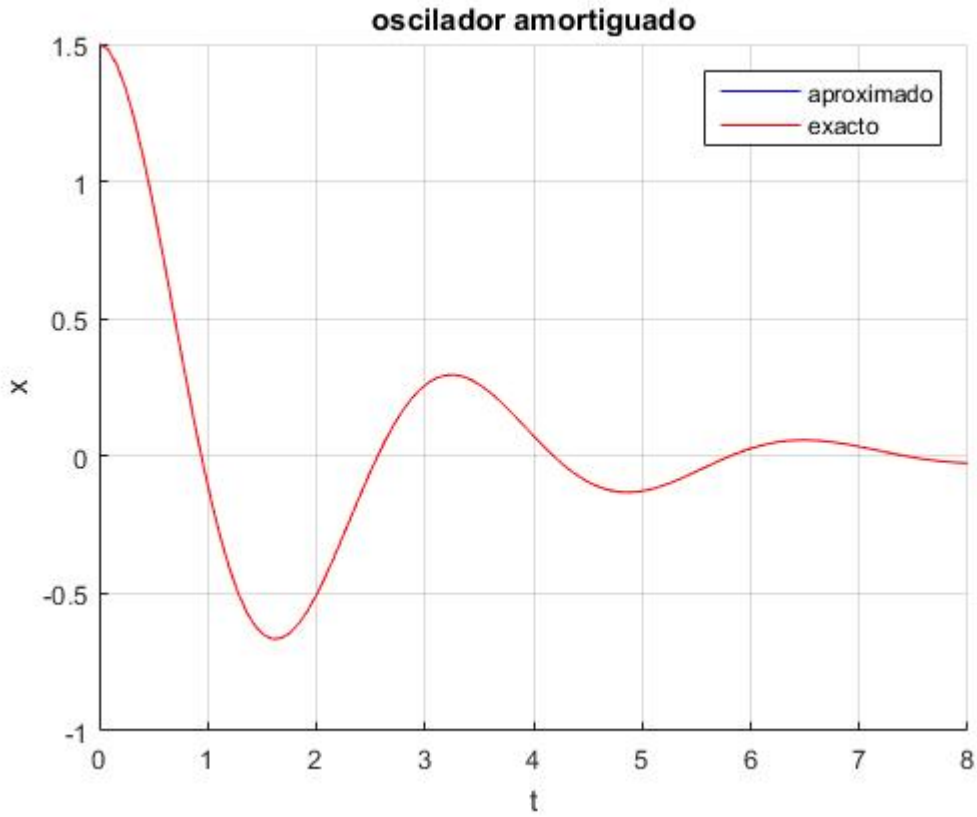
$$x=\exp(-\gamma t)\left(\frac{v_0+\gamma x_0}{\omega}\sin(\omega t)+x_0\cos(\omega t)\right)$$

Escribimos un script en el que definiremos la función *f*(*t*,*x*,*v*), las condiciones iniciales y llamaremos a la función *rk*_2

```
w0=input('frecuencia angular w0: ');
g=input('rozamiento, gamma: ');
x0=input('posición inicial, x0: ');
v0=input('velocidad inicial,v0: ');
tf=input('tiempo final, tf: ');
n=input('número de pasos, n: ');
f=@(t,x,v) -2*g*v-w0*w0*x;
%condiciones iniciales
t0=0;
hold on
%solución numérica
[t,x,v]=rk_2(f,t0,tf,x0,v0,n);
plot(t,x,'b')
%solución analítica
w=sqrt(w0*w0-g*g);
x=((v0+g*x0)*sin(w*t)/w+x0*cos(w*t)).*exp(-g*t);
plot(t,x,'r')
grid on
xlabel('t')
ylabel('x');
legend('aproximado','exacto')
title('oscilador amortiguado')
hold off
```

En la ventana de comandos corremos el script *oscilador* con distintas condiciones iniciales

```
>> oscilador
frecuencia angular, w0: 2
rozamiento, gamma: 0.5
posición inicial, x0: 1.5
velocidad inicial, v0: 0
tiempo final, tf: 8
número de pasos, n: 100
```



No se aprecia tampoco diferencia entre la solución exacta y la numérica, aplicando el procedimiento de Runge_Kutta.

Grado en Ingeniería de Energías Renovables