METODOS NUMÉRICOS: EULER, EULER MODIFICADO Y RUNGE KUTTA con MATLAB

Recopilación por: Ing. J. Orbegoso L. UNJFSC

Los presentes ejercicios constituyen unos pequeños y sencillos ejemplos de cómo usar un poco de MatLab. Para ello se ha elegido las *EDO's*, y dentro de ellas los sencillos *PVI's*. Los métodos matemáticos para la resolución numérica de los problemas de valores iniciales son el *método de Euler, el* de *Euler mejorado* y el de *Runge-Kutta.*

Recordemos que el algoritmo resultante de aplicar el método de Euler es

$$x_{i+1} = x_i + hf(t_i, x_i) x_0 = \xi_0$$

donde h es el paso que podemos elegir y ξ_0 es la aproximación numérica del valor $x(0)=x_0$ (pensad por ejemplo en $x_0=2\sqrt{}$, un programa de cálculo numérico -sí uno de cálculo simbólico al estilo *Maple* ó *Máxima*- no puede trabajar con el valor exacto de $2\sqrt{}$ por lo que necesita aproximarlo). Empecemos con un ejemplo muy sencillo, una ecuación de primer orden

$$\{x'=\sin(t)x(0)=1 cont\in[0,2\pi]$$

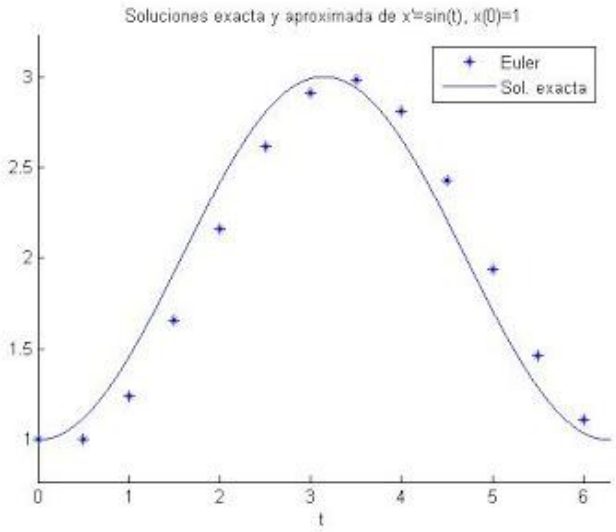
Aplicando el algoritmo nos queda el siguiente problema numérico recurrente

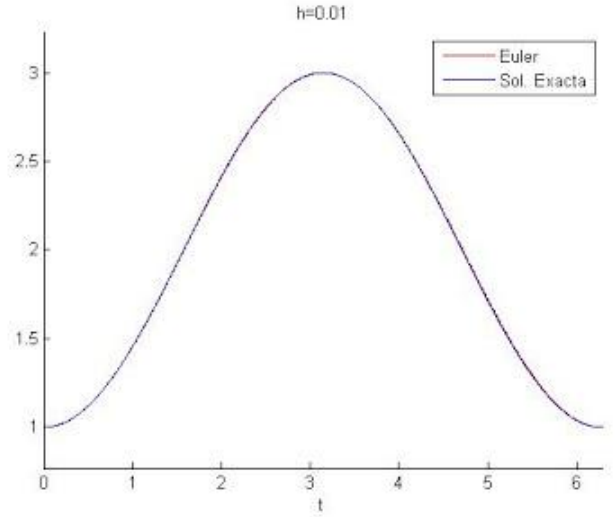
$$\{x_{i+1}=x_i+h\sin(t_i)x_0=1$$

que es muy sencillo de programar en MatLab, quedándonos

```
clear all
%Datos de entrada
f=@(t)[sin(t)];
a=0;
b=2*pi;
h=0.5;
N=(b-a)/h;
t(1)=a;
x(1)=1;
%Programa Principal
for k=1:N
t(k+1)=t(k)+h;
x(k+1)=x(k)+f(t(k))*h;
end
%Salida
hold on
plot(t,x,'*') % solución aproximada
z=dsolve('Dz=sin(t)','z(0)=1'); %solución exacta en color azul
ezplot(z,[a,b])
hold off
```

y MatLab nos devuelve las gráficas siguientes; la primera es la que realmente devuelve, la segunda es la que se obtiene variando un poco el código de arriba: refinamos el paso hasta $h=0.01$ y dibujamos ambas gráficas sin estrellas.





Como puede verse, incluso en un problema tan sencillo, el método de Euler necesita de un paso bastante pequeño para proporcionar una solución que se ajuste a la real, por eso existen modificaciones del mismo y otros métodos más eficientes: *método de Euler mejorado, método de Runge-Kutta,…* Antes de pasar a escribir el código para ellos vamos con otro ejemplo, ahora de un pvi de segundo orden. Consideramos el caso del oscilador armónico que modeliza la oscilación de un muelle ideal

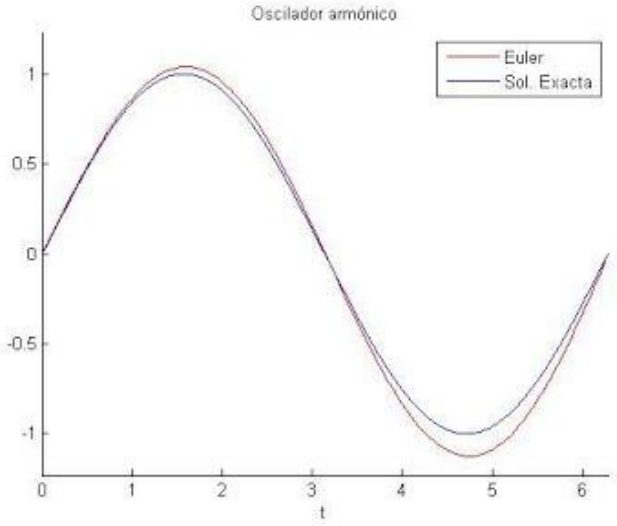
$$\begin{cases} x''+x=0 \\ x(0)=0 \\ x'(0)=1 \end{cases} cont\in[0,2\pi]$$

¿Cómo aplicamos el método de Euler aquí? muy sencillo, transformamos la ecuación de segundo orden en un sistema de dos ecuaciones de primer orden introduciendo una nueva variable, $y:=x'$. La idea es aplicar la aproximación del método de Euler a cada ecuación por separado, pero resolviéndolas al mismo tiempo pues cada una necesita de la otra para resolverse. El código no diferirá demasiado del anterior

```
 clear all
%datos
a=0;
b=2*pi;
h=0.05;
N=(b-a)/h;
t(1)=a;
x(1)=0;
y(1)=1;
%programa
for k=1:N
t(k+1)=t(k)+h;
```

```
x(k+1)=x(k)+h*y(k);
y(k+1)=y(k)-h*x(k);
end
%salida
hold on
plot(t,x,'r')
z=dsolve('D2x+x=0','x(0)=0','Dx(0)=1')
ezplot(z,[0,2*pi])
hold off
```

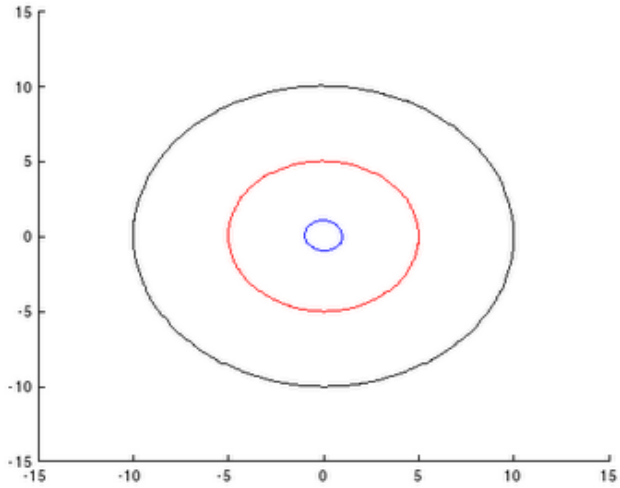
y lo que MatLab nos devuelve es la figura



incluso podemos dibujar el diagrama de fases del oscilador armónico de tal manera que contenga en una misma figura las órbitas que pasan por lo puntos, por ejemplo, (0,1),(0,5),(0,10). Elijo para ello un paso de h=0.0001, pues sino las órbitas no quedan bien dibujadas (si tomáis por ejemplo un paso de 0.1 éstas aparecerán como curvas abiertas, y todos sabemos que no tiene que ser así... por eso siempre hay que ser analítico con lo que te devuelve el ordenador; es decir, saber interpretar lo que aparece por pantalla. Pues éste no se confunde, únicamente saca lo que tú le mandas... si sabes la solución y no es lo que esperas casi seguro que te has confundido ó necesitas más precisión)

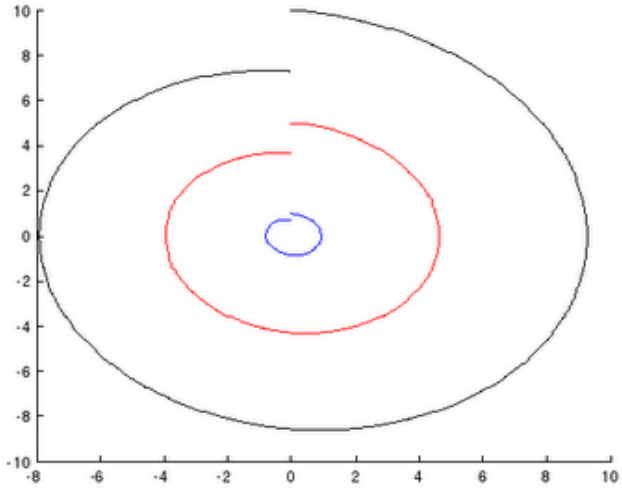
```
clear all
a=0;
b=2*pi;
h=0.0001;
N=(b-a)/h;
t(1)=0;
z=[0 0 0];
q=[1 5 10];
hold on
for i=1:3
x(1)=z(i);
y(1)=q(i);
for k=1:N
t(k+1)=t(k)+h;
x(k+1)=x(k)+h*y(k);y(k+1)=y(k)-h*x(k);
end
if i==1
plot(x,y)
end
if i==2
plot(x,y,'r')
```

```
end
if i==3
plot(x,y,'k')
end
end
hold off
```



¡Ojo con lo que parecen las figuras!, aparentan ser elipses pero hay que darse cuenta de que la escala en ambos ejes no es la misma. Como siempre, lo mejor que se puede hacer es ser crítico.

Si consideramos ahora el oscilador armónico amortiguado, de ecuación genérica $x''+kx'+x=g(t)$ con k la constante de oscilación, únicamente tendremos que cambiar en el código la parte correspondiente a las ecuaciones de aproximación del método de Euler; y obtendremos las esperadas espirales del diagrama de fases



Comparemos ahora los tres métodos que he mencionado antes: *Euler*, *Euler mejorado* y *Runge-Kutta*. Para ello lo mejor es verlo con un ejemplo. Pero antes recordemos cómo eran los algoritmos numéricos de cada uno: el de Euler ya le conocemos, el de Euler mejorado es

$$x_{i+1}=x_i+h2(f_i+f(t_{i+1},x_i+hf_i))$$

mientras que el de Runge-Kutta es

$$x_{i+1}=x_i+16(F_1+2F_2+2F_3+F_4) \text{ con } \begin{cases} F_1=hf_i \\ F_2=hf(t_i+h/2,x_i+F_1/2) \\ F_3=hf(t_i+h/2,x_i+F_2/2) \\ F_4=hf(t_i+h,x_i+F_3) \end{cases}$$

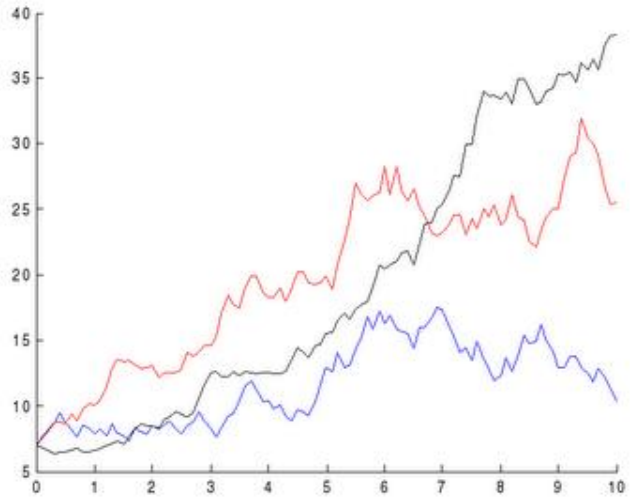
Resolvamos el siguiente problema por los tres métodos con diferentes pasos (h=0.1,0.05,0.001)

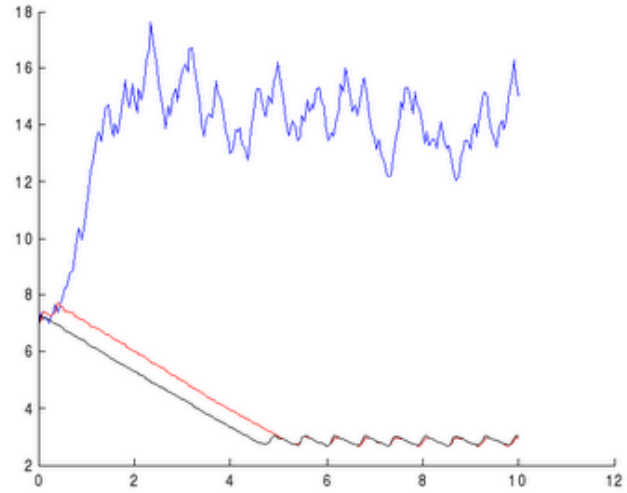
$$\{x'=2+x\sin(10(x+t))x(0)=7 cont\in[0,10]$$

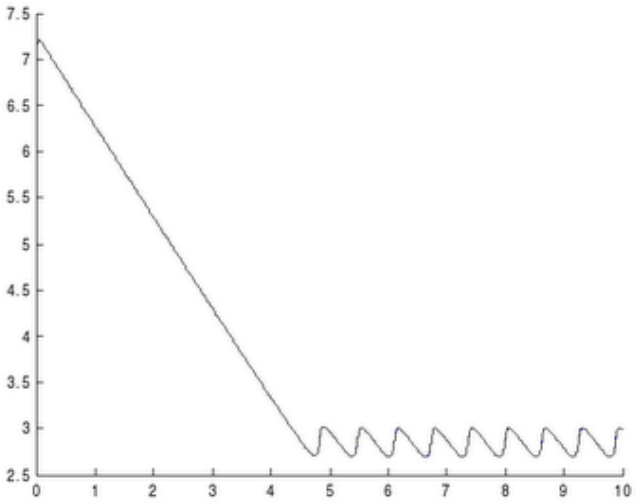
El código es

```
clear all
a=0;
b=10;
f=@(t,x)[2+x*sin(10*(x+t))];
F1=@(t,x,h)[h*f(t,x)];
F2=@(t,x,h)[h*f(t+h/2,x+F1(t,x,h)/2)];
F3=@(t,x,h)[h*f(t+h/2,x+F2(t,x,h)/2)];
F4=@(t,x,h)[h*f(t+h,x+F3(t,x,h))];
t(1)=a; x1(1)=7;x2(1)=7;x3(1)=7;
h=0.1;
N=(b-a)/h;
for k=1:N
t(k+1)=t(k)+h;
x1(k+1)=x1(k)+f(t(k),x1(k))*h;
x2(k+1)=x2(k)+h/2*(f(t(k),x2(k))+f(t(k+1),x2(k)+h*f(t(k),x2(k))));
x3(k+1)=x3(k)+(F1(t(k),x3(k),h)+2*F2(t(k),x3(k),h)+2*F3(t(k),x3(k)
,h)+F4(t(k),x3(k),h))/6;
end
hold on
plot(t,x1,'b')%solucion en azul para el metodo de Euler
plot(t,x2,'r')%solucion en rojo para euler mejorado
plot(t,x3,'k')%solucion en negro para Runge-Kutta
hold off
```

y las figuras por orden decreciente del paso son







Como veis el método de Euler necesita de un paso bastante fino para proporcionar una solución cercana a la real, mientras de los otros dos son capaces de llegar hasta ella sin necesidad de pasos tan pequeños. Realmente ésto en los ejemplos que he expuesto no tiene mucha importancia ya que los programas no necesitan de gran cantidad de operaciones para ejecutarse. Donde la eficiencia se pone de manifiesto es en aquellos programas que usan gran cantidad de operaciones ó necesitan de mucha precisión ó trabajan con datos muy grandes (por ejemplo, con matrices de... no sé, digamos 500×500).

Por cierto, en la última figura no es que sólo dibuje la solución que proporciona Runge-Kutta. Dibuja todas, lo que pasa es que coinciden y la última que pinta tapa a las demás. Viendo la primera figura cualquiera diría que la solución es como la última... pero esto son métodos numéricos, y ya sabemos como son las primeras aproximaciones.

Publicado por Vic : http://mates-vic.blogspot.com/2012/11/usando-matlab-resolucion-de-pvis.html