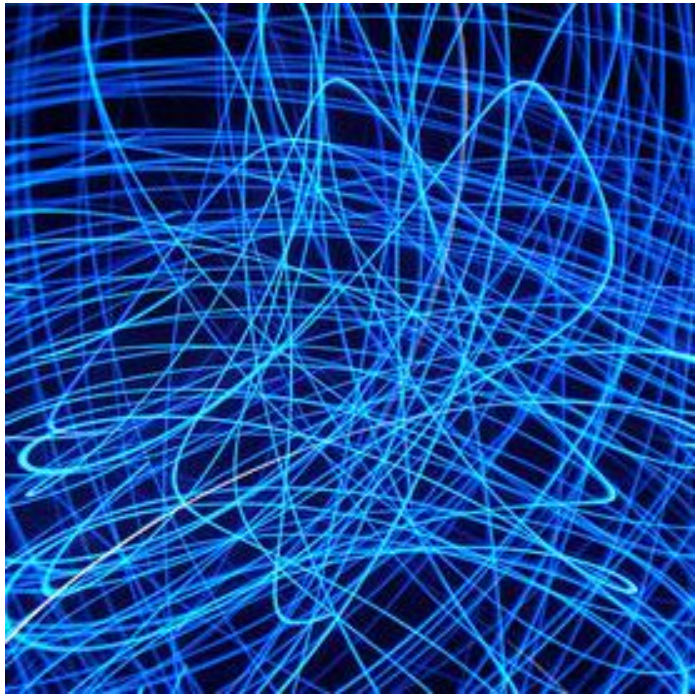
Unidad 5: Integración de ecuaciones de movimiento

Métodos:

- Euler
- Verlet
- Verlet de velocidades
- Runge-Kutta

Aplicación a integración de ecuaciones de movimiento de Newton.

Órbitas.
Sistemas de partículas.
Sistemas mecánicos.



Ecuaciones diferenciales ordinarias (EDO) de primer orden.

Sea la EDO de primer orden, con condición inicial:

$$dx/dt=f(x) \quad \text{con condición inicial } x(t_0)=x_0$$

Se conoce $f(x)$ y se pretende obtener $x(t)$.

La serie de Taylor de $x(t)$ (hasta orden uno)

$$x(t)=x(t_0)+(dx/dt)_{t=t0}(t-t_0)+...$$

que tomando

$$\Delta t=(t-t_0) \quad y \quad f(x_0)=(dx/dt)_{t=t0}$$

se tiene

$$x(t)=x_0+f(x_0)\,\Delta t$$

El **método de Euler** se basa en utilizar recursivamente (iterativamente) la expresión anterior para encontrar una secuencia de soluciones x en valores sucesivos de t:

$$x_1=x_0+f(x_0)\,\Delta t$$
$$x_2=x_1+f(x_1)\,\Delta t$$
$$x_3=x_2+f(x_2)\,\Delta t$$
$$...$$
$$x_{n+1}=x_n+f(x_n)\,\Delta t$$

Ejemplo:

Considérese la EDO:

$dx/dt=a*(x-f)$ con a=1, f=20, $x_0=x(t=0)=100$.

La solución analítica es conocida:

$x(t)=f+e^{at}\,(x_0-f)$

Escribir un script que obtenga x(t) en el intervalo [0,5] mediante el método de Euler, y comparar gráficamente con la solución analítica para diversos valores de Δt.

```
a=1 ; f=20 ; x0=100 ;
Dt=0.1 ; t_ini=0 ; t_fin=5 ;
n=(t_fin-t_ini)/Dt ; % número de iteraciones de Euler
x=zeros(1,n) ;
t=zeros(1,n) ;
t(1)=t_ini ; x(1)=x0 ;
for i=2:n
    t(i)=t(i-1)+Dt ;
    x(i)=x(i-1)+a*(x(i-1)-f)*Dt ;
end

t_ana=[t_ini:Dt/10:t_fin] ; x_ana=f+exp(a*t_ana)*(x0-f) ;
plot(t_ana,x_ana,t,x,'*') ;
xlabel('t'); ylabel('x'); legend('Analítico','Numérico');
```

Método de Euler aplicado a las ecuaciones de movimiento de Newton

En coordenadas cartesianas (para *x*), para una fuerza conocida *F(x,v,t)* sobre una partícula de masa *m*, con aceleración *a(x)=F(x,v,t)/*m se tiene una EDO de segundo orden que puede reducirse a un sistema de dos EDO de primer orden:

dv/dt=a(x,v,t)
dx/dt=v donde *v* se obtiene de la ecuación anterior.
Con condiciones iniciales x_0 y v_0.

Se aplica el método de Euler de forma iterativa, escogiendo un intervalo Δt *suficientemente* pequeño:

$$v_{n+1}=v_n+a_n \Delta t$$
$$x_{n+1}=x_n+v_n \Delta t$$

Ejercicio: Emplear el método de Euler para encontrar la trayectoria 2D y(x) de un proyectil en el campo gravitatorio terrestre (a_y=-9.8 m/s^2). Posteriormente, añadir cierta fricción dependiente de la velocidad.

Método de Verlet

Este método se basa en desarrollar en serie de Taylor hasta orden 3, hacia delante y hacia detrás en el tiempo:

$$x(t_i+\Delta t) = x(t_i)+\Delta t\, v(t_i)+\frac{1}{2}\Delta t^2\, a(t_i)+\frac{1}{6}\Delta t^3 b(t_i)+O\left(\Delta t^4\right)$$

$$x(t_i-\Delta t) = x(t_i)-\Delta t\, v(t_i)+\frac{1}{2}\Delta t^2\, a(t_i)-\frac{1}{6}\Delta t^3 b(t_i)+O\left(\Delta t^4\right)$$

Operando llegamos a:

$$x(t_i+\Delta t) = 2x(t_i)-x(t_i-\Delta t)+\Delta t^2\, a(t_i)+O\left(\Delta t^4\right)$$

Puesto de forma recursiva tenemos que

$$x_{n+1} = 2x_n - x_{n-1} + \Delta t^2\, a_n;$$

$$a_{n+1} = F(x_{n+1})/m;$$

NOTAS:

Es más exacto que Euler, ya que desarrolla Taylor hasta orden 3.

No calcula la velocidad, aunque puede calcularse *a posteriori*:

$$v_i = \frac{x_{i+1}-x_{i-1}}{2\Delta t}$$

También puede calcularse la velocidad en el momento, aunque con menos exactitud:

$$v_{i+1} = \frac{x_{i+1}-x_i}{\Delta t}$$

Método Verlet de velocidades

Es adecuado cuando hace falta calcular la velocidad o la energía cinética durante el movimiento. Véase https://en.wikipedia.org/wiki/Verlet_integration#Velocity_Verlet

Las ecuaciones recursivas son:

$$x_{n+1} = x_n + v_n\,\Delta t + \frac{1}{2} a_n\,\Delta t^2;$$

$$a_{n+1} = F(x_{n+1})/m;$$

$$v_{n+1} = v_n + \frac{a_n + a_{n+1}}{2}\,\Delta t\,;$$

Ejercicios

Ejercicio: Para poner a prueba los métodos anteriores, puede integrarse el movimiento de un oscilador armónico de masa 0.1 kg y constante elástica 1.2 N/m durante un intervalo de tiempo de 10 s, con posición inicial 0.2 m y velocidad inicial -0.3 m/s.

Deben compararse gráficamente las soluciones que se obtienen con los métodos de Euler, Verlet, Verlet de velocidades y analítico. Comprobar la conservación de la energía. Añadir rozamiento para generar el movimiento de un oscilador amortiguado.

Ejercicio: Resolver el problema de la órbita de la Luna o de un planeta (con datos reales) seleccionando la posición y velocidad iniciales y obteniendo en periodo de revolución y los semiejes de la órbita (encontrando los ceros de la distancia radial y sus máximos y mínimos).

Ejercicio: Obtener el periodo de un péndulo (no linealizado) en función de la amplitud de oscilación.

Métodos Runge-Kutta

Se trata de métodos avanzados que proporciona MATLAB, cuyo funcionamiento interno se estudiará en Computación II. Aquí podrá emplearse como *caja negra*.

Sirven para resolver sistemas de ecuaciones diferenciales de primer orden. Veamos un ejemplo para resolver el tiro parabólico.

Sintaxis básica:

```
[t,y]=ode45(odefun,tspan,y0)
```

`odefun` define el sistema de ecuaciones que se pretende resolver.
`tspan` es el intervalo (de tiempo, por ejemplo) en el que se quiere la solución.
`y0` son las condiciones iniciales.
`t` es un vector con los instantes de tiempo en los que se ha obtenido la solución.
`y` son las soluciones del sistema de ecuaciones diferenciales.

Runge-Kutta: un ejemplo sencillo

Se pretende obtener la solución *z(t)* de un tiro parabólico, con posición inicial z_0=100 m, velocidad inicial v_{z0}=30 m/s y aceleración a_z=-10 m/s2 en el intervalo de tiempo de 0 a 10 s.

El sistema de ecuaciones es:

$v_z=dz/dt$

$a_z=-10$

con condiciones iniciales $z_0=100$, $v_{z0}=30$.

Tomando $y(1)=z$, $y(2)=v_z$ se tiene:

$dy(1)/dt=y(2)$

$dy(2)/dt=-10$

con condiciones iniciales $y_0(1)=100$, $y_0(2)=30$.

```
fun=@(t,y) [y(2);-10] ; % función sistema ODE
[t,y]=ode45(fun,[0 10],[100 30]) ;
plot(t,y(:,1)) ; xlabel('tiempo (s)') ; ylabel('altura (m)') ;
```

Ejercicio: Integrar el movimiento de un oscilador armónico de masa 0.1 kg y constante elástica 1.2 N/m durante un intervalo de tiempo de 10 s, con posición inicial 0.2 m y velocidad inicial -0.3 m/s. Comprobar la conservación de la energía.

```
fun=@(t,y) [y(2);-1.2*y(1)/0.1] ; % función sistema ODE
[t,y]=ode45(fun,[0 10],[0.2 -0.3]) ;
plot(t,y(:,1)) ; xlabel('tiempo (s)') ; ylabel('posición (m)') ;
```

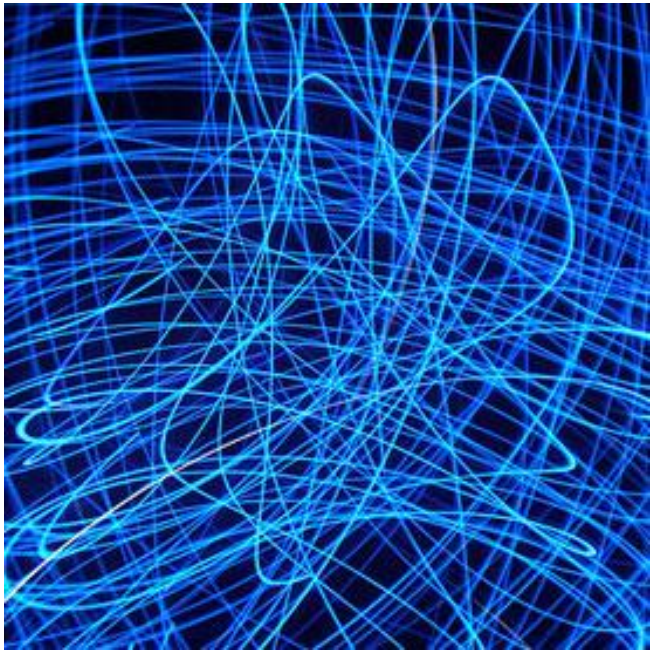
Ejercicio: Añadir al ejercicio anterior una fuerza de fricción proporcional a la velocidad de -0.1 N/(m/s).

```
fun=@(t,y) [y(2);(-1.2*y(1)-0.1*y(2))/0.1] ; % función sistema ODE
[t,y]=ode45(fun,[0 10],[0.2 -0.3]) ;
plot(t,y(:,1)) ; xlabel('tiempo (s)') ; ylabel('posición (m)') ;
```

Ejercicio: añadir al ejercicio anterior una fuerza que dependa del tiempo.

Sistemas de partículas.

Puede resolverse el movimiento combinado de un sistema de muchas partículas sometidas a interacciones mutuas. Para ello hay que integrar iterativamente la ecuación de movimiento de cada partícula, que en cada momento está sujeta a la fuerza que ejercen sobre ella todas las demás partículas. Este tipo de cálculos se llama Dinámica Molecular. Hay varias propuestas de proyectos de la asignatura basados en éste tipo de cálculos.



UNIDAD 5. Destrezas

Resolución de EDO de primer orden por método de Euler

Reducción de EDO de segundo orden a sistema de EDOs de primer orden

Resolución de sistemas de EDOs de segundo orden mediante: Euler, Verlet, Verlet velocidades y Runge-Kutta.

Resolución de sistemas mecánicos newtonianos, en varias dimensiones, con fuerzas que dependan de la posición y/o la velocidad.

EJEMPLO: Órbita Lunar SIMPLIFICADA (NO está optimizado NI vectorizado)

```
%% datos y constantes
clear all ;
MT=5.97219E24 ; % kg
ML=7.34767309E22 ; % kg
G=6.673E-11 ; % N*(m/kg)^2

%% valores iniciales
x0=384E6 ; y0=0 ; % m
vx0=0 ; % m/s (empezamos en perigeo o apogeo)
vy0=1.3E3 ; % La verdadera es vy0=1.023E6 ; % m/s

%% funciones anónimas, calculan las componentes x e y de la Fuerza
Rfun=@(x,y) sqrt(x.^2+y.^2) ;
Ffun=@(x,y) -G*MT*ML./(Rfun(x,y).^2) ;
Fxfun=@(x,y) Ffun(x,y).*x./Rfun(x,y) ;
Fyfun=@(x,y) Ffun(x,y).*y./Rfun(x,y) ;

%% inicializa vectores de variables
n=500 ; % n puntos que se integran (deberían ser más)
t_min=0 ; t_max=4*1E7 ; % intervalo de tiempo (s)
x=zeros(1,n) ; y=x ; vx=x ; vy=y ; ax=x ; ay=x ;
x(1)=x0 ; y(1)=y0 ; vx(1)=vx0 ; vy(1)=vy0 ;
```

SIGUE

Continuación órbita lunar

```
%% cálculo x(t) y(t) por Verlet velocidades
t=zeros(1,n) ; Dt=(t_max-t_min)/n ; t(1)=t_min ;

for i=1:n-1
    ax(i)=Fxfun(x(i),y(i))/ML ;
    x(i+1)=x(i)+vx(i)*Dt+ax(i)*Dt^2/2 ;
    ay(i)=Fyfun(x(i),y(i))/ML ;
    y(i+1)=y(i)+vy(i)*Dt+ay(i)*Dt^2/2 ;

    ax(i+1)=Fxfun(x(i+1),y(i+1))/ML ;
    ay(i+1)=Fyfun(x(i+1),y(i+1))/ML ;

    vx(i+1)=vx(i)+(ax(i)+ax(i+1))*Dt/2 ;
    vy(i+1)=vy(i)+(ay(i)+ay(i+1))*Dt/2 ;

    t(i+1)=t(i)+Dt ; % vector tiempos
end

plot(x,y,0,0,'o') ; axis equal ; grid on ;
xlabel('x (m)') ; ylabel('y (m)') ; title('Órbita lunar') ;
```