

# Física Computacional

**Alberto Pérez Muñozuri** (despacho107)

Contacto:

[alberto.perez.munuzuri@usc.es](mailto:alberto.perez.munuzuri@usc.es)

[fisicacomputacional.usc@gmail.com](mailto:fisicacomputacional.usc@gmail.com)

## **Referencias del Capítulo:**

- Numerical Recipes. W.H. Press, B.P. Flannery, S.A. Teukolsky and W.T. Vetterling. Cambridge University Press (1988).
- Computational Techniques for Fluid Dynamics. C.A.J. Fletcher. Springer-Verlag (1991).

## **NORMAS:**

- Básicamente igual que con Diego
- Control de asistencia por firma en el listado.
- Programación en Python (o Matlab)
- Evaluación continua: boletines entregados, controles (3 o 4, se avisara unos días antes de cada), defensa de los boletines entregados, asistencia
- Examen final opcional, para los que no hagan completa la evaluación continua, en la fecha prevista por el rectorado.
- Todos los boletines se enviarán al Aula Virtual o a la dirección:

**[fisicacomputacional.usc@gmail.com](mailto:fisicacomputacional.usc@gmail.com)**

# Ecuaciones Diferenciales Ordinarias (ODE)

---

Cualquier ecuación diferencial ordinaria se puede reducir a un conjunto de ecuaciones diferenciales ordinarias de primer orden:

$$\text{Ej.: } \frac{d^2x}{dt^2} + q(t)\frac{dx}{dt} = r(t) \Rightarrow \begin{aligned} \frac{dx}{dt} &= y(t) \\ \frac{dy}{dt} &= r(t) - q(t)y \end{aligned}$$

Se procede de modo equivalente para ecuaciones de orden superior. Por tanto, consideramos el caso de ecuaciones diferenciales de primer orden.

$$\frac{dx_i}{dt} = f_i(t, x_1, \dots, x_N) \quad i = 1, \dots, N$$

Para tener el problema correctamente planteado y tener una solución única al mismo necesitamos conocer las condiciones de frontera (boundary conditions), o condición inicial (initial conditions):

$$x_i(t = t_0) = A_i \quad i = 1, \dots, N$$

## Ec. Diferenciales Ordinarias: Método de Euler

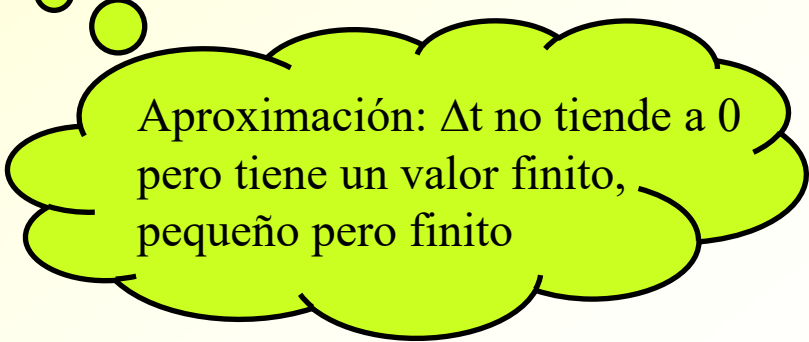
---

Consideremos el siguiente problema:  $\frac{dx}{dt} = \text{sen}(x)$

El ordenador no sabe como hacer derivadas, solo sabe hacer sumas. Aplicamos conceptos vistos en la parte anterior para manipular esta ecuación.

Aplicamos la definición de derivada:

$$\frac{dx}{dt} = \lim_{\Delta t \rightarrow 0} \frac{x(t + \Delta t) - x(t)}{\Delta t} \approx \frac{x(t + \Delta t) - x(t)}{\Delta t}$$



Aproximación:  $\Delta t$  no tiende a 0 pero tiene un valor finito, pequeño pero finito

$$\Rightarrow x(t + \Delta t) = x(t) + \Delta t \text{ sen}(x(t))$$

Podemos conocer el siguiente valor de la variable  $x$  a partir de lo que valía un instante de tiempo  $\Delta t$  antes.

## Ec. Diferenciales Ordinarias: Método de Euler

Introducimos la siguiente notación:

$$x(t = 0) = x_0$$

$$x(t = \Delta t) = x_1$$

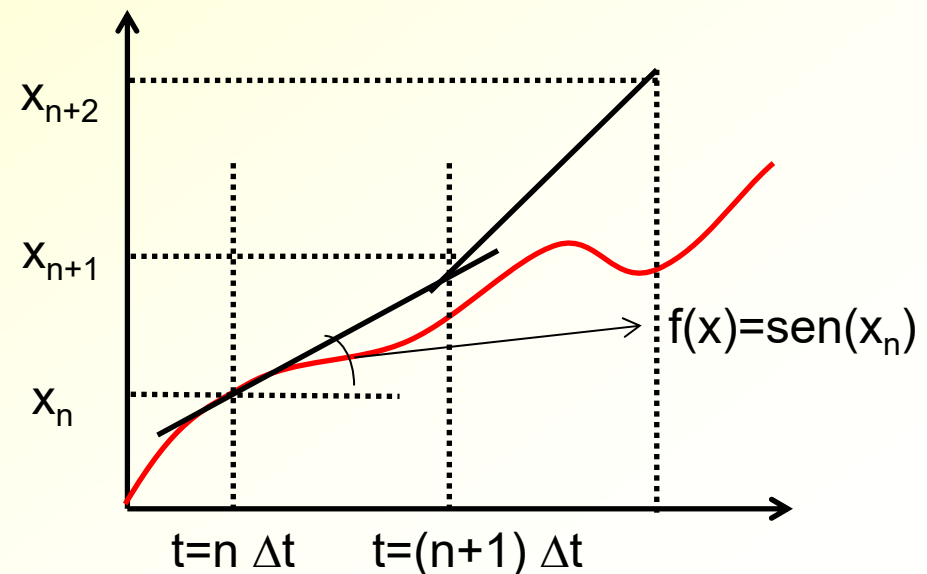
$$x(t = n \Delta t) = x_n$$

De modo que la ecuación diferencial anterior queda de la forma

$$x_{n+1} = x_n + \Delta t \operatorname{sen}(x_n)$$

que ya es una ecuación algebraica que el ordenador sabe resolver.

Interpretación geométrica:



## Ec. Diferenciales Ordinarias: Método de Euler

Error del esquema de integración para la ecuación general:  $\frac{dx}{dt} = f(x, t)$

$$\begin{aligned}x_{n+1} &= x_n + \Delta t \, f(x_n, t_n) \\ &= x(t + \Delta t)\end{aligned}$$

$$\approx x(t) + \Delta t \left. \frac{dx}{dt} \right|_{\Delta t=0} + O(\Delta t^2)$$

Desarrollo en  
Serie de Taylor en  
torno a  $\Delta t=0$

$$= f(x(t), t)$$

Método de orden 1  $\rightarrow$   
aproximación a primer orden  $\rightarrow$   
el error va con  $\Delta t^2$

```
import numpy as np
import matplotlib.pyplot as plt
```

```
deltat=0.1
```

```
t=0
```

```
x=1
```

```
for i in range(100):
```

```
    t=t+deltat
```

```
    xnew=x+deltat*np.sin(x)
```

```
    x=xnew
```

```
    plt.plot(t,x, '*')
```

```
    plt.xlabel('Tiempo')
```

```
    plt.ylabel('x')
```

% Probar diferentes valores de  $\Delta t$ .

% Probar diferentes condiciones iniciales, cada condición inicial da una solución

% diferente.

% Resolución numérica del problema  $dx/dt = \sin(x)$  por el Método de Euler  
close all; clear all;

deltat=0.01; % Valores de los parámetros  
dimt=1000; % Numero total de iteraciones  
x(1)=10; % Condición inicial

% Cálculo de las iteraciones sucesivas  
for n=1:dimt-1  
     $x(n+1)=x(n)+\text{deltat}*\sin(x(n));$   
end

% Dibuja la dependencia temporal  
t=[1:dimt]\*deltat;  
figure(1);hold off;plot(x,'-','LineWidth',2.000);grid on;

% Probar diferentes valores de  $\Delta t$ .

% Probar diferentes condiciones iniciales, cada condición inicial da una solución

% diferente.





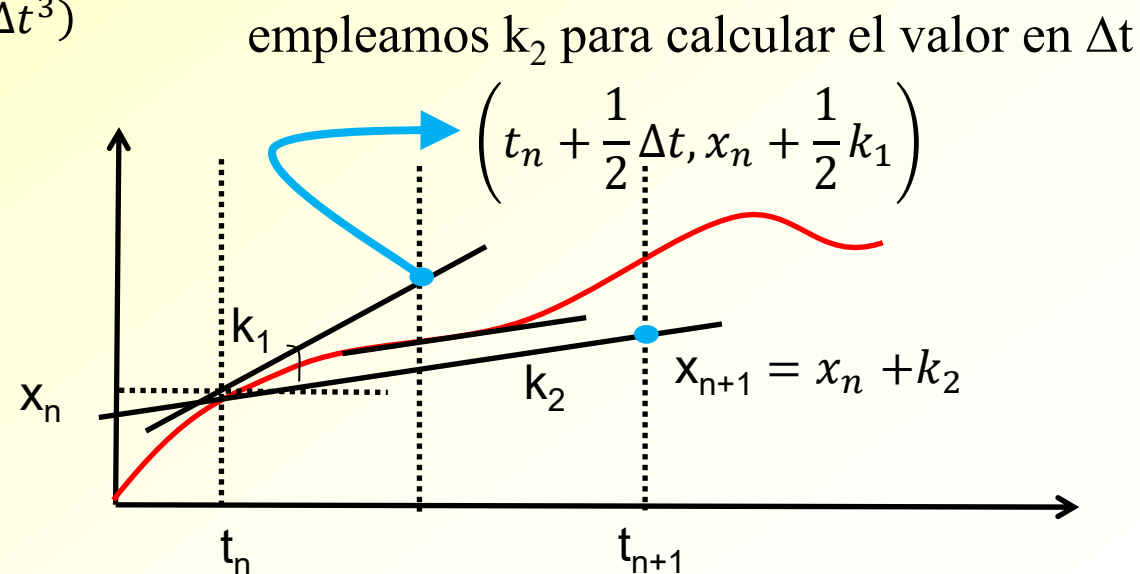
## Ec. Diferenciales Ordinarias: Runge-Kutta 2º orden

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

$$k_1 = \Delta t f(t_n, x_n) \quad \text{Como en el método de Euler}$$

$$k_2 = \Delta t f\left(t_n + \frac{1}{2}\Delta t, x_n + \frac{1}{2}k_1\right) \quad k_1 \text{ se emplea para estimar la pendiente de la función en el punto medio del intervalo}$$

$$x_{n+1} = x_n + k_2 + O(\Delta t^3)$$



```
import numpy as np
import matplotlib.pyplot as plt
```

```
deltat=0.1
```

```
t=0
```

```
x=1
```

```
for i in range(100):
```

```
    t=t+deltat
```

```
    k1=deltat*np.sin(x)
```

```
    k2=deltat*np.sin(x+k1/2.)
```

```
    xnew=x+k2
```

```
    x=xnew
```

```
    plt.plot(t,x, '*')
```

```
    plt.xlabel('t')
```

```
    plt.ylabel('x')
```

```
plt.xlim(0,10)
```

```
plt.ylim(0,4)
```

```
% Probar diferentes valores de  $\Delta t$ .
```

```
% Probar diferentes condiciones iniciales, cada condición inicial da una solución
```

```
% diferente.
```

```
% Resolución numérica del problema  $dx/dt = \sin(x)$   
% por el Método de Runge-Kutta de 2º orden  
close all; clear all;
```

```
deltat=0.01; % Valores de los parámetros  
dimt=1000; % Numero total de iteraciones  
x(1)=10; % Condición inicial
```

```
% Cálculo de las iteraciones sucesivas  
for n=1:dimt-1  
    k1=deltat*sin(x(n));  
    k2=deltat*sin(x(n)+k1/2);  
    x(n+1)=x(n)+k2;  
end
```

```
% Dibuja la dependencia temporal  
t=[1:dimt]*deltat;  
figure(1);hold off;plot(x,'-','LineWidth',2.000);grid on;
```

```
% Probar diferentes valores de  $\Delta t$ .  
% Probar diferentes condiciones iniciales, cada condición inicial da una solución  
% diferente.
```



## Ec. Diferenciales Ordinarias: Runge-Kutta 4º orden

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

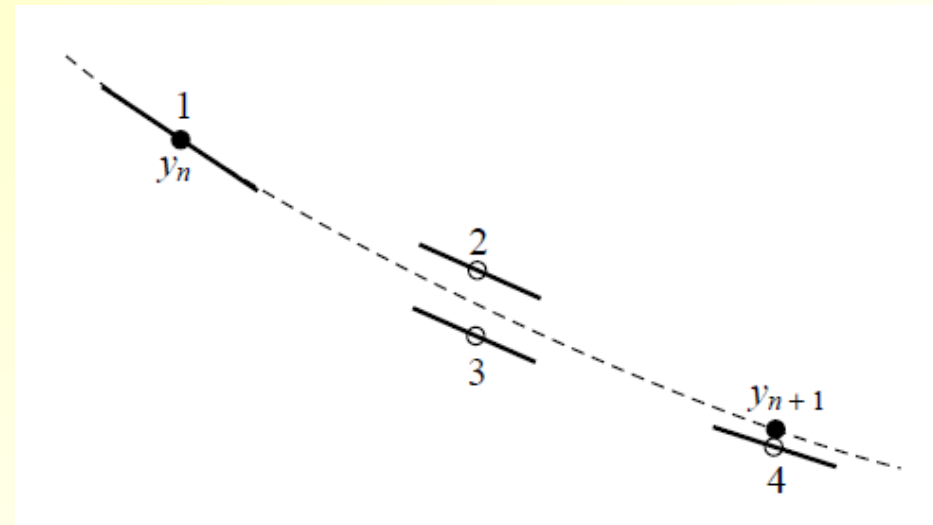
Es una combinación inteligente de pendientes que minimizan el error

$$k_1 = \Delta t f(t_n, x_n)$$

$$k_2 = \Delta t f\left(t_n + \frac{1}{2}\Delta t, x_n + \frac{1}{2}k_1\right)$$

$$k_3 = \Delta t f\left(t_n + \frac{1}{2}\Delta t, x_n + \frac{1}{2}k_2\right)$$

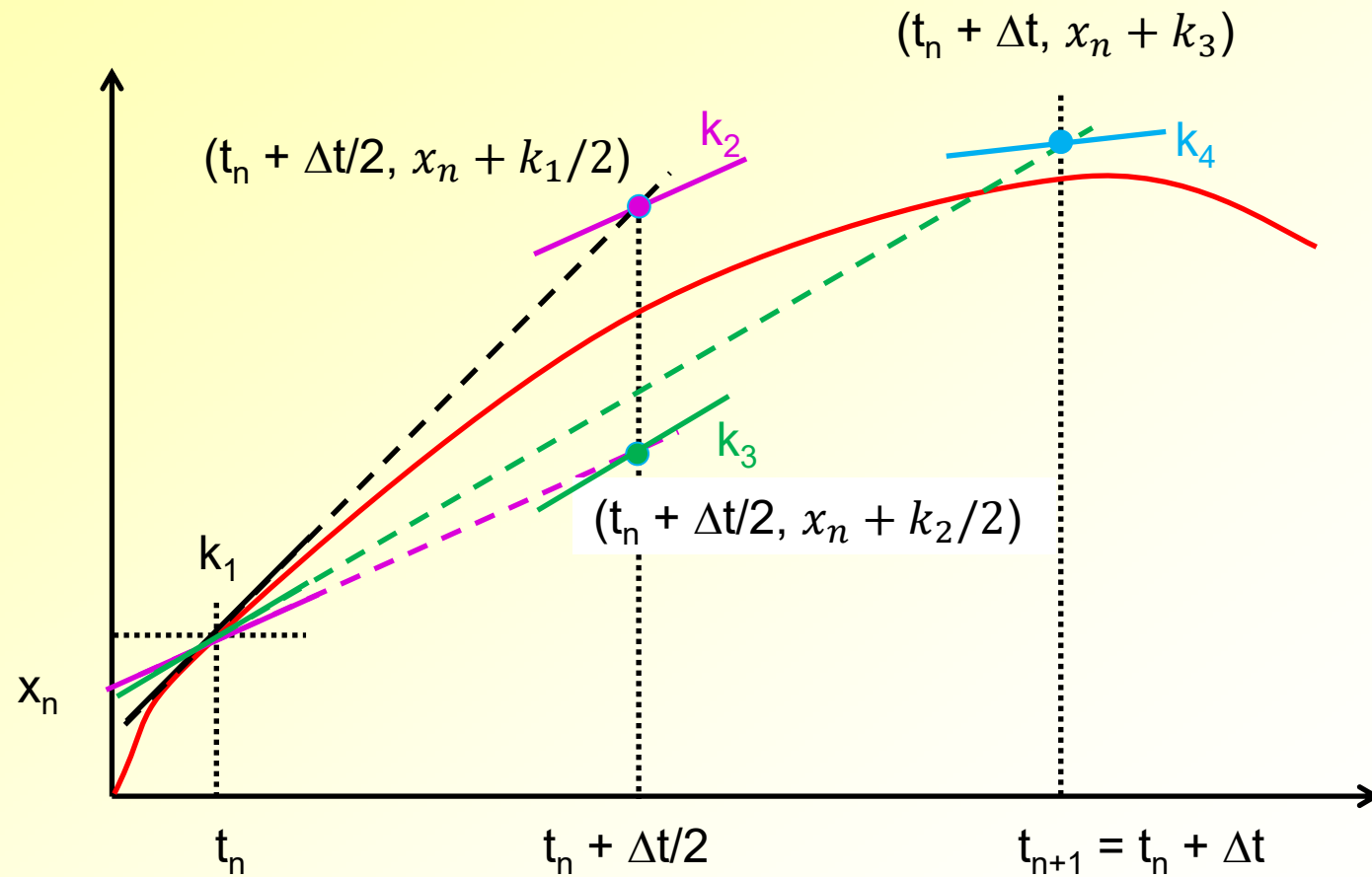
$$k_4 = \Delta t f(t_n + \Delta t, x_n + k_3)$$



$$x_{n+1} = x_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(\Delta t^5)$$

## Ec. Dif. Ordinarias: RK4, interpretación geométrica

$$x_{n+1} = x_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(\Delta t^5)$$



```
import numpy as np
import matplotlib.pyplot as plt
```

```
deltat=0.1; t=0; x=1
for i in range(100):
    t=t+deltat
    k1=deltat*np.sin(x)
    k2=deltat*np.sin(x+k1/2.)
    k3=deltat*np.sin(x+k2/2.)
    k4=deltat*np.sin(x+k3)
    xnew=x+k1/6.+k2/3.+k3/3.+k4/6.
    x=xnew
    plt.plot(t,x, '*')
    plt.xlabel('t')
    plt.ylabel('x')
```

```
plt.xlim(0,10)
plt.ylim(0,4)
```

% Probar diferentes valores de  $\Delta t$ .

% Probar diferentes condiciones iniciales, cada condición inicial da una solución  
% diferente.

```

1 from math import *
2 from matplotlib.pyplot import *
3
4 deltat=0.1
5 t=0
6 x=1
7 for i in range(100):
8     t=t+deltat
9     k1=deltat*sin(x)
10    k2=deltat*sin(x+k1/2.)
11    k3=deltat*sin(x+k2/2.)
12    k4=deltat*sin(x+k3)
13    x=x+k1/6.+k2/3.+k3/3.+k4/6.
14    plot(t,x, '*')
15    xlabel('t')
16    ylabel('x')
17
18 xlim(0,10)
19 ylim(0,4)

```

% Probar diferentes valores de  $\Delta t$ .

% Probar diferentes cond. inic., cada cond. inic. da una solución diferente.

```
% Resolución numérica del problema  $dx/dt = \sin(x)$   
% por el Método de Runge-Kutta de 4º orden  
close all; clear all;
```

```
deltat=0.01; % Valores de los parámetros  
dimt=1000; % Numero total de iteraciones  
x(1)=10; % Condición inicial
```

```
for n=1:dimt-1 % Cálculo de las iteraciones sucesivas  
    k1=deltat*sin(x(n));  
    k2=deltat*sin(x(n)+k1/2);  
    k3=deltat*sin(x(n)+k2/2);  
    k4=deltat*sin(x(n)+k3);  
    x(n+1)=x(n)+k1/6+k2/3+k3/3+k4/6;  
end
```

```
t=[1:dimt]*deltat; % Dibuja la dependencia temporal  
figure(1);hold off;plot(x,'-','LineWidth',2.000);grid on;
```

```
% Probar diferentes valores de Dt.  
% Probar diferentes cond. inic., cada cond. inic. da una solución diferente.
```





## Ec. Dif. Ordinarias: sistemas de 2 ecuaciones

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

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

### Método de Euler

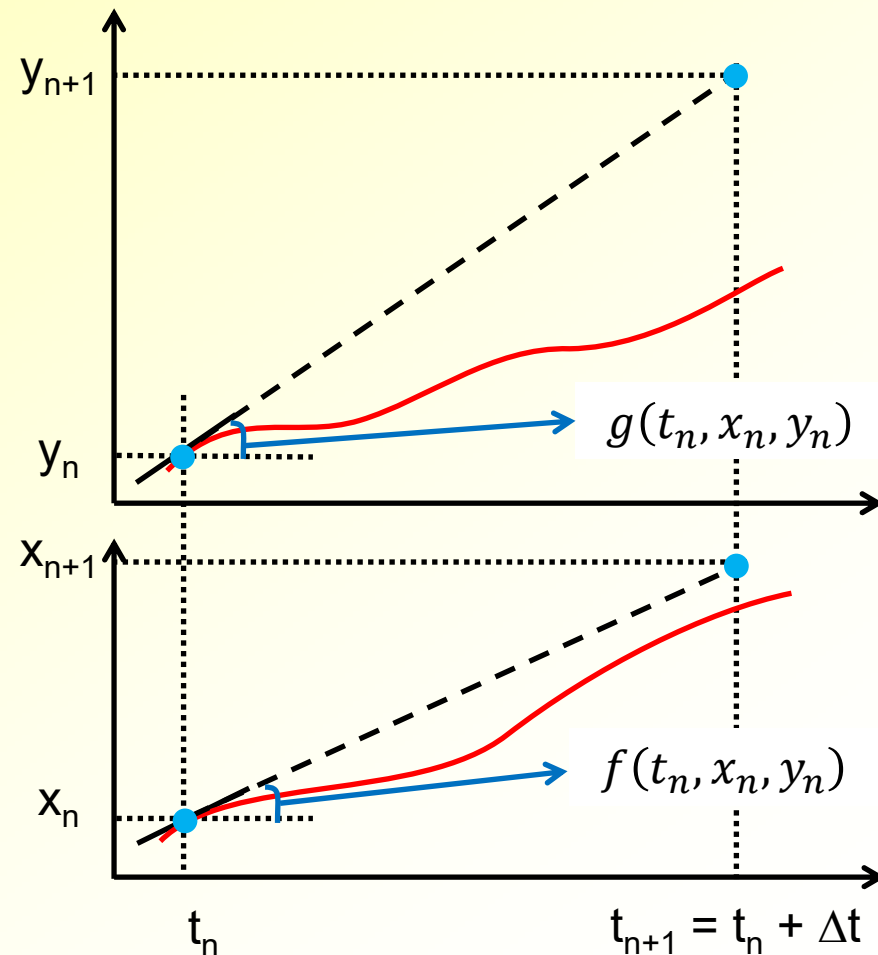
$$x_{n+1} = x_n + \Delta t f(t_n, x_n, y_n)$$

$$y_{n+1} = y_n + \Delta t g(t_n, x_n, y_n)$$

Ejemplo el ejercicio 2 del  
boletín: oscilador armónico  
sin amortiguamiento.

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

$$\frac{dy}{dt} = g(t, x, y) = -\omega_0^2 x$$



## EDO: RK2

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

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

Ejemplo: oscilador

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

$$\frac{dy}{dt} = g(t, x, y) = -\omega_0^2 x$$

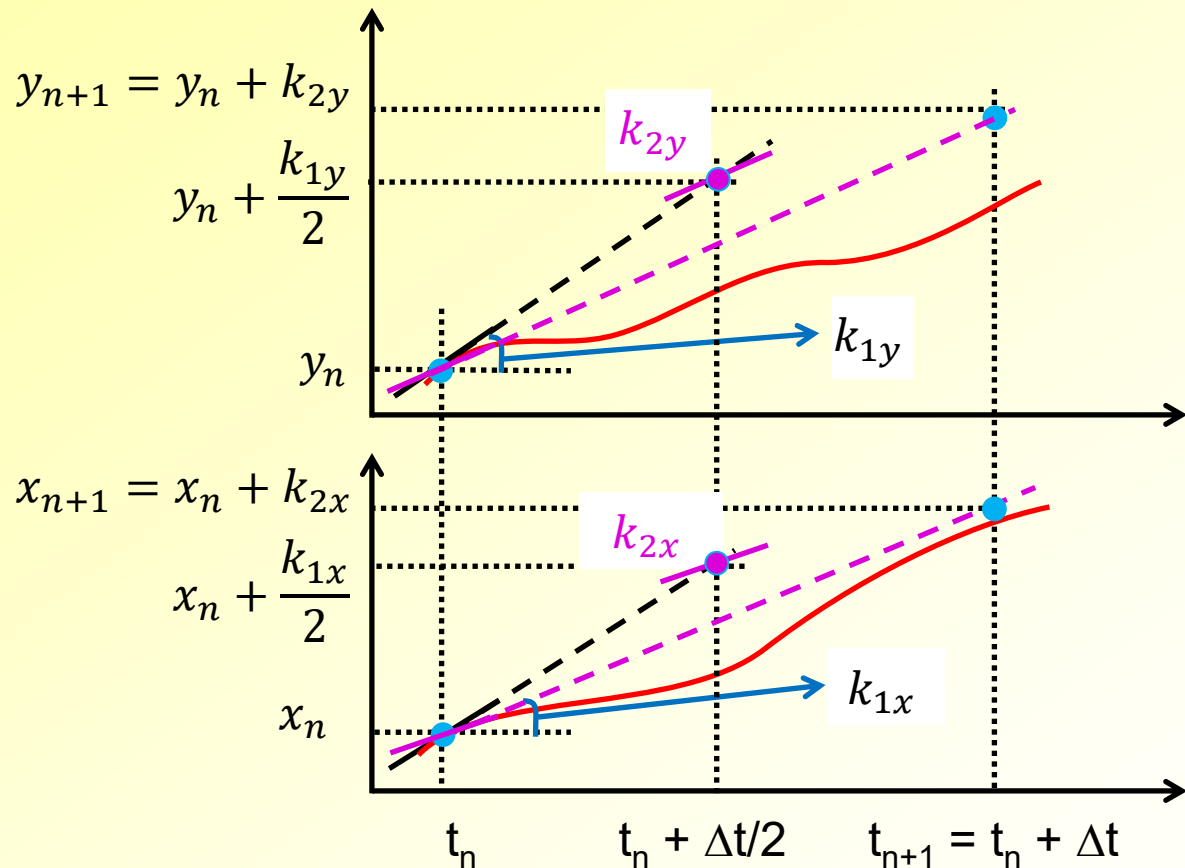
### Runge Kutta 2 orden

$$k_{1x} = \Delta t f(t_n, x_n, y_n)$$

$$k_{1y} = \Delta t g(t_n, x_n, y_n)$$

$$k_{2x} = \Delta t f\left(t_n + \frac{\Delta t}{2}, x_n + \frac{k_{1x}}{2}, y_n + \frac{k_{1y}}{2}\right)$$

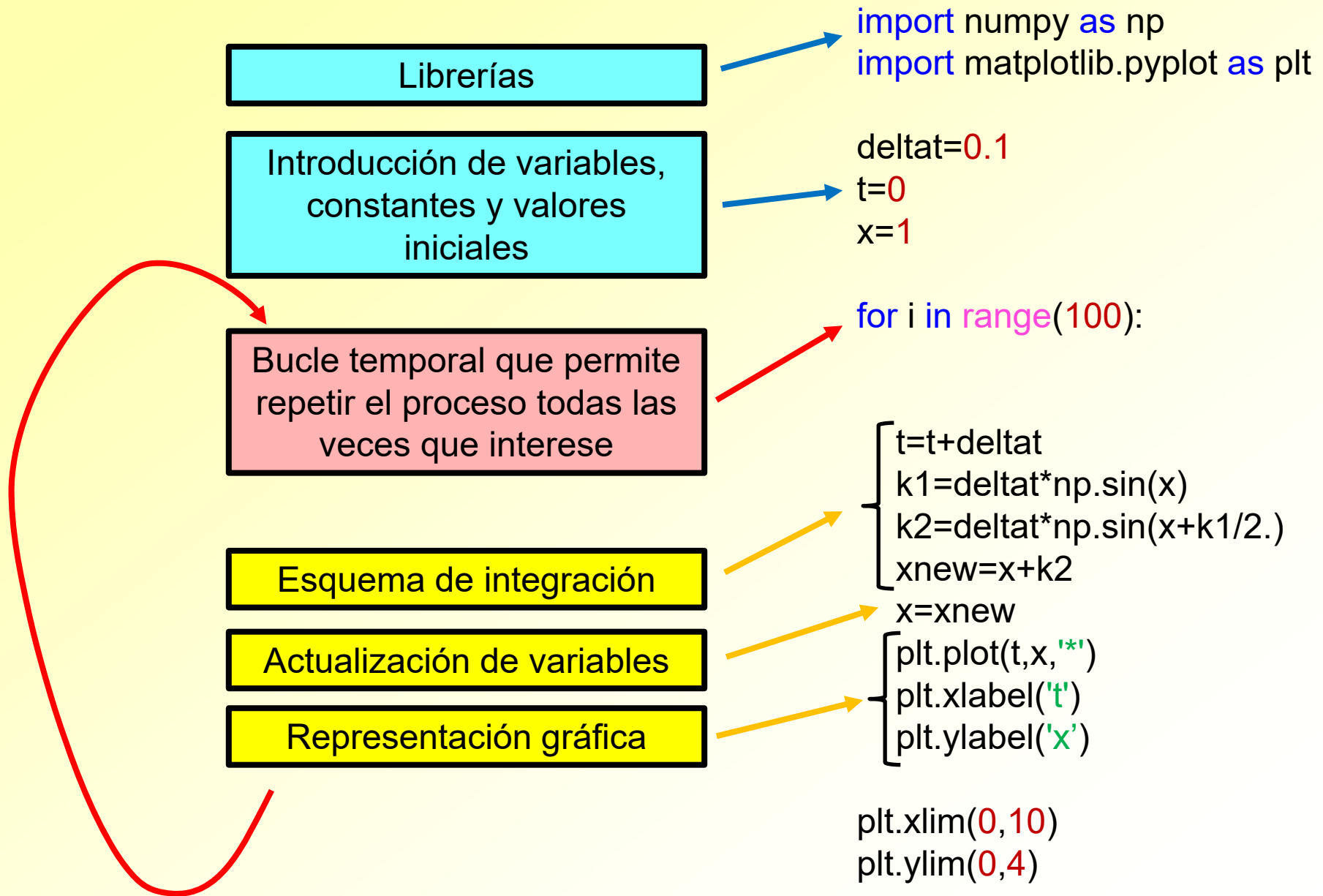
$$k_{2y} = \Delta t g\left(t_n + \frac{\Delta t}{2}, x_n + \frac{k_{1x}}{2}, y_n + \frac{k_{1y}}{2}\right)$$



$$x_{n+1} = x_n + k_{2x} + O(\Delta t^3)$$

$$y_{n+1} = y_n + k_{2y} + O(\Delta t^3)$$

## Ec. Diferenciales Ordinarias: Esquema general de un programa



## **Ec. Diferenciales Ordinarias**

---

- La forma de proceder anterior para un sistema de 2 ecuaciones diferenciales ordinarias se generaliza de forma directa para el Runge Kutta de 4° orden.
- Para sistemas de ec. dif. de orden superior (3 o más ecuaciones, por ejemplo el ejercicio 4 del boletín), se procede igual que antes calculando pendientes para cada una de las variables del sistema.

## Notas sobre los ejercicios del Boletín

- El ejercicio 1 (obligatorio) es una única ecuación diferencial ordinaria que se resuelve como el ejemplo del primer día.
- Los ejercicios 2 y 3 (obligatorios) corresponden con un sistema de dos ecuaciones diferenciales de primer orden que se resuelven como se indicó antes.

Ejercicio 2: oscilador armónico

$$\ddot{x} + \omega_0^2 x = 0 \Rightarrow$$

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

$$\frac{dy}{dt} = g(t, x, y) = -\omega_0^2 x$$

Ejercicio 3: oscilador armónico amortiguado y forzado

$$\ddot{x} + b \dot{x} + \omega_0^2 x = F \cos(\omega t) \Rightarrow \frac{dx}{dt} = f(t, x, y) = y$$

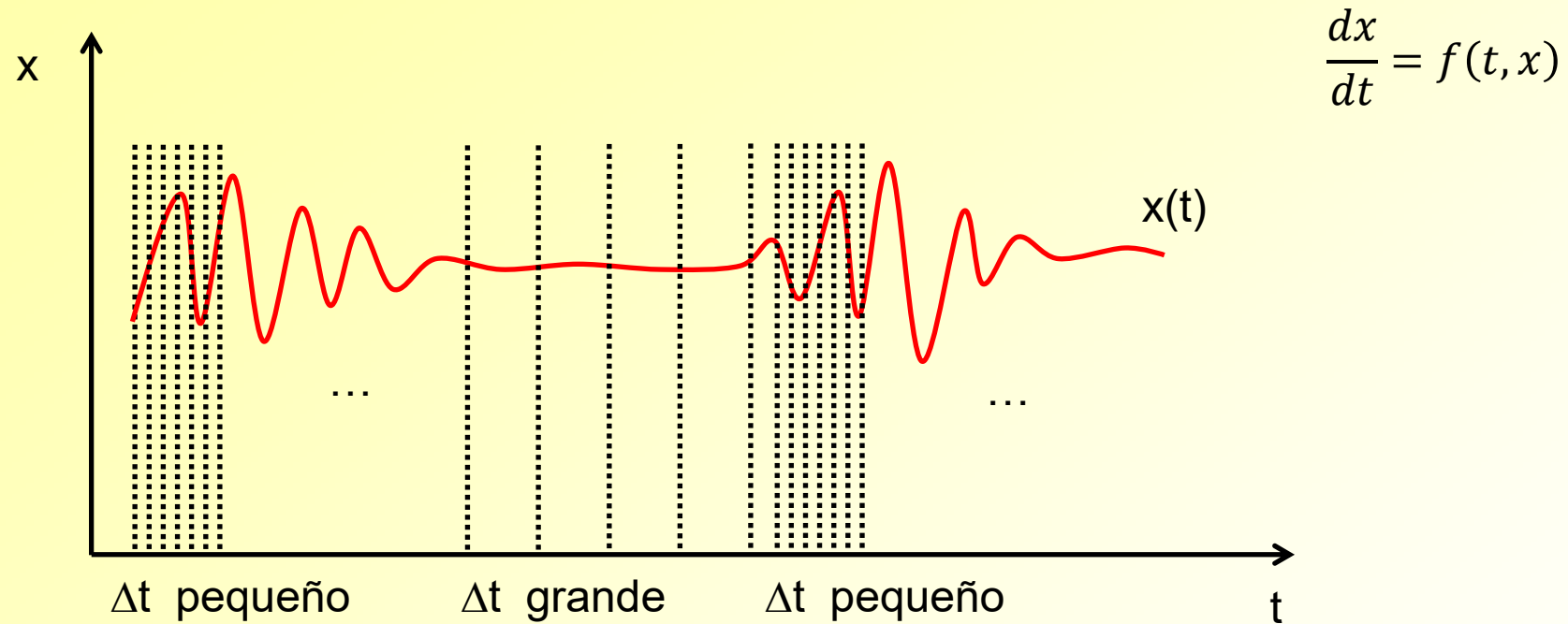
$$\frac{dy}{dt} = g(t, x, y) = -\omega_0^2 x - b y + F \cos(\omega t)$$

- El ejercicio 4 (obligatorio) es un sistema de tres ecuaciones diferenciales de primer orden y se resuelve generalizando el método anterior. Recordad que primero se calcula la primera pendiente para **todas** las variables y estas se emplean para calcular las segundas pendientes de **todas** las variables y así sucesivamente.

## Notas sobre los ejercicios del Boletín

- Ejercicio 5 (opcional): integrar la ec. del ejercicio 1 por un método de paso variable.
- Cuestiones a considerar:
- El método calcula por cualquier esquema (Euler por ejemplo) el siguiente valor  $x_{n+1}$ . En este momento evalúa si la diferencia con el valor anterior,  $x_n$ , es muy grande:
  - Si es muy grande reduce  $\Delta t$  (por ejemplo a la mitad) y vuelve a calcular.
  - Si es muy pequeña la diferencia, incrementa  $\Delta t$  (por ej. lo dobla) y vuelve a calcular.
  - Si está en un intervalo bueno, mantiene el valor de  $\Delta t$  y pasa a calcular el siguiente valor  $x_{n+2}$ .
- Se valorará:
  - que funcione correctamente y dé una solución
  - que el programa sea capaz de incrementar y reducir el valor de  $\Delta t$ .
  - que  $\Delta t$  no diverja o se haga cero.
  - llevar el control correcto del tiempo

## Notas sobre los ejercicios del Boletín



- Se valorará:
- que funcione correctamente y dé una solución
  - que el programa sea capaz de incrementar y reducir el valor de  $\Delta t$ .
  - que  $\Delta t$  no diverja o se haga cero.
  - llevar el control correcto del tiempo

## Modelo de Lorenz

---

La teoría del Caos empezó en 1963 con el trabajo de Lorenz publicado en una oscura revista de ciencias de la atmósfera. Este trabajo fue desconocido por los físicos y matemáticos hasta que poco a poco se lo llegó a ver tan importante como el nacimiento de una nueva ciencia.

En el trabajo de Lorenz se intentaba dar una explicación y predicción global del clima atmosférico. Las ecuaciones que modelaban la atmósfera y que hoy se saben que son insuficientes, son aproximaciones a las ecuaciones de Navier-Stokes. Se llegaba finalmente a estas tres ecuaciones diferenciales :

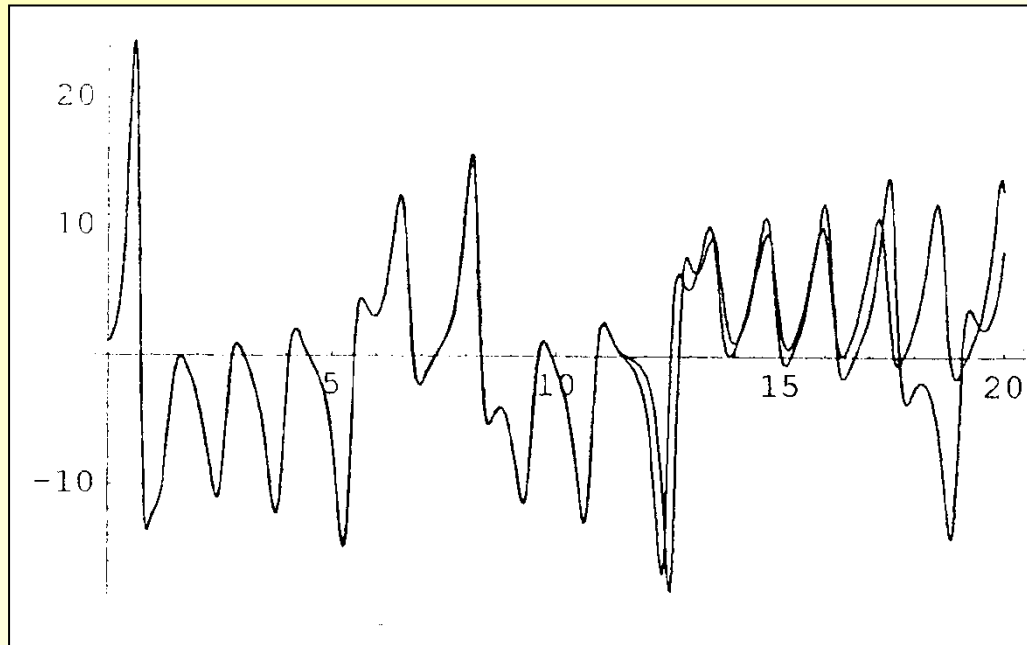
$$\begin{aligned}\frac{dX}{dt} &= \sigma (Y - X) \\ \frac{dY}{dt} &= rX - Y - XZ \\ \frac{dZ}{dt} &= XY - bZ\end{aligned}$$

con  $\sigma$ ,  $r$ ,  $b$  parámetros definidos positivos.



## T1: Dinámica Temporal: Atractor de Lorenz

La sensibilidad a las condiciones iniciales las mostramos en esta figura donde hemos tomado los valores  $\sigma = 3$ ,  $r = 26.5$ ,  $b = 1$ ;  $x_1, y_1, z_1 = (0, 1, 0)$ ;  $x_2, y_2, z_2 = (0, 1.1, 0)$



De todos modos quedaría un punto de duda. En todo esto, hemos analizado el sistema de Lorenz numéricamente con computadora. Cabría desconfianza ante este método. Sin embargo Tucker, en 1999, probó que el sistema de Lorenz es un verdadero atractor caótico basándose en el concepto de hiperbolicidad singular y en la técnica de formas normales .