Ecuaciones diferenciales ordinarias 3

Curso de Física Computacional

M. en C. Gustavo Contreras Mayén

23 de abril de 2014

Contenido

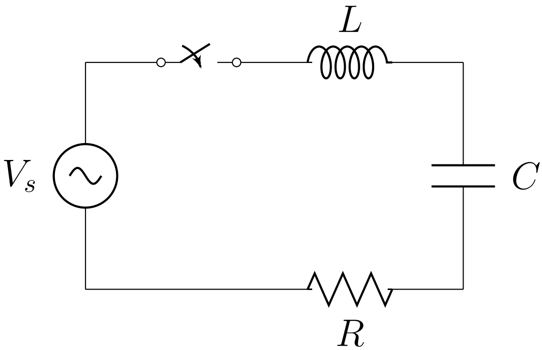
Contenido

Contenido

La corriente eléctrica de un circuito RLC en serie, satisface la ecuación

$$L\frac{di}{dt} + Ri + \frac{1}{C}\int_0^t i(t')dt' + \frac{1}{C}q(0) = E(t), \qquad t > 0 \tag{1}$$

cuando el circuito se cierra en el instante $t = 0$, se tiene que $i = i(t)$ es la corriente, R es la resistencia, L, C, E vienen dadas por: $L = 200H$, $C = 0.001F$, $E(t) = 1V$ para $t > 0$.



Las condiciones iniciales son $q(0) = 0$ (carga inicial del condensador), $i(0) = 0$.

Calcular la corriente para $0 \leq t \leq 5$ segundos y el factor de amortiguamiento y la frecuencia de oscilación del circuito RLC para los siguientes valores de R:

1. $R = 0\,\Omega$
2. $R = 50\,\Omega$
3. $R = 100\,\Omega$
4. $R = 300\,\Omega$

Si definimos

$$q(t) = \int_0^{t'} i(t')dt' \tag{2}$$

derivando la expresión anterior

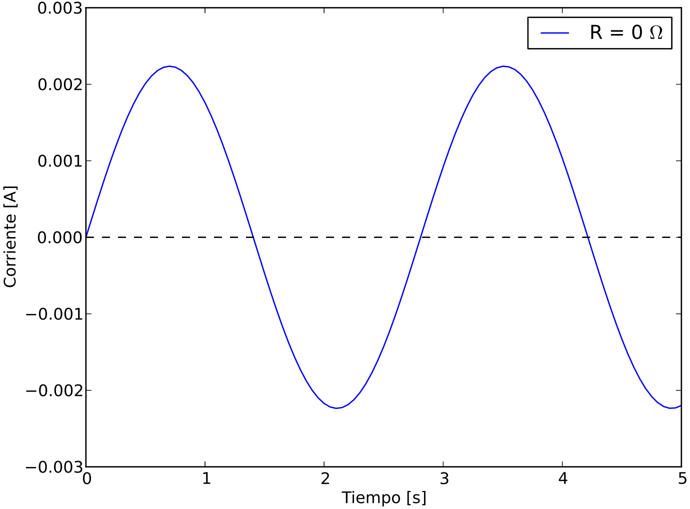
$$\frac{d}{dt}q(t) = i(t), \qquad q(0) = 0 \tag{3}$$

Sustituimos en la ecuació inicial, para re-escribir

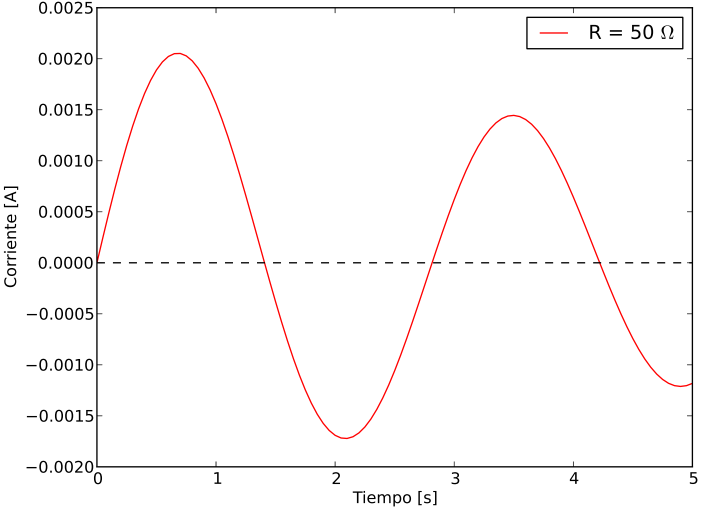
$$\frac{d}{dt}i(t) = -\frac{R}{L}i(t) - \frac{1}{LC}q(t) + \frac{1}{LC}q(0) + \frac{E(t)}{L}, i(0) = 0 \tag{4}$$

La ecuación (1) se transformó en un sistema de dos EDO de primer orden: las ecuaciones (3) y (4).

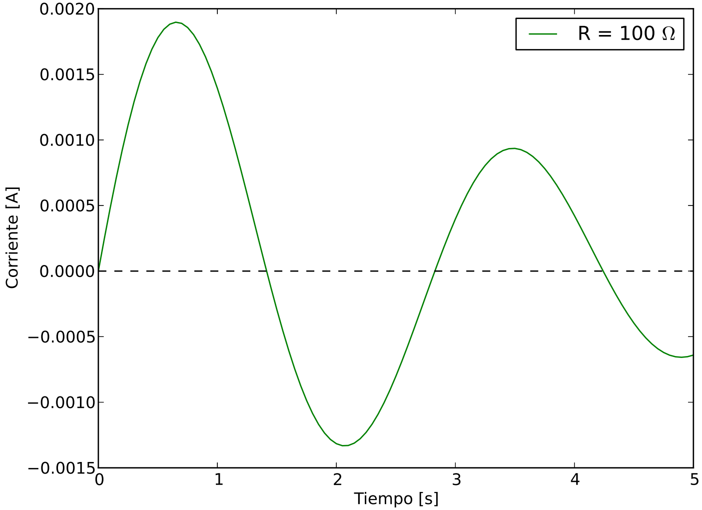
Solución gráfica con $R = 0\Omega$



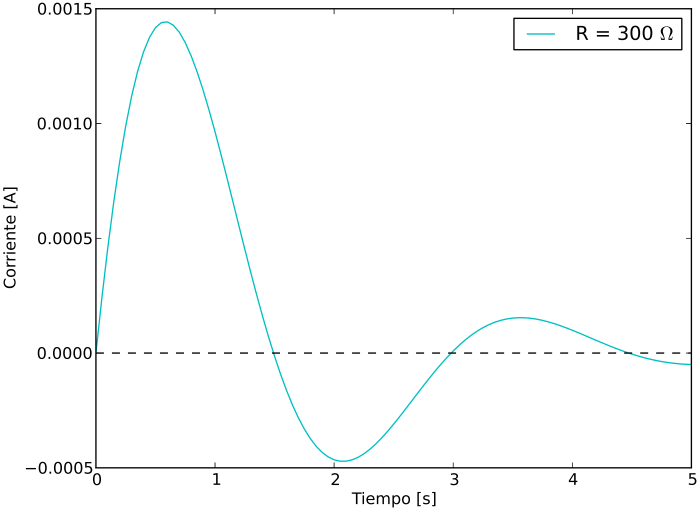
Solución gráfica con $R = 50\Omega$



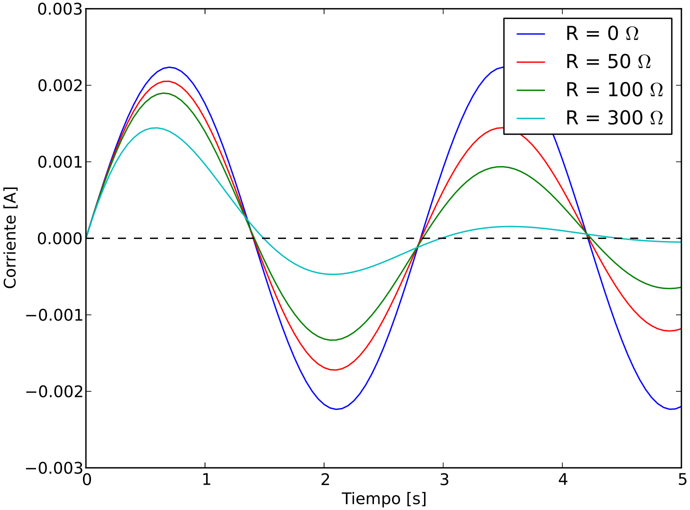
Solución gráfica con $R = 100\Omega$



Solución gráfica con $R = 300\Omega$



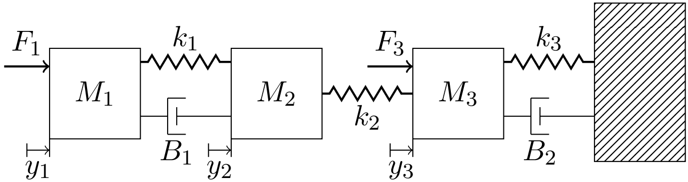
Solución gráfica con valores de R superpuestos



Ejercicio 2

En la figura se muestra un sistema de tres masas. Los desplazamientos de estas tres masas satisfacen las ecuaciones dadas por:

$$
\begin{aligned}
M_1 y_1'' + B_1 y_1' + K_1 y_1 - B_1 y_2' - K_2 y_2 &= F_1(t) \\
-B_1 y_1' - K_1 y_1 + M_2 y_2'' + B_1 y_2' + (K_1 + K_2) y_2 - K_2 y_3 &= 0 \\
-K_2 y_2 + M_3 y_3'' + B_2 y_3' + (K_2 + K_3) y_3 &= F_3(t)
\end{aligned}
$$



Las constantes y condiciones iniciales son

$K_1 = K_2 = K_3 = 1$ (constantes de los resortes, kgm$/s^2$)
$M_1 = M_2 = M_3 = 1$ (masa, kg)
$F_1(t) = 1, F_3(t) = 0$ (fuerza, N)
$B_1 = B_2 = 0.1$ (coeficientes de amortiguamiento, kg/s)

$y_1(0) = y_1'(0) = y_2(0) = y_2'(0) = y_3(0) = y_3'(0) = 0$
(condiciones iniciales)

Resuelve y grafica las ecuaciones anteriores mediante RK4, para $0 \leq t \leq 30$ segundos y $h = 0.1$

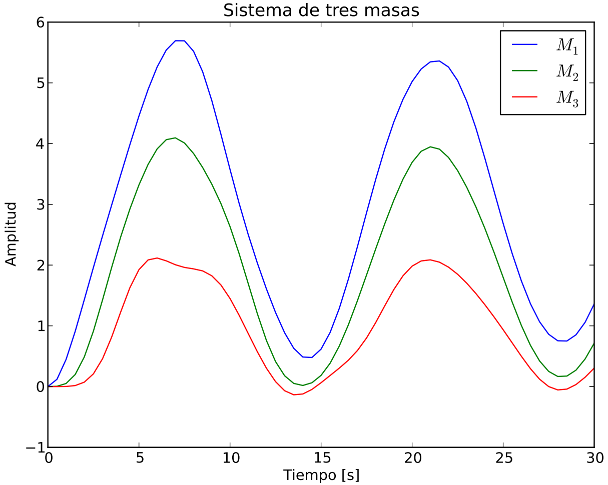
Hint: Definiendo

$$y_4 = y_1', \qquad y_5 = y_2', \qquad y_6 = y_3'$$

La ecuación inicial se escribe como un conjunto de seis EDO de primer orden, de la siguiente manera:

$$\begin{aligned}
y_1' &= y_4 \\
y_2' &= y_5 \\
y_3' &= y_6 \\
y_4' &= \left[-B_1 y_4 - K_1 y_1 + B_1 y_5 + K_2 y_2 + F_1\right]/M_1 \\
y_5' &= \left[B_1 y_4 + K_1 y_1 - B_1 y_5 - \left(K_1 + K_2\right) y_2 + K_2 y_3\right]/M_2 \\
y_6' &= \left[K_2 y_2 - B_2 y_6 - \left(K_2 + K_3\right) y_3 + F_3\right]/M_3
\end{aligned}$$

Solución al problema



Contenido

Sistema Lotka-Volterra

Las ecuaciones de Lotka-Volterra, también son conocidas como ecuaciones de depredador-presa, descrito por un sistema de 2 ecuaciones diferenciales no lineales de primer orden, que se utiliza frecuentemente para describir la dinámica de sistemas biológicos donde interaccionan dos especies, un depredador y una de sus presas.

El sistema evoluciona de acuerdo al par de ecuaciones:

$$\frac{du}{dt} = au - buv$$
$$\frac{dv}{dt} = -cv + dbuv$$

donde:

- u es el número de presas (ej. Conejos)
- v es el número de depredadores (ej. Zorros)
- a es la tasa natural de crecimiento de conejos, sin que haya zorros.
- b es la tasa natural de la muerte de conejos, debido a la depredación.
- c es la tasa natural de la muerte del zorro, cuando no hay conejos.
- d es el factor que describe el número de conejos capturados.

Vamos a utilizar $X = [u, v]$ para describir el estado de las poblaciones.

Definiendo las ecuaciones

```
from numpy import *
import pylab as p

a = 1.
b = 0.1
c = 1.5
d = 0.75

def dXdt(X, t=0):
    return array([ a*X[0] -   b*X[0]*X[1] ,  -
        c*X[1] + d*b*X[0]*X[1] ])
```

Población en equilibrio

Antes de usar Scipy para integrar el sistema, veremos de cerca la posición de equilibrio. El equilibrio ocurre cuando la tasa de crecimiento es igual a 0, lo que nos da dos puntos fijos:

```
X_f0 = array([     0.    ,  0.])
X_f1 = array([ c/(d*b), a/b])
all(dXdt(X_f0) == zeros(2) ) and all(dXdt(X_f1) == zeros(2))
```

Para usar la función `odeint`, hay que definir el parámetro de tiempo t, así como las condiciones inciales de la población: 10 conejos y 5 zorros.

```
t = linspace(0, 15, 1000)

X0 = array([10, 5])

X = integrate.odeint(dXdt, X0, t)
```

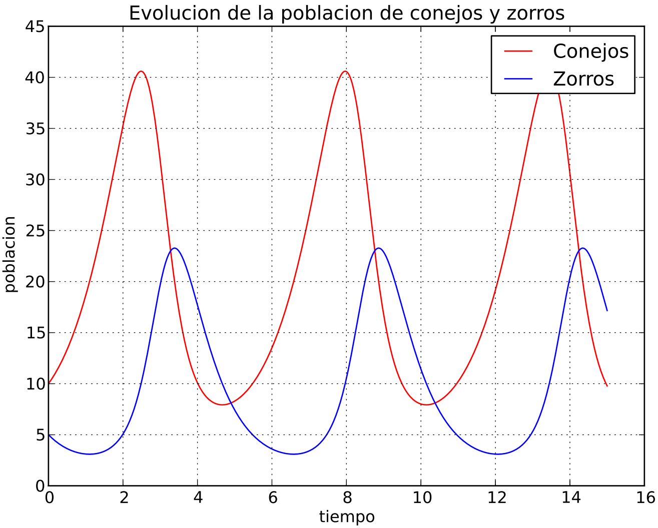
La función `odeint` requiere de tres argumentos: la función (o arreglo de funciones), la condición inicial (o arreglo de condiciones iniciales) y el tiempo.

Una vez obtenido el código para la solución del problema, ahora nos corresponde graficar el conjunto de datos obtenido, para ello, usamos la siguiente rutina en Python:

```python
conejos, zorros = X.T

f1 = p.figure()
p.plot(t, conejos, 'r-', label='Conejos')
p.plot(t, zorros  , 'b-', label='Zorros')
p.grid()
p.legend(loc='best')
p.xlabel('tiempo')
p.ylabel('poblacion')
p.title('Evolucion de la poblacion de conejos y zorros')
p.show()
```

Resultado gráfico



La gráfica anterior nos da la información sobre el número tanto de conejos como de zorros durante el intervalo de tiempo estudiado, es decir, tenemos una especie de "censo".

Para ver la dinámica de las poblaciones propiamente, ahora representamos el espacio fase del sistema, por lo que tenemos que hacer algunos ajustes en el código que usamos anteriormente.

Consideremos las condiciones de equilibrio, es decir, donde la tasa de crecimiento es cero:

```
X_f0 = array([      0.   ,   0.])
X_f1 = array([ c/(d*b), a/b])
```

Dibujaremos el espacio fase con algunos elementos visuales con el fin de decoración nada más.

```
values  = linspace(0.3, 0.9, 5)

vcolors = p.cm.autumn_r(linspace(0.3, 1., len(
    values)))

f2 = p.figure()
```

El módulo `cm` proporciona un gran conjunto de mapas de colores, así como las funciones para crear nuevos mapas de color; existen varios mapas ya definidos: `autumn`, `bone`, `cool`, `copper`, `flag`, `gray`, `hot`, `hsv`, `jet`, `pink`, `prism`, `spring`, `summer`, `winter`, `spectral`.

Se van a dibujar ahora las trayectorias para diferentes condiciones iniciales (número de conejos y zorros)

```
for v, col in zip(values, vcolors):
    X0 = v * X_f1
    X = integrate.odeint( dX_dt, X0, t)
    p.plot( X[:,0], X[:,1], lw=3.5*v, color=
        col, label='X0=(%.f, %.f)' % ( X0[0],
        X0[1]) )
```

La función `zip` sirve para reorganizar las listas en Python. Como parámetros admite un conjunto de listas. Lo que realmente hace es tomar el elemento i-ésimo elemento de cada lista y los une en una tupla, después une todas las tuplas en una lista.

En cada gráfica se modifica el grosor de la línea y el color que se le asocia.

Se define una malla y se calcula la dirección

```
ymax = p.ylim(ymin=0)[1]
xmax = p.xlim(xmin=0)[1]
nb_points    = 20

x = linspace(0, xmax, nb_points)
y = linspace(0, ymax, nb_points)
```

```
X1 , Y1 = meshgrid(x, y)
DX1, DY1 = dX_dt([X1, Y1])
M = (hypot(DX1, DY1))
M[ M == 0] = 1.
DX1 /= M
DY1 /= M
```

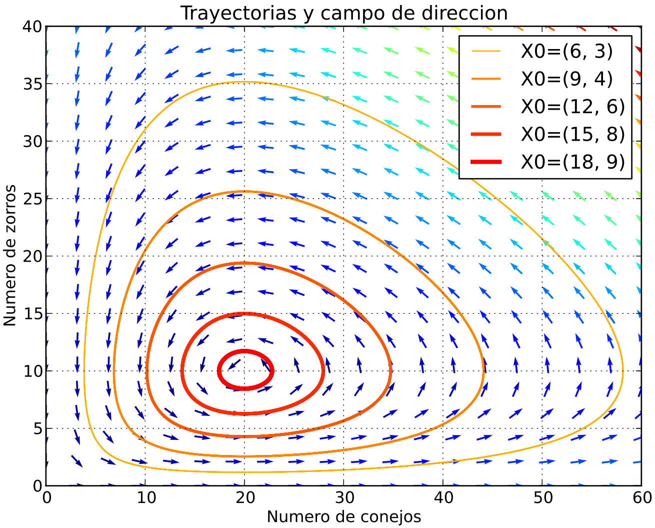
Con X1 y Y1 se crea una malla, con DX1 y DY1 se calcula el crecimiento de las poblaciones en la malla, M calcula la norma de la tasa de crecimiento, para ello, usa la funciónhypot, M[M==0] =1. evita que hagamos una división entre cero, DX1/M y DY1/M normaliza cada vector.

Se dibujan las direcciones usando `quiver`

```
p.title('Trayectorias y campo de direccion')
Q = p.quiver(X1, Y1, DX1, DY1, M, pivot='mid',
    cmap=p.cm.jet)
p.xlabel('Numero de conejos')
p.ylabel('Numero de zorros')
p.legend()
p.grid()
p.xlim(0, xmax)
p.ylim(0, ymax)
p.show()
```

La función `quiver` genera el mapa vectorial, requiere de cinco argumentos: las posiciones X1,Y1 de inicio, el valor de las componentes del vector DX1, DY1 y el color asociado, el argumento `pivot` indica en qué parte de la malla se va a colocar el vector.

Resultado gráfico



Ejercicio para resolver

El modelo de Lorenz se usa para estudiar la formación de torbellinos en la atmósfera, aunque abordó el problema de manera general, estableció las bases para el estudio de sistemas dinámicos, el conjunto de ecuaciones está dado por

$$\frac{dy_1}{dt} = a(y_2 - y_1)$$
$$\frac{dy_2}{dt} = (b - y_3)y_1 - y_2$$
$$\frac{dy_3}{dt} = y_1 y_2 - c y_3$$

en el modelo, a, b y c son parámetros positivos. Resuelve este modelo numéricamente, grafica la solución. Utiliza $a = 10$, $b = 28$ y $c = 8/3$.

Resultado gráfico

Usando $a = 10$, $b = 28$ y $c = 8/3$