Tarea 2 - Sistemas con retardos en la entrada

Roberto Cadena Vega

19 de enero de 2015

```
In [1]: # Se importan librerias para graficar, y se define un estilo especifico
        %matplotlib inline
        from matplotlib.pyplot import plot, figure, style
        style.use("ggplot")
```

1. Tarea 2 - Simulación de sistema simple con retardos

1.0.1. Problema

Dado el sistema:

$$\dot{x} = -3x(t) + x(t-1) + x(t-2)$$

Simular el sistema con las siguientes condiciones iniciales φ:

1. $\varphi(t) = 1 \quad t \in [-2, 0]$
2. $\varphi(t) = \sin(t) \quad t \in [-2, 0]$
3. $\varphi(t) = \begin{cases} 1 & t = 0 \\ 0 & t \neq 0 \end{cases} \quad t \in [-2, 0]$

1.0.2. Solución

Para simular y gráficar este sistema, tenemos que definir una función la cual sea capaz de comportarse como el sistema descrito, esto lo logramos con el código de MATLAB descrito en el archivo `f.m`, el cual podemos ver a continuación:

```
In [56]: cat MATLAB/f.m
```

```
function dydt = f(t, y, Z)
    yret1 = Z(:, 1);
    yret2 = Z(:, 2);

    dydt = -3*y + yret1(1) + yret2(1);
end
```

Esta función se puede analizar por lineas, la primer linea

```
function dydt = f(t, y, Z)
```

tan solo describe la función `f`, con variables de entrada, `t`, `y`, `Z` y una salida `dydt`. En esta función `t` es el tiempo para el que queremos calcular el valor de nuestra función; en especifico para nuestro ejemplo, como nuestro sistema es invariante en el tiempo, no necesitamos incluirla dentro de los calculos, `y` es el valor de nuestra variable, el cual será aproximado por la función `dde23` y `Z` es el valor de la variable del sistema, con un retardo aplicado.

Las siguientes lineas tan solo guardan los valores de la variable del sistema en variables con un nombre significativo:

```
yret1 = Z(:, 1);
yret2 = Z(:, 2);
```

Y la ultima linea calcula el valor de la salida del sistema, para nosostros $\dot{x}$,

```
dydt = -3*y + yret1(1) + yret2(1);
```

Sin embargo, en este punto, nuestro modelo computacional se acerca mas bien a:

$$\dot{x} = -3x(t) + x(t - \tau_1) + x(t - \tau_2)$$

y los valores de los retrasos τ_1 y τ_2 los ingresaremos en la función `dde23`.

Para poder simular este sistema, ahora tenemos que introducir la función que define el comportamiento de nuestro sistema, en la función `dde23`, lo cual se verá así:

```
sol0 = dde23(@f, [1, 2], @phi0, [0, 5])
```

En donde `@fes` la sintaxis para pasar como argumento a la función `f`, `[1, 2]` es el conjunto de los retardos a utilizar, `@phi0` es la función que define a las condiciones iniciales y `[0, 5]` es el intervalo de simulación.

Una vez que hemos simulado, la variable `sol0` es una estructura que contiene a las variables asociadas a la simulación, de donde podemos obtener `x` y `y`, las cuales corresponden a t y x.

Ahora importamos los resultados de la simulación en MATLAB:

```
In [3]: # Se importa libreria de entrada y salida y se importan datos de simulacion en MATLAB
        import scipy.io
        sistemaretardos = scipy.io.loadmat('./MATLAB/sistemaretardos.mat')

In [5]: # Se importan datos de simulación a variables de Python para poder manipularlas
        x0 = sistemaretardos.get("x0")
        x1 = sistemaretardos.get("x1")
        x2 = sistemaretardos.get("x2")
        y0 = sistemaretardos.get("y0")
        y1 = sistemaretardos.get("y1")
        y2 = sistemaretardos.get("y2")
```

Pero aun no hemos hablado de las funciones ϕ, las cuales deben ser definidas de la misma manera que la función que nos da el comportamiento del sistema, por lo que para la primer condición inicial:

$$\varphi(t) = 1 \quad t \in [-2, 0]$$

nos quedará:

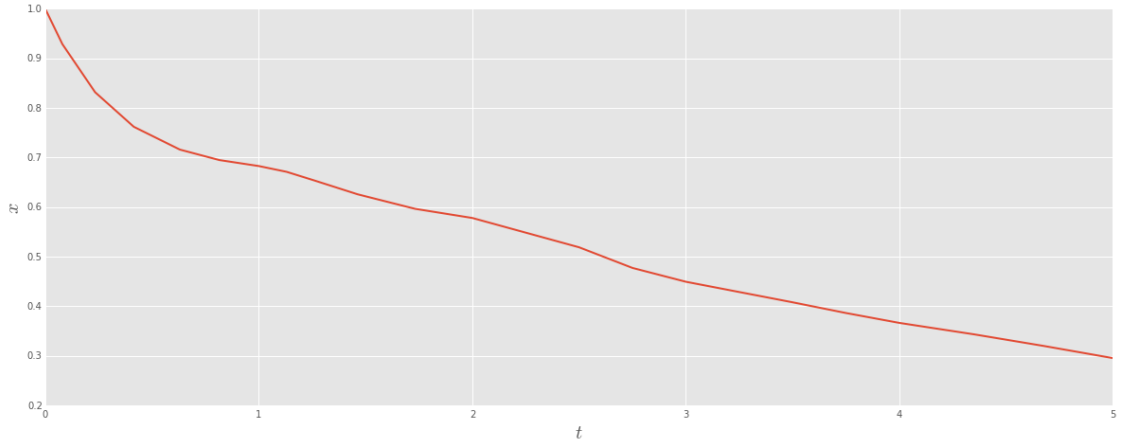
```
In [49]: cat MATLAB/phi0.m

function y = phi0(t)
    y = 1;
end
```

y por lo tanto, para esta condición inicial, el comportamiento del sistema nos queda:

```
In [61]: f = figure(figsize=(18, 7))
         plot(x0[0], y0[0])

         ax = f.gca()
         ax.set_xlabel(r'$t$', fontsize=20)
         ax.set_ylabel(r'$x$', fontsize=20);
```



De la misma manera, para la segunda condición inicial:

$$\varphi(t) = \sin(t) \quad t \in [-2, 0]$$

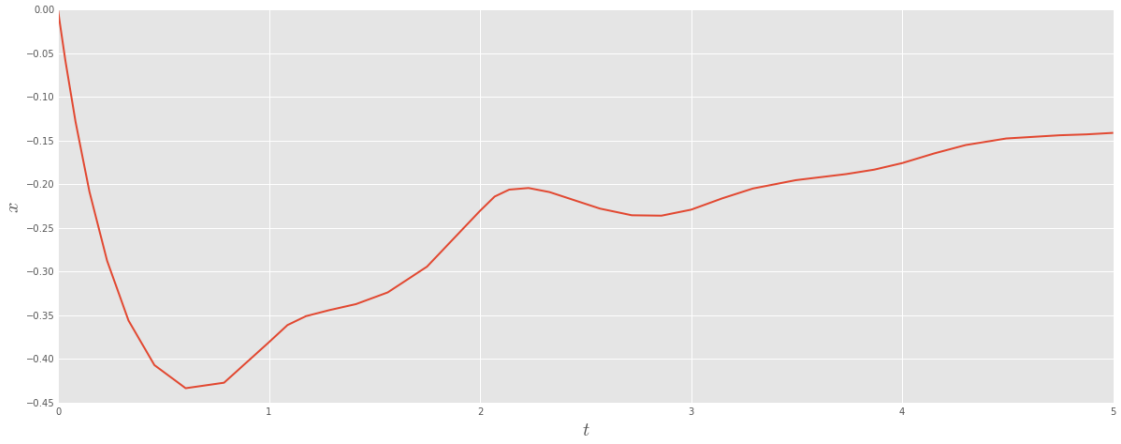
tenemos que la función `phi1` nos quedará:

```
In [50]: cat MATLAB/phi1.m
```

```
function y = phi1(t)
    y = sin(t);
end
```

```
In [62]: f = figure(figsize=(18, 7))
         plot(x1[0], y1[0])

         ax = f.gca()
         ax.set_xlabel(r'$t$', fontsize=20)
         ax.set_ylabel(r'$x$', fontsize=20);
```



Y para la tercer condición inicial:

$$\varphi(t) = \begin{cases} 1 & t = 0 \\ 0 & t \neq 0 \end{cases} \quad t \in [-2, 0]$$

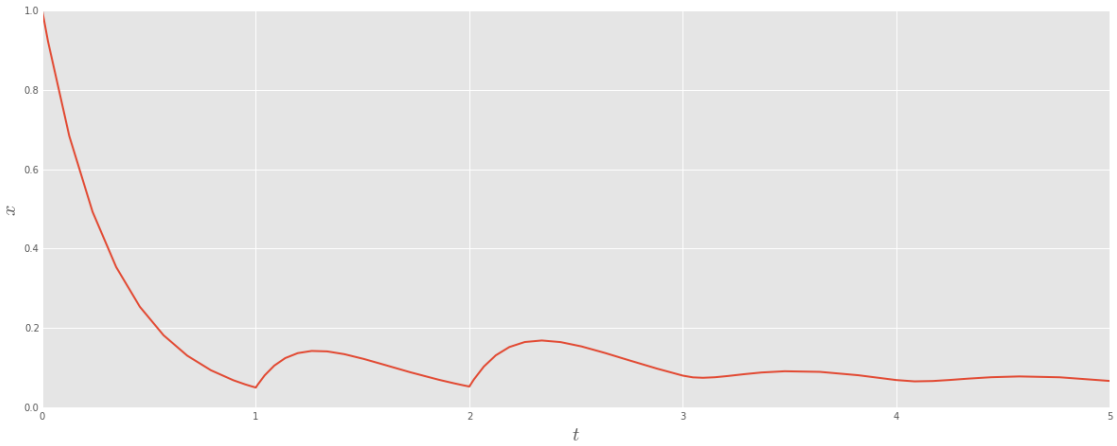
tenemos:

```
In [55]: cat MATLAB/phi2.m

function y = phi2(t)
    if t == 0
        y = 1;
    else
        y = 0;
    end
end
```

```
In [63]: f = figure(figsize=(18, 7))
         plot(x2[0], y2[0])

         ax = f.gca()
         ax.set_xlabel(r'$t$', fontsize=20)
         ax.set_ylabel(r'$x$', fontsize=20);
```



Puedes acceder a este notebook a traves de la página
http://nbviewer.ipython.org/github/robblack007/DCA/blob/master/IPythonNotebooks/Sistemas%20con%20retardo%20en%20la%20entrada/Tarea%202.ipynb
o escaneando el siguiente código:

In [12]:
```
# Codigo para generar codigo :)
from qrcode import make
img = make("http://nbviewer.ipython.org/github/robblack007/DCA/blob/master/IPythonNotebooks/Si
img.save("codigos/prueba.jpg")
```