Tarea 6

Roberto Cadena Vega

10 de febrero de 2015

Simulación de un sistema con retardo por medio de la construcción de una Matriz de Lyapunov

Dado el sistema:

$$\dot{x}(t) = A_0 x(t) + A_1 x(t-\tau)$$

podemos obtener la solución $x(t)$, por medio de la construcción de la matriz de Lyapunov y el calculo de la siguiente ecuación matricial:

$$\begin{pmatrix} \bar{Y}(\tau) \\ \bar{Z}(\tau) \end{pmatrix} = e^{L\tau} \left(M + N e^{L\tau} \right)^{-1} \begin{pmatrix} \bar{0} \\ \bar{W} \end{pmatrix}$$

en donde $\bar{W}$ es la vectorización de la matriz W, y las matrices L, M y N son de la forma:

$$L = \begin{pmatrix} A_0^T \otimes I & A_1^T \otimes I \\ -I \otimes A_1^T & -I \otimes A_0^T \end{pmatrix}$$

$$M = \begin{pmatrix} I \otimes I & 0 \\ A_0^T \otimes I + I \otimes A_0^T & A_1^T \otimes I \end{pmatrix}$$

$$N = \begin{pmatrix} 0 & -I \otimes I \\ I \otimes A_1^T & 0 \end{pmatrix}$$

en donde $A \otimes B$ es el producto de Kronecker de A y B.
Para esta simulación utilizaremos los siguientes valores:

$$A_0 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \quad A_1 = \begin{pmatrix} -2 & 0 \\ 0{,}3 & 0 \end{pmatrix} \quad W = I \quad \tau \in [0, 1]$$

Declaramos estas matrices, asi como una matriz de ceros, la identidad y las vectorizaciones de una matriz de ceros, asi como la de la matriz W.

```
In [1]: from numpy import array, matrix, kron, eye, zeros, vstack, hstack, linspace
        from scipy.linalg import expm
```

```
In [2]: A0 = matrix([[0, 1], [-1, 0]])
        A0
```

```
Out[2]: matrix([[ 0,  1],
                [-1,  0]])
```

```
In [3]: A1 = matrix([[-2, 0], [0.3, 0]])
        A1
```

```
Out[3]: matrix([[-2. ,  0. ],
                [ 0.3,  0. ]])
```

```
In [4]: I = matrix(eye(2))
        I

Out[4]: matrix([[ 1.,  0.],
                [ 0.,  1.]])

In [5]: cero = zeros((2**2, 2**2))
        cero

Out[5]: array([[ 0.,  0.,  0.,  0.],
               [ 0.,  0.,  0.,  0.],
               [ 0.,  0.,  0.,  0.],
               [ 0.,  0.,  0.,  0.]])

In [6]: v0 = matrix(zeros(4))
        v0

Out[6]: matrix([[ 0.,  0.,  0.,  0.]])

In [7]: W = I
        W.flatten("F")

Out[7]: matrix([[ 1.,  0.,  0.,  1.]])
```

Ahora calculamos la matriz de Lyapunov, asi como las matrices M y N

```
In [8]: L = vstack((hstack((kron(A0.T, I), kron(A1.T, I))),
                hstack((kron(-I, A1.T), kron(-I, A0.T)))))
        L

Out[8]: matrix([[ 0. ,  0. , -1. , -0. , -2. , -0. ,  0.3,  0. ],
                [ 0. ,  0. , -0. , -1. , -0. , -2. ,  0. ,  0.3],
                [ 1. ,  0. ,  0. ,  0. ,  0. ,  0. ,  0. ,  0. ],
                [ 0. ,  1. ,  0. ,  0. ,  0. ,  0. ,  0. ,  0. ],
                [ 2. , -0.3,  0. , -0. , -0. ,  1. , -0. ,  0. ],
                [-0. , -0. , -0. , -0. , -1. , -0. , -0. , -0. ],
                [ 0. , -0. ,  2. , -0.3, -0. ,  0. , -0. ,  1. ],
                [-0. , -0. , -0. , -0. , -0. , -0. , -1. , -0. ]])

In [9]: M = vstack((hstack((kron(I, I), cero)),
                    hstack((kron(A0.T, I) + kron(I, A0.T), kron(A1.T, I)))))
        M

Out[9]: matrix([[ 1. ,  0. ,  0. ,  0. ,  0. ,  0. ,  0. ,  0. ],
                [ 0. ,  1. ,  0. ,  0. ,  0. ,  0. ,  0. ,  0. ],
                [ 0. ,  0. ,  1. ,  0. ,  0. ,  0. ,  0. ,  0. ],
                [ 0. ,  0. ,  0. ,  1. ,  0. ,  0. ,  0. ,  0. ],
                [ 0. , -1. , -1. , -0. , -2. , -0. ,  0.3,  0. ],
                [ 1. ,  0. ,  0. , -1. , -0. , -2. ,  0. ,  0.3],
                [ 1. ,  0. ,  0. , -1. ,  0. ,  0. ,  0. ,  0. ],
                [ 0. ,  1. ,  1. ,  0. ,  0. ,  0. ,  0. ,  0. ]])

In [10]: N = vstack((hstack((cero, kron(-I, I))),
                     hstack((kron(I, A1.T), cero))))
         N
```

```
Out[10]: matrix([[ 0. ,  0. ,  0. ,  0. , -1. , -0. , -0. , -0. ],
                 [ 0. ,  0. ,  0. ,  0. , -0. , -1. , -0. , -0. ],
                 [ 0. ,  0. ,  0. ,  0. , -0. , -0. , -1. , -0. ],
                 [ 0. ,  0. ,  0. ,  0. , -0. , -0. , -0. , -1. ],
                 [-2. ,  0.3, -0. ,  0. ,  0. ,  0. ,  0. ,  0. ],
                 [ 0. ,  0. ,  0. ,  0. ,  0. ,  0. ,  0. ,  0. ],
                 [-0. ,  0. , -2. ,  0.3,  0. ,  0. ,  0. ,  0. ],
                 [ 0. ,  0. ,  0. ,  0. ,  0. ,  0. ,  0. ,  0. ]])
```

```
In [11]: v = vstack((v0.T, W.flatten("F").T))
         v
```

```
Out[11]: matrix([[ 0.],
                 [ 0.],
                 [ 0.],
                 [ 0.],
                 [ 1.],
                 [ 0.],
                 [ 0.],
                 [ 1.]])
```

Ahora declaramos la función $f(\tau)$ la cual es de la forma:

$$f(\tau) := e^{L\tau}\left(M + Ne^{L\tau}\right)^{-1}\begin{pmatrix}\bar{0}\\ \bar{W}\end{pmatrix}$$

con la que podremos calcular los valores de:

$$\begin{pmatrix}\bar{Y}(\tau)\\ \bar{Z}(\tau)\end{pmatrix}$$

para cada τ que le demos como argumento:

```
In [12]: f = lambda tau: (expm(L*tau)*(M + N*expm(L*tau)).I*v).flatten().tolist()[0]
```

utilizamos $h = \frac{1}{2}$ para probar nuestra función:

```
In [13]: # Fijamos la precision de los datos mostrados a 3 cifras significativas
         # para facilitar la lectura

         %precision 3
```

```
Out[13]: u'%.3f'
```

```
In [14]: h = 0.5
         v1 = f(h)
         v1
```

```
Out[14]: [-0.373, 0.844, 0.034, -1.434, -1.303, 0.500, 0.500, -1.800]
```

y para un incremento en el retardo de $\frac{1}{10}$, tenemos:

```
In [15]: v2 = f(h + 0.1)
         v2
```

```
Out[15]: [-0.280, 1.465, -0.639, -1.317, -2.888, 0.500, 0.500, -2.005]
```

por lo que solo queda declarar el dominio de definición de los retardos:

$$T = [0, 1]$$

y obtener el conjunto de soluciones para cada retardo en el dominio definido:

$$V_{sol} = \{f(t) \mid t \in T\}$$

In [16]:
```
T = linspace(0, 1, 1000)
vsol = [f(t) for t in T]
```

In [17]:
```
sol = zip(*vsol)
```

Ahora solo queda graficar la soluciónes contra los retardos del sistema.

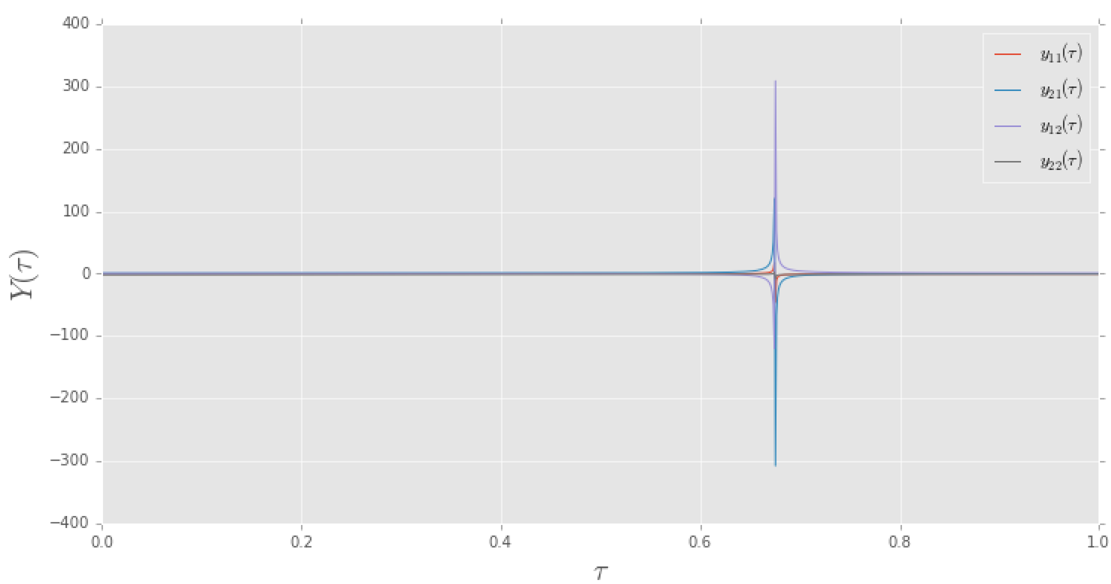
In [18]:
```
%matplotlib inline
from matplotlib.pyplot import plot, figure, style, legend
style.use("ggplot")
```

In [19]:
```
fig = figure(figsize=(12, 6))

p1, = plot(T, sol[0])
p2, = plot(T, sol[1])
p3, = plot(T, sol[2])
p4, = plot(T, sol[3])

ax = fig.gca()
ax.set_ylabel(r"$Y(\tau)$", fontsize=20)
ax.set_xlabel(r"$\tau$", fontsize=20)

legend([p1, p2, p3, p4],
       [r"$y_{11}(\tau)$",
        r"$y_{21}(\tau)$",
        r"$y_{12}(\tau)$",
        r"$y_{22}(\tau)$"]);
```



Puedes acceder a este notebook a traves de la página
http://bit.ly/1CccnJ9
o escaneando el siguiente código: