Tarea 8

Roberto Cadena Vega

24 de febrero de 2015

Tarea 8

Transformada de Laplace de una integral de convolución

Queremos obtener:

$$\mathcal{L}\left\{\int_{-\tau}^{0} G(\theta)x(\tau+\theta)d\theta\right\}$$

Podemos definir la integral de convolución para dos funciones continuas $f(t)$ y $g(t)$, tales que la convolución $(f*g)(t)$ es de la forma:

$$(f*g)(t) = \int_0^t f(t-\tau)g(\tau)d\tau$$

y la transformada de Laplace de esta integral de convolución es:

$$\mathcal{L}\{(f*g)(t)\} = F(s)G(s)$$

por lo que nos interesa poner nuestra integral original de manera que sea una integral de convolución.

Podemos ver que al aplicar el cambio de variable $\delta = \theta + \tau$, cuando θ variaba de $-\tau \to 0$, δ variará de $0 \to \tau$ y el diferencial $d\theta$ es equivalente a $d\delta$, por lo que nuestra integral quedará:

$$\mathcal{L}\left\{\int_0^{\tau} G(\delta-\tau)x(\tau+\delta-\tau)d\delta\right\} = \mathcal{L}\left\{\int_0^{\tau} G(\delta-\tau)x(\delta)d\delta\right\}$$

y etsa integral ya es de la forma de la integral de convolución, por lo que podemos escribirla como:

$$\mathcal{L}\{(G*x)(t)\} = G(s)x(s)$$

Diseño de un controlador predictivo para un sistema con retardo en la entrada

Dado el sistema con retardo en la entrada:

$$\dot{x}(t) = Ax(t) + Bu(t-h)$$

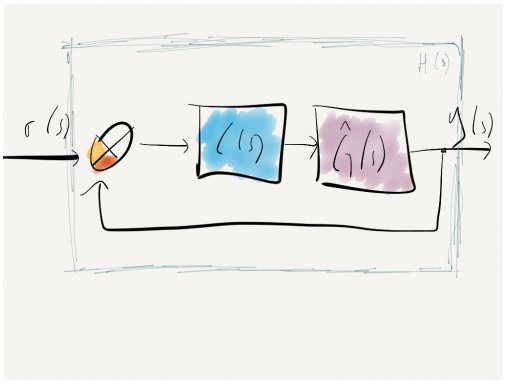
$$y(t) = Cx(t)$$

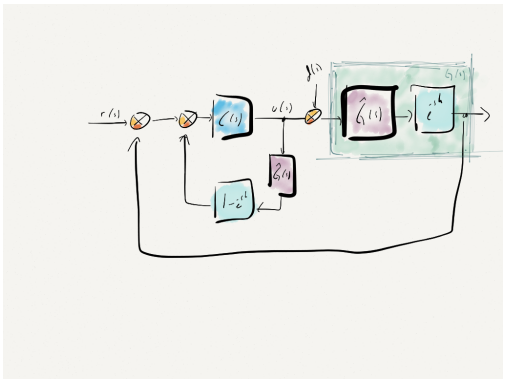
Empezaremos creando un modelo sin retardo para el diseño del controlador predictor de Smith. Si logramos diseñar un controlador $C(s)$ que estabilice de manera aceptable al sistema sin retardos, podremos diseñar otro $\tilde{C}(s)$ que tenga el mismo comportamiento agregado el efecto del retardo.

en donde $\tilde{C}(s)$ será de la forma:

es decir:

$$\tilde{C}(s) = \frac{C(s)}{1 + C(s)\hat{G}(s)\left(1 - e^{-sh}\right)}$$





en donde $\hat{G}(s)$ es la parte de la función de transferencia del sistema que no incluye al retardo. Por lo que nuestra primera tarea es encontrar la función de transferencia para nuestro sistema sin el efecto del retardo.

Consideremos pues el sistema

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t)$$

para el cual la función de transferencia se puede obtener por la siguiente ecuación:

$$\hat{G}(s) = C\left(sI - A\right)^{-1} B$$

cabe mencionar que la función de transferencia para el sistema con retardo es:

$$G(s) = C\left(sI - A\right)^{-1} Be^{-sh}$$

por lo que en efecto $\hat{G}(s)$ es la parte de la función de transferencia del sistema sin el efecto del retardo:

$$G(s) = \hat{G}(s)e^{-sh}$$

```
In [1]: # Se importan las librerias para calculo simbolico
        from IPython.display import display

        from sympy import var, simplify, collect, expand, solve, sin, cos, Matrix, eye, diff, Function,
        from sympy.physics.mechanics import mlatex, mechanics_printing
        mechanics_printing()
```

```
In [2]: %matplotlib inline
        from matplotlib.pyplot import plot, style, figure, legend
        style.use("ggplot")
```

```
In [3]: from control import tf, step, pade, acker, ss, feedback
        from numpy import linspace, matrix, eye

In [4]: var("s")
```

Out[4]:

$$s$$

Empezaremos con el sistema $A_1 = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$, $B_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, $C_1 = \begin{pmatrix} 1 & 0 \end{pmatrix}$ y $h = 1$.

```
In [5]: A1 = Matrix([[0, 1], [0, 0]])
        B1 = Matrix([[0], [1]])
        C1 = Matrix([[1, 0]])

        G1 = (C1*(s*eye(2) - A1).inv()*B1)[0]
        G1
```

Out[5]:

$$\frac{1{,}0}{s^2}$$

```
In [6]: t = linspace(0, 10, 100)
        t.max()
```

Out[6]:

$$10{,}0$$

Podemos notar que la función de transferencia de este sistema es:

$$G(s) = \frac{1}{s^2}$$

```
In [7]: # Convertimos las matrices simbolicas en matrices de tipo numerico
        A1 = matrix(A1.tolist(), dtype=float)
        B1 = matrix(B1.tolist(), dtype=float)
        C1 = matrix(C1.tolist(), dtype=float)
```

Para obtener dos polos en -1 podemos usar la función `acker()` para obtener las ganancias de nuestro controlador PD, el cual utilizaremos para estabilizar nuestro sistema.

```
In [8]: k1 = acker(A1, B1, (-1, -1))
        k1
```

Out[8]: `matrix([[ 1.,  2.]])`

Definimos una función que calculará todos los controladores que deseamos, segun la ecuación que ya dimos:

$$\tilde{C}(s) = \frac{C(s)}{1 + C(s)\hat{G}(s)\left(1 - e^{-sh}\right)}$$

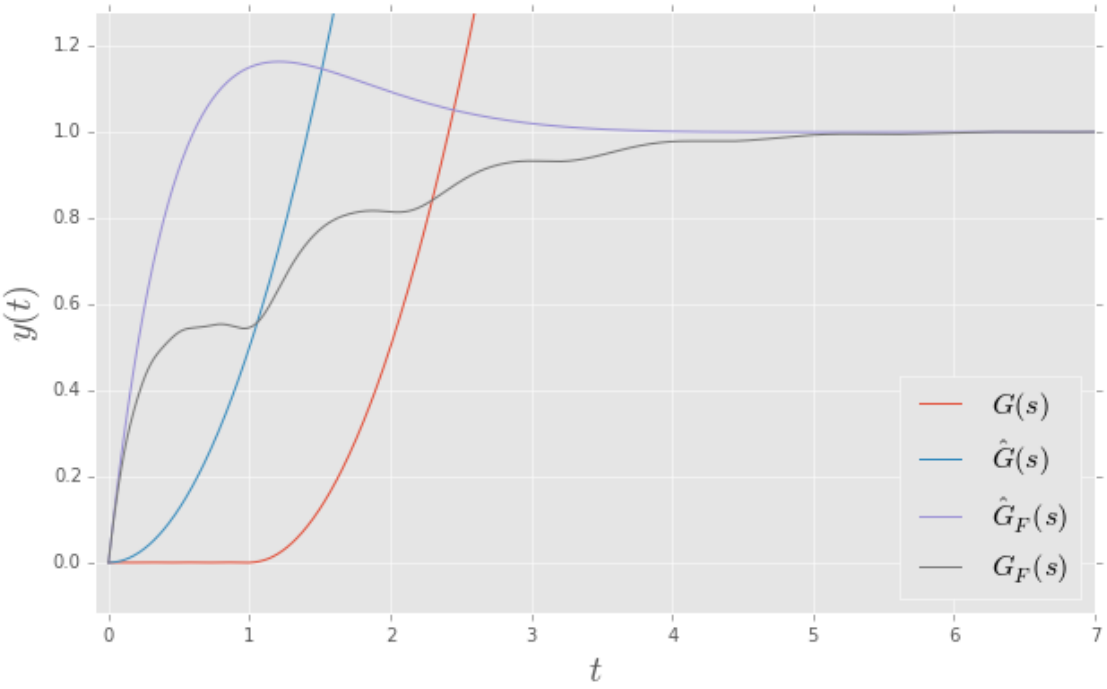
Al final gráfica la respuesta del sistema con y sin efecto del retardo, asi como con y sin controlador.

In [9]:
```
def smith_predictor(A, B, C, D, k, tau=1, t=(0, 10)):
    '''Predictor de Smith

    Esta función toma los arreglos A, B, C, D de un sistema con retardo en la entrada
    y crea un controlador PD con los valores de k = [kp, kd] que estabiliza al sistema
    sin retardos. Ademas crea un controlador que estabiliza al sistema con retardo,
    dada la condición de que este sistema sea estable, y grafica la salida de los
    con y sin realimentación y con y sin retardo tau del tiempo t0 al tiempo t1
    especificados en t = (t0, t1).

    Ejemplo
    -------
    >>> A1 = [[0, 1], [0, 0]]
    >>> B1 = [[0], [1]]
    >>> C1 = [[1, 0]]
    >>> D1 = 0
    >>> tau1 = 1
    >>> t = (0, 10)
    >>> smith_predictor(A1, B1, C1, D1, [1, 2], tau, t)
    '''
    from control import ss, tf, pade, step, feedback
    from numpy import linspace
    from matplotlib.pyplot import figure, plot, legend

    kp, kd = k
    ts = linspace(t[0], t[1], 1000)

    sis = ss(A, B, C, D)
    cont = tf([kd, kp], [0, 1])

    num, den = pade(T=tau, n=10)
    delay = tf(num, den)

    delcont = ((1 - delay)*sis)

    y, t1 = step(sis, ts)
    yd, td = step(sis*delay, ts)
    ycont, tcont = step((cont*sis).feedback(), ts)
    ycontdel, tcontdel = step(feedback(feedback(cont, delcont)*sis), ts)

    f = figure(figsize=(10, 6))

    p1, = plot(td, yd)
    p2, = plot(t1, y)
    p3, = plot(tcont, ycont)
    p4, = plot(tcontdel, ycontdel)

    ax = f.gca()
    ax.set_xlim(t[0] - 0.1, t[1])
    ax.set_ylim(-0.10*ycont.max(), 1.1*ycont.max())

    ax.set_ylabel(r"$y(t)$", fontsize=20)
    ax.set_xlabel(r"$t$", fontsize=20)
```

```
        legend([p1, p2, p3, p4],
               [r"$G(s)$",
                r"$\hat{G}(s)$",
                r"$\hat{G}_F(s)$",
                r"$G_F(s)$"], loc=4, fontsize=16);
```

Por lo que si le damos nuestro sistema con las ganancias deseadas, asi como el tiempo para el que nos interesa, y el retardo, tendremos:

```
In [17]: smith_predictor(A1, B1, C1, 0, k=[3, 3], t=(0, 7), tau=1)
```



Ahora definimos el sistema $A_2 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$, $B_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, $C_2 = \begin{pmatrix} 1 & 0 \end{pmatrix}$ y $h = 1$.

```
In [11]: A2 = Matrix([[0, 1], [-1, 0]])
         B2 = Matrix([[0], [1]])
         C2 = Matrix([[1, 0]])

         G2 = (C2*(s*eye(2) - A2).inv()*B2)[0]
         G2
```

```
Out[11]:
```

$$\frac{1{,}0}{s\left(1{,}0s + \frac{1{,}0}{s}\right)}$$

Por lo que la función de transferencia de este sistema es:

$$G(s) = \frac{1}{s^2 + 1}$$

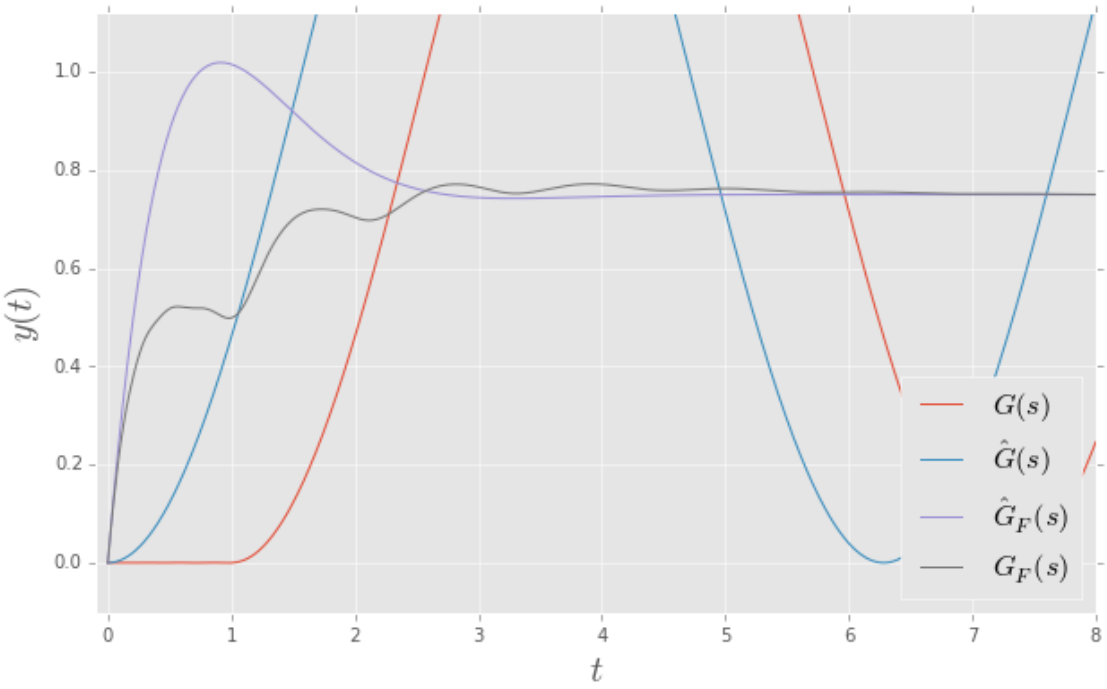
```
In [12]: # Convertimos las matrices simbolicas en matrices de tipo numerico
         A2 = matrix(A2.tolist(), dtype=float)
         B2 = matrix(B2.tolist(), dtype=float)
         C2 = matrix(C2.tolist(), dtype=float)
```

Para obtener dos polos en -1 podemos usar la función `acker()` para obtener las ganancias de nuestro controlador PD.

```
In [13]: k2 = acker(A2, B2, (-1, -1))
         k2

Out[13]: matrix([[ 0.,  2.]])
```

Y la respuesta del sistema es:

```
In [18]: smith_predictor(A2, B2, C2, 0, k=[3, 3], t=(0, 8))
```



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