

Lab5: Implementación de TF discreta.

Por:
Ian Gabriel Cañas Fernández, 1092228

Profesor: Amín Deschamps,
Asignatura: INL365L, Secc 01

Resumen:

En el presente experimento se estará trabajando con un tiempo discreto para así poder reconocer su comportamiento considerando que en ciertos sistemas de control conviene trabajar con elementos digitales y un tiempo de muestreo. Estaremos analizando dos funciones de transferencia y transformándola a tiempos discretos a varias frecuencias de muestreo.

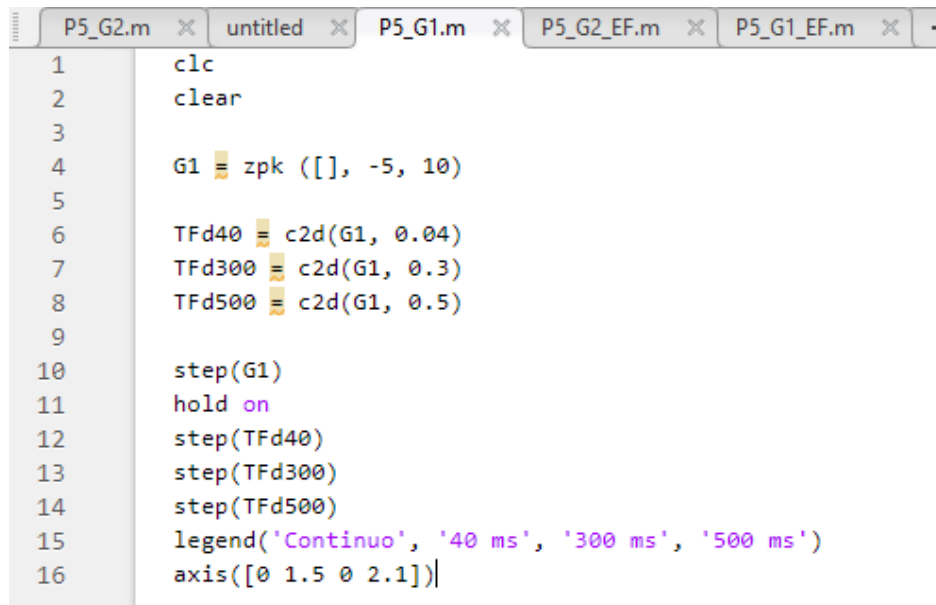
Ejercicios previos:

Se solicita utilizar la función `c2d` para obtener las funciones de transferencias de las plantas dadas en el plano Z a diferentes tiempos de muestreo: 40 ms, 300 ms y 500 ms.

P5.1 Control discreto de planta G1.

Mediante el uso de la función de `c2d` (visto en la ilustración 5.1.1) y mostrando su respuesta a step en la misma gráfica, se obtienen las funciones presentadas en la ilustración 5.1.2 y su respectiva respuesta vista en la ilustración 5.1.3.

$$G1 = \frac{10}{s + 5}$$



```
1  clc
2  clear
3
4  G1 = zpk([], -5, 10)
5
6  TFd40 = c2d(G1, 0.04)
7  TFd300 = c2d(G1, 0.3)
8  TFd500 = c2d(G1, 0.5)
9
10 step(G1)
11 hold on
12 step(TFd40)
13 step(TFd300)
14 step(TFd500)
15 legend('Continuo', '40 ms', '300 ms', '500 ms')
16 axis([0 1.5 0 2.1])
```

Ilustración 5.1.1. Código para obtención de TF discreto para G1.

```

TFd40 =

    0.36254
    -----
    (z-0.8187)

Sample time: 0.04 seconds
Discrete-time zero/pole/gain model.

TFd300 =

    1.5537
    -----
    (z-0.2231)

Sample time: 0.3 seconds
Discrete-time zero/pole/gain model.

TFd500 =

    1.8358
    -----
    (z-0.08208)

Sample time: 0.5 seconds
Discrete-time zero/pole/gain model.

```

Ilustración 5.1.2. Funciones de transferencia discretas para G1.

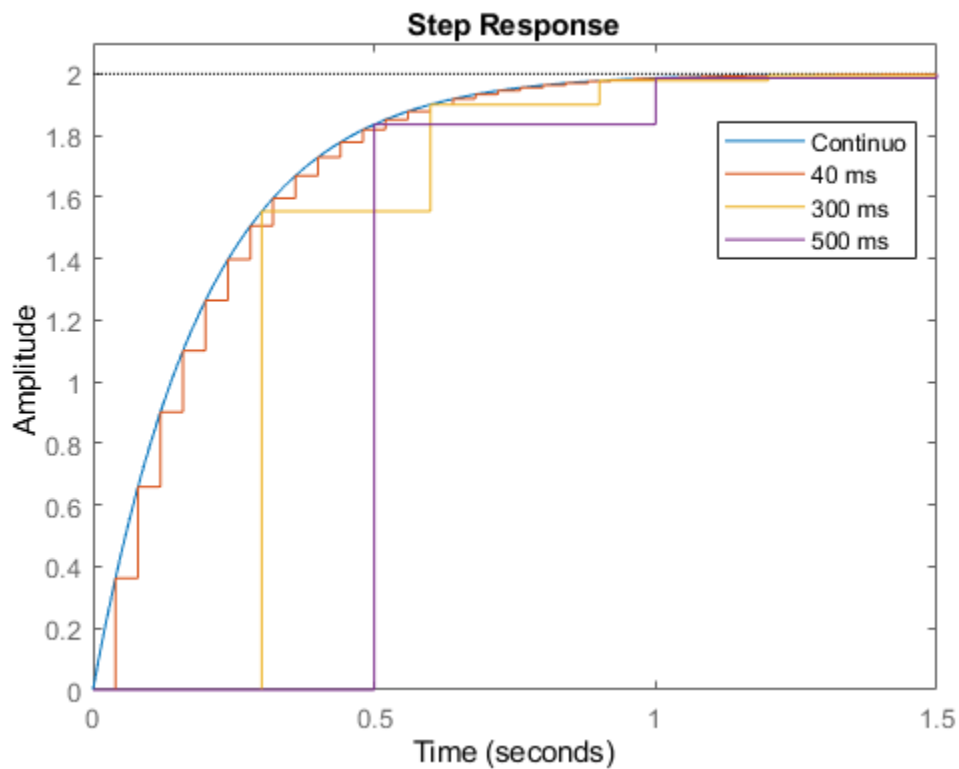


Ilustración 5.1.3. Respuesta a step G1 continuo vs. discreto.

P5.2 Control discreto de planta G1.

Replicando el uso de la función de c2d (visto en la ilustración 5.2.4) y mostrando su respuesta a step en la misma gráfica, se obtienen las funciones presentadas en la ilustración 5.2.5 y su respectiva respuesta vista en la ilustración 5.2.6.

$$G2 = \frac{200}{s^2 + 14s + 100}$$

```
P5_G2.m x untitle x P5_G1.m x P5_G2_EF.m x P5_G1_EF.m x +
1      clc
2      clear
3
4      G2 = tf(200, [1 14 100])
5
6      TFd40 = c2d(G2, 0.04)
7      TFd300 = c2d(G2, 0.3)
8      TFd500 = c2d(G2, 0.5)
9
10     step(G2)
11     hold on
12     step(TFd40)
13     step(TFd300)
14     step(TFd500)
15     legend('Continuo', '40 ms', '300 ms', '500 ms')
16     axis([0 1.5 0 2.2])
```

Ilustración 5.2.4. Código para obtención de TF discreto para G2.

```
TFd40 =

      0.1322 z + 0.1096
      -----
      z^2 - 1.45 z + 0.5712

Sample time: 0.04 seconds
Discrete-time transfer function.

TFd300 =

      1.931 z + 0.3644
      -----
      z^2 + 0.1325 z + 0.015

Sample time: 0.3 seconds
Discrete-time transfer function.

TFd500 =

      2.08 z + 0.03211
      -----
      z^2 + 0.05492 z + 0.0009119

Sample time: 0.5 seconds
Discrete-time transfer function.
```

Ilustración 5.2.5. Funciones de transferencia discretas para G2.

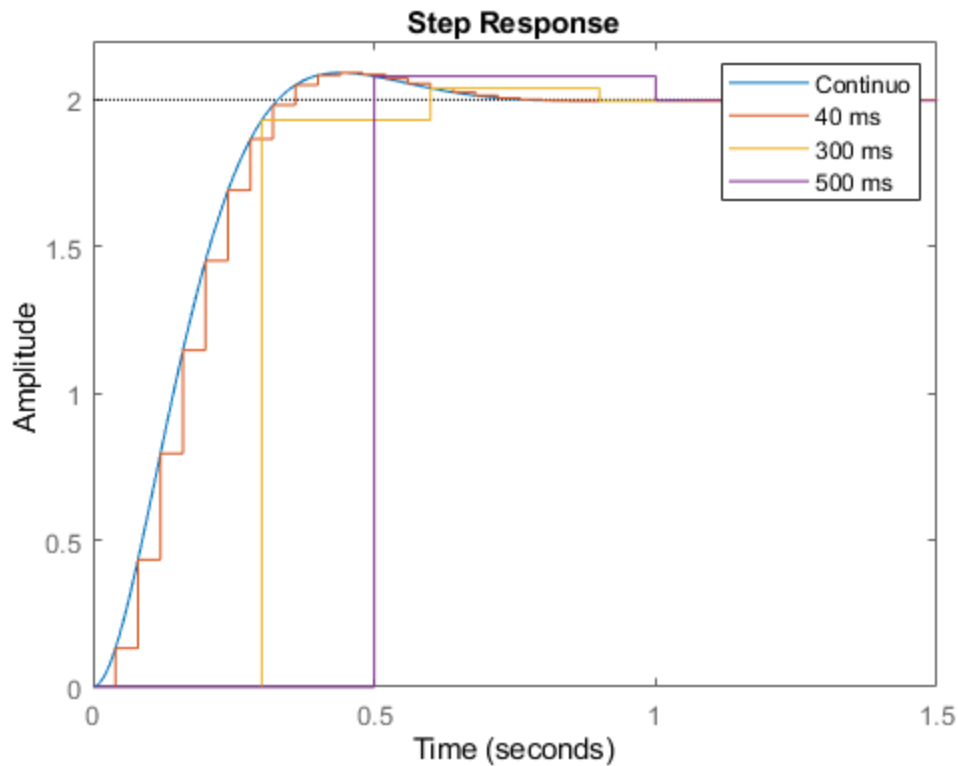


Ilustración 5.2.6. Respuesta a step $G2$ continuo vs. discreto.

E5.1 Implementando código de $G1$

El código de la función de transferencia obtenida será aplicado a código para conocer su comportamiento en mundo digital mediante el uso, por ejemplo, de Arduino, este código será presentado en la misma gráfica que la función de transferencia en tiempo continuo para ver si se comporta como se espera. El diagrama para su aplicación se presenta en la ilustración 5.1.7 y el código implementado en el sistema $G1$ a un tiempo de muestreo de 40 ms se aprecia en la ilustración 5.1.8.

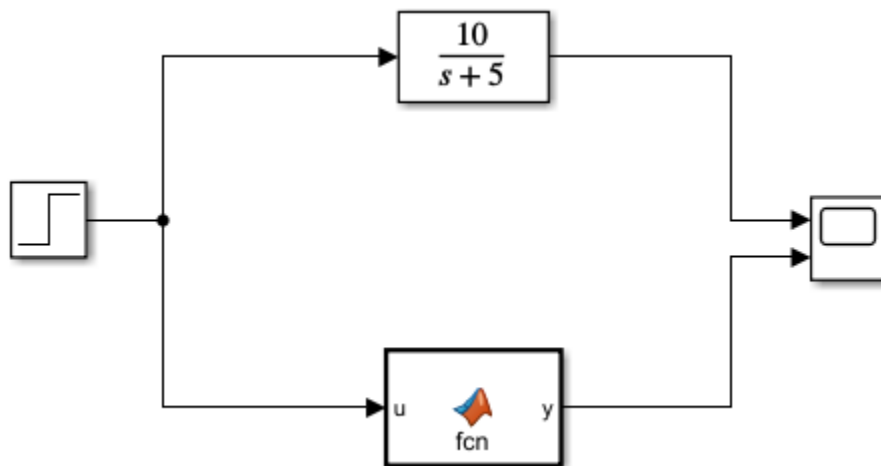
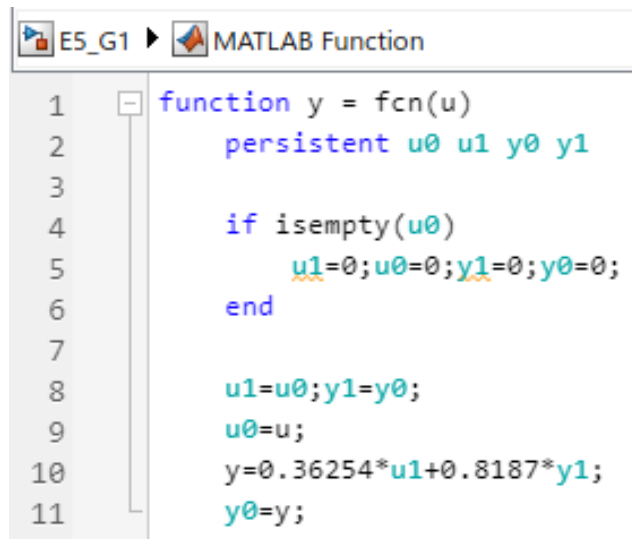


Ilustración 5.1.7. Diagrama de función de transferencia real vs. código implementado.



```

1  function y = fcn(u)
2      persistent u0 u1 y0 y1
3
4      if isempty(u0)
5          u1=0;u0=0;y1=0;y0=0;
6      end
7
8      u1=u0;y1=y0;
9      u0=u;
10     y=0.36254*u1+0.8187*y1;
11     y0=y;

```

Ilustración 5.1.8. Código implementado para G1 a 40 ms.

R5.1 Resultados:

Se presentan en la misma gráfica, el *scope*, la respuesta obtenida en la ilustración 5.1.9.

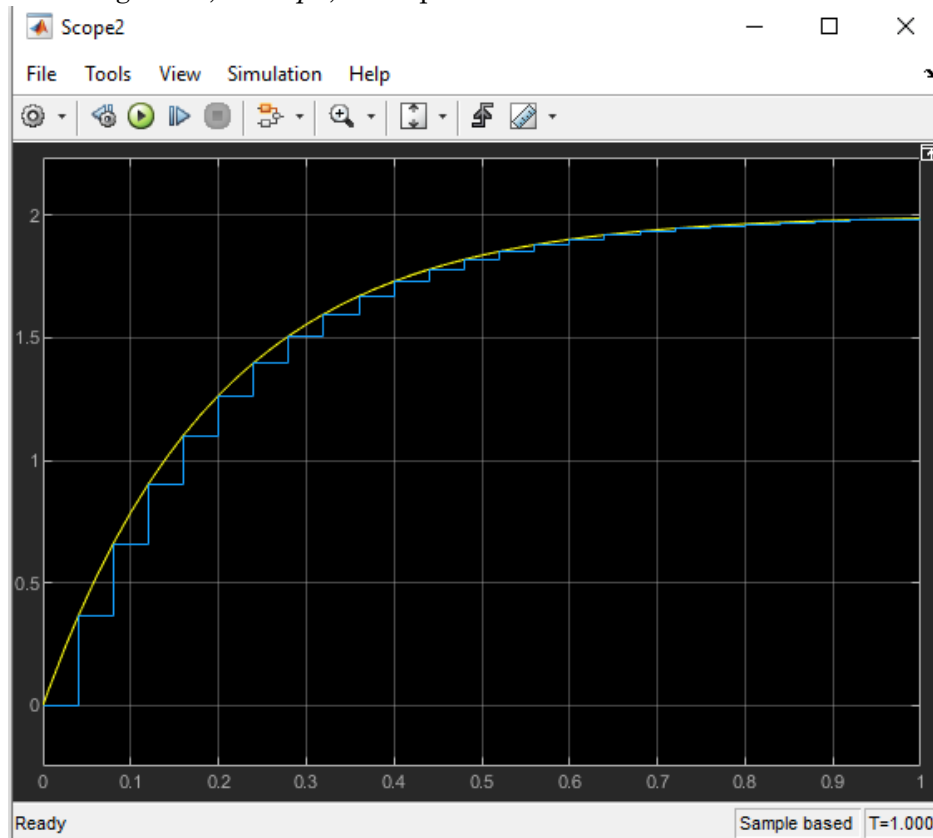


Ilustración 5.1.9. Respuesta código frente a tiempo continuo.

E5.2 Implementando código de G2

El código de la función e transferencia obtenida será aplicado a código para conocer su comportamiento en mundo digital mediante el uso, por ejemplo, de Arduino, este código será presentado en la misma gráfica que la función de transferencia en tiempo continuo para ver si se comporta como se espera. El diagrama para su aplicación se presenta en la ilustración 5.2.10 y el código implementado en el sistema G2 a un tiempo de muestreo de 40 ms se aprecia en la ilustración 5.2.11.

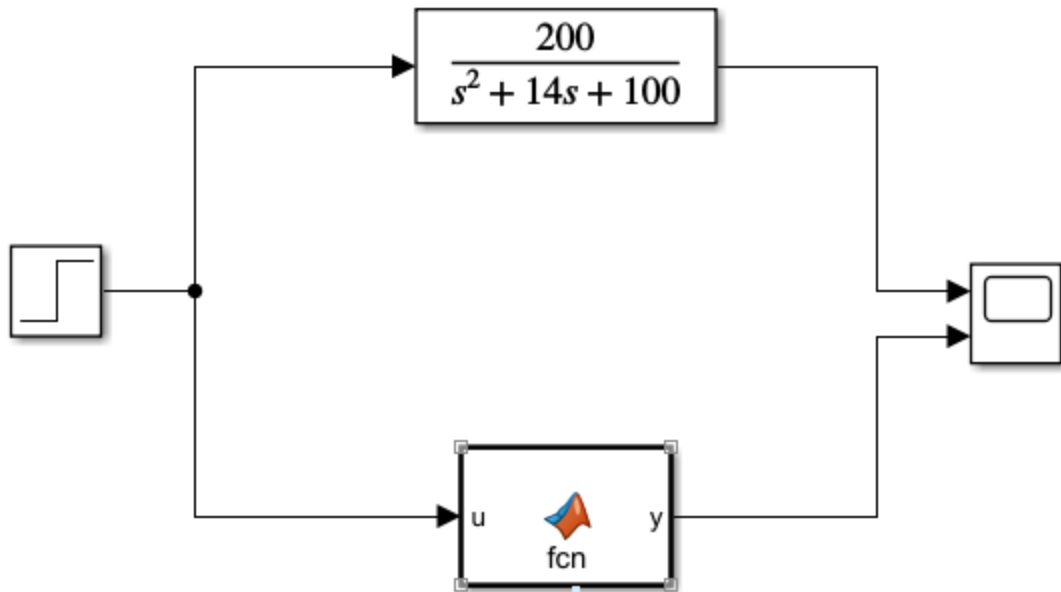


Ilustración 5.2.10. Diagrama de función de transferencia real vs. código implementado.

```

E5_G2 ▶ MATLAB Function
1  function y = fcn(u)
2      persistent u0 u1 y0 y1 u2 y2
3
4      if isempty(u0)
5          u1=0; u0=0; y1=0; y0=0; u2=0; y2=0;
6      end
7
8      u2=u1; u1=u0; y2=y1; y1=y0;
9      u0=u;
10     y = 0.1322*u1 + 0.1096*u2 + 1.45*y1 - 0.5712*y2;
11     y0=y;

```

Ilustración 5.2.11. Código implementado para G2 a 40 ms.

R5.2 Resultados:

Se presentan en la misma gráfica, el scope, la respuesta obtenida en la ilustración 5.2.12.

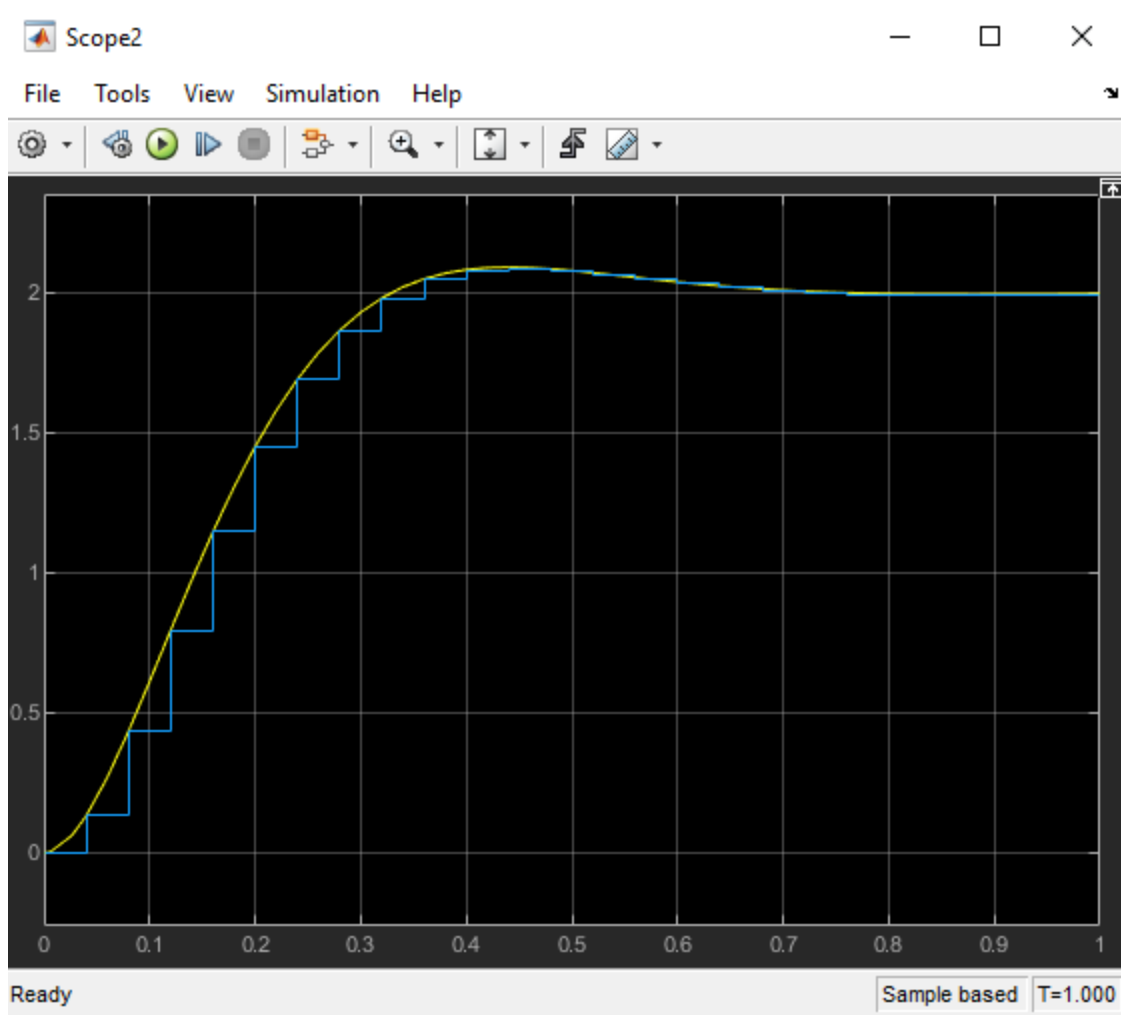


Ilustración 5.2.12. Respuesta código frente a tiempo continuo.

A5 Análisis general:

Se pudo observar como la implementación de dos códigos, mediante el uso de la función *persistent*, se pudo obtener una respuesta con el comportamiento de un ZOH de un tiempo de muestreo precisamente de 0.04 segundos.

Se comprende, por lo tanto, que el código consiste en una función que sería llamada cada cierto tiempo (en este caso tiempo de muestreo, 0.04 segundos), esta función obtiene un valor inicial de 0 tanto en la salida como la entrada, cuyos valores se van conservando cada vez que se llame dicha función; al siguiente paso se vuelve a llamar la función en el que cada vez las variables cambian y desplazan su valor para obtener el nuevo valor de la función en tal punto.

En general, se muestra el comportamiento de código como una sumatoria de valores en varios tiempos de muestreo, con dependencia del tiempo anterior, según las expresiones conocidas en el tiempo discreto.