TRANSFORMADA Z INVERSA POR EL MÉTODO COMPUTACIONAL

Práctica 2.1

Catedrático: Dr. Yahir Hernández Mier

Alumno: Ángel Arturo Ramírez Suárez

Ingeniería Mecatrónica 8 - 1

Fecha: 27 de enero de 2013

Cd. Victoria, Tam.

Resumen

En el documento presentado a continuación se muestra el método de resolución de la transformada Z inversa utilizando métodos computacionales para su obtención, lo cual puede ser utilizado en el cálculo de la respuesta y búsqueda de una acción de control para un sistema de control discreto.

Palabras clave: Sistema de control discreto, transformada z, metodo computacional.

Índice general

1. Introducción.

“La transformada Z en sistemas de control de tiempo discreto juega el mismo papel que la transformada de Laplace en sistemas de control de tiempo continuo. Para que la transformada Z sea útil, se debe estar familiarizado con los métodos para hallar la transformada Z inversa.

La notación para la transformada Z inversa será Z-1. La transformada Z inversa de X[Z] da como resultado la correspondiente secuencia X[n].

Existen cuatro métodos para obtener la transformada Z inversa, los cuales son:

Método de la División Directa. Método Computacional. Método de expansión en fracciones parciales. Método de la Integral de inversión.”[2]

Por tanto esta técnica es ampliamente utilizada para encontrar las soluciones a las ecuaciones en diferencias en el dominio del tiempo una vez que éstas han sido procesadas en el dominio Z.

En este documento se ilustran dos métodos utilizados en clase para obtener la transformada Z inversa de tres funciones utilizando los dos métodos computacionales vistos en clase que son el método de impulso unitario y el de ecuaciones en diferencias.

1.1. Objetivo.

Realizar un reporte en formato PDF que incluya los programas de la práctica hecha en clase sobre la transformada Z inversa por los dos métodos computacionales, así como las gráficas generadas. Subir el archivo al sitio del curso.

2. Método computacional.

1. Se abrió la consola de Scilab.
2. Se inicializó el editor de aplicaciones de Scilab llamado Scinotes.
3. Se introdujo el código que se describe a continuación para declarar los numeradores y denominadores de las 3 funciones a procesar:

```
//18-enero-2013.
//Algoritmo para obtener la transformada Z^-1 del sistema introducido a la entrada, esto es, la secuencia de valores (coeficientes) de la se al original.
//No genera una funci n cerrada en tiempo discreto sino las muestras particulares.
clc
clear
exec('/home/arthur/Desktop/sci_stem.sce',-1);
z = %z;
num = 0.4673*z -0.3393;
den = z^2 - 1.5327*z + 0.6607;    //Debido a que Scilab no acepta numeros negativos, se multiplica por la unidad (z^2/z^2).

num2 = 10*z + 5;
den2 = (z-1)*(z-0.2);

num3 = 1;
den3 = z + 1;
```

Donde num, num2 y num3 representan el numerador de la función de transferencia y den, den2 y den3 representan los denominadores. La letra z= %z es utilizada para que Scilab active como variable inicial a z que es un número complejo.

4. Posteriormente se dividen los numeradores con su respectivo denominador para obtener la función de transferencia. Cada una se almacena en una variable Yz, Yz2 y Yz3. Se elige además el número de muestras a visualizar, que en este ejemplo es 50 y que se indica en la variable N.

```
Yz = num/den;
Yz2 = num2/den2;
Yz3 = num3/den3;

N = 50;    //Se desea observar 50 muestras.
```

2.1. Método del impulso unitario.

Consiste en la aplicación de un impulso unitario a la función en el dominio Z, la cual es considerada como un sistema que es alimentado por dichos impulsos y que permite obtener uno a uno, los valores originales de la señal.

5. Se concatenan dos vectores, uno conteniendo un único número de valor 1 que es colocado al inicio del nuevo vector resultante y el resto conteniento ceros.

```
  deltakT = [1, zeros(1,N)];    //Se concatenan dos vectores para obtener la funci n impulso unitario.
deltakT2 = [1,zeros(1,N)];
deltakT3 = [1,zeros(1,N)];
```

6. Se realiza el análisis de la respuesta del sistema a la entrada de tipo impulso unitario, que nos permite visualizar las muestras de la señal original.

```
//Simulaci n en tiempo discreto.
yk = flts(deltakT, Yz);    //La funci n de filtros realiza la simulaci n en tiempo discreto donde deltakT es el impulso alimentado a Yz que es el sistema discreto que se desea simular. La funci n flts permite simular la respuesta del sistema a una se al de entrada determinada, en este caso el impulso Yz.

yk2 = flts(deltakT2,Yz2);

yk3 = flts(deltakT3,Yz3);
```

7. Se elaboran las gráficas de la resolución por impulso unitario.

```
scf();    //Se elaboran las graficas.
sci_stem([0:N], yk, 'red')
xgrid

scf();
sci_stem([0:N],yk2,'red')
xgrid

scf();
sci_stem([0:N],yk3,'red')
xgrid
ha = gca();
ha.data_bounds = [0 -1.5; 50 1.5]
```

Funciones graficadas.

$$X(z) = \frac{0{,}4673z - 0{,}3393}{z^2 - 1,5327z + 0,6607}$$

En la figura 2.1 puede observarse el resultado de dicha graficación.

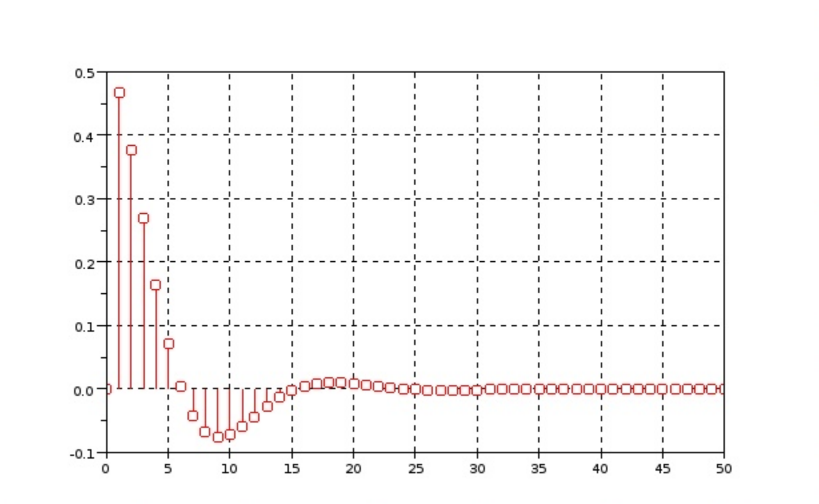

Figura 2.1: Gráfica de la función original uno resuelta utilizando el método de impulso unitario.

$$X(z) = \frac{10z + 5}{(z-1)(z-2)}$$

La gráfica resultante se muestra a continuación en la figura 2.2.

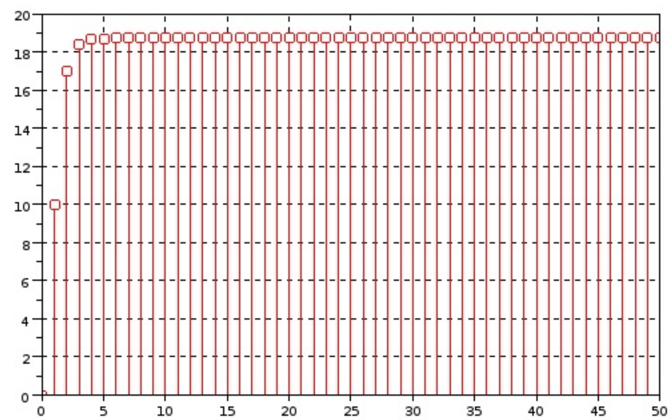

Figura 2.2: Gráfica de la función original dos resuelta utilizando el método de impulso unitario.

$$X(z) = \frac{1}{z+1}$$

Cuya gráfica se visualiza en la figura 2.3.

2.2. Ecuaciones en diferencias.

Por su parte el método de ecuaciones en diferencias es ampliamente utilizado cuando se conocen los primeros dos valores de la función, y al valerse de un método iterativo, logra obtener las muestras originales de la función.

8. Se define el número de muestras a visualizar, en este caso 50 muestras que son almacenadas en el vector N.

```
//Utilizando el m todo de ecuaciones en diferencias.

N = 50;    //Cantidad de muestras.
```

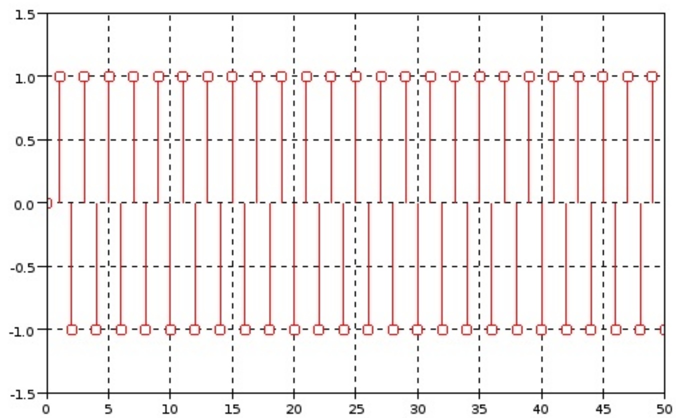

Figura 2.3: Gráfica de la función original tres resuelta utilizando el método de impulso unitario.

9. Se declara la señal de entrada, la cual tendrá un valor de impulso unitario. Ésta será colocada en x. Posteriormente se realiza una prealocación de memoria en una variable y, la cual es despejada con el propósito de permitir almacenar en ésta el valor de la señal original. Ésta tendrá el mismo tamaño o largo que x.

```
//Se al de entrada.
x = [1,zeros(1,N)];
//Se al de entrada(solo una prealocaci n de memoria).
y = zeros(1,length(x));
```

10. Se introducen a mano los primeros dos valores de las muestras de salida, los cuales deben conocerse para poder realizar las iteraciones.

```
//Sabiendo que el adelanto mayor era de 2 muestras, introducimos a
   mano los valores de las primeras dos muestras de salida.
//Salida y(0).
y(1) = 0;
//Salida y(1);
y(2) = 0.4673;
```

11. Se calcula el resto de los valores sustituyendo y operando desde la muestra uno hasta un valor

de muestra igual a $N-1$.

```
//Calculamos el resto de las muestras de forma iterativa.
for k=1:N-1
    y(k+2) = 1.5327*y(k+1)-0.6607*y(k)+0.4673*x(k+1)-0.3393*x(k);
end
```

12. Se grafica el resultado de la función y se repite el proceso para la otra señal.

```
//Graficamos.
scf();
sci_stem(0:N,y,'red')
xgrid
```

13. De la misma forma para la señal 2 se elige la cantidad de valores y se rellenan dos vectores con ceros. Posteriormente se calcula la transformada Z inversa utilizando ecuaciones en diferencias.

```
//Utilizando el m todo de ecuaciones en diferencias para (10z+5) / (z^2 -1.02z  0 .2)
N = 50;    //Cantidad de muestras.
//Se al de entrada.
x = [1,zeros(1,N)];
//Se al de entrada(solo una prealocaci n de memoria).
y = zeros(1,length(x));
//Sabiendo que el adelanto mayor era de 2 muestras, introducimos a mano los valores de las primeras dos muestras de salida.
//Salida y(0).
y(1) = 0;
//Salida y(1);
y(2) = 10;
//Calculamos el resto de las muestras de forma iterativa.
for k=1:N-1
    y(k+2) = 10*x(k+1) + 5*x(k) + 1.02*y(k+1) -0.2*y(k);
end

//Graficamos.
scf();
sci_stem(0:N,y,'red')
```

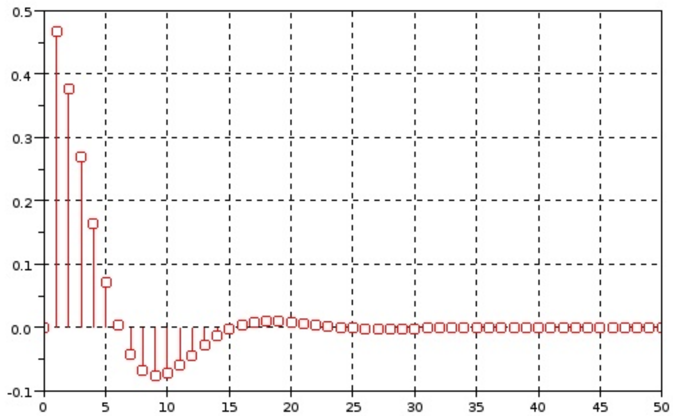

Figura 2.4: Gráfica de la función original uno resuelta utilizando el método de impulso unitario.

```
xgrid
```

$$y(k+2) = 1{,}5327y(k+1) - 0,6607y(k) + 0,4673x(k+1) - 0{,}3393x(k);$$

Su respectiva gráfica puede observarse en la figura 2.4.

$$y(k+2) = 1,02y(k+1) - 0{,}2y(k) + 10x(k+1) + 5x(k);$$

En la figura 2.5 se puede observar la gráfica resultante.

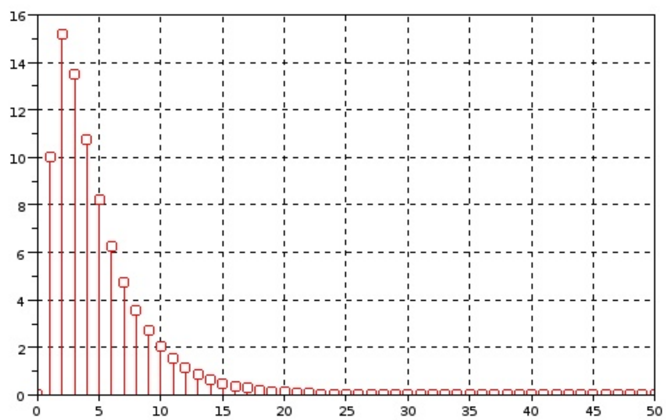

Figura 2.5: Gráfica de la función original dos resuelta utilizando el método de impulso unitario.

3. Conclusiones.

Mediante la elaboración de la práctica 2.1 fue posible el validar la funcionalidad de los métodos computacionales para la resolución de la transformada Z en funciones de transferencia, lo cual resulta de suma importancia para la elaboración de un control funcional ya que acelera enormemente el proceso al liberar al desarrollador de la tarea de realizar la resolución de las ecuaciones para lograr el recuperar una señal en el dominio de Z a su forma discreta en el dominio del tiempo.

De los dos métodos, el que considero de mayor aplicación es el de impulso unitario cuando no se tiene la posibilidad de conocer la respuesta de las dos primeras señales a la salida de un proceso ya que en este caso el método de ecuaciones en diferencias se vuelve más complicado.

Puede notarse además que el resultado obtenido en la gráfica 1 y la 4 al igual que la gráfica 2 y 5 son iguales, por lo que no se perciben variaciones apreciables en el resultado utilizando cualquiera de los dos métodos.

Debido a que el tiempo requerido para la resolución de las funciones por ambos métodos es relativamente similar, cualquiera de los dos métodos pueden ser implementados sin problemas para la obtención de la transformada Z inversa de los sistemas.

A. Código para la resolución de la transformada Z inversa por el método computacional.

```
 //18-enero-2013.
//Algoritmo para obtener la transformada Z^-1 del sistema introducido a la entrada, esto es, la secuencia de valores (coeficientes) de la se al original.
//No genera una funci n cerrada en tiempo discreto sino las muestras particulares.
clc
clear
exec('/home/arthur/Desktop/sci_stem.sce',-1);
z = %z;
num = 0.4673*z -0.3393;
den = z^2 - 1.5327*z + 0.6607;    //Debido a que Scilab no acepta numeros negativos, se multiplica por la unidad (z^2/z^2).

num2 = 10*z + 5;
den2 = (z-1)*(z-0.2);

num3 = 1;
den3 = z + 1;

Yz = num/den;
Yz2 = num2/den2;
Yz3 = num3/den3;

N = 50;    //Se desea observar 50 muestras.
```

```

deltakT = [1, zeros(1,N)];    //Se concatenan dos vectores para obtener la funci n impulso unitario.
deltakT2 = [1,zeros(1,N)];
deltakT3 = [1,zeros(1,N)];

//Simulaci n en tiempo discreto.
yk = flts(deltakT, Yz);    //La funci n de filtros realiza la simulaci n en tiempo discreto donde deltakT es el impulso alimentado a Yz que es el sistema discreto que se desea simular. La funci n flts permite simular la respuesta del sistema a una se al de entrada determinada, en este caso el impulso Yz.

yk2 = flts(deltakT2,Yz2);

yk3 = flts(deltakT3,Yz3);

scf();    //Se elaboran las graficas.
sci_stem([0:N], yk, 'red')
xgrid

scf();
sci_stem([0:N],yk2,'red')
xgrid

scf();
sci_stem([0:N],yk3,'red')
xgrid
ha = gca();
ha.data_bounds = [0 -1.5; 50 1.5]



//Utilizando el m todo de ecuaciones en diferencias.

N = 50;    //Cantidad de muestras.
//Se al de entrada.
x = [1,zeros(1,N)];
//Se al de entrada(solo una prealocaci n de memoria).
y = zeros(1,length(x));
```

```
//Sabiendo que el adelanto mayor era de 2 muestras, introducimos a mano los valores de las primeras dos muestras de salida.
//Salida y(0).
y(1) = 0;
//Salida y(1);
y(2) = 0.4673;
//Calculamos el resto de las muestras de forma iterativa.
for k=1:N-1
    y(k+2) = 1.5327*y(k+1)-0.6607*y(k)+0.4673*x(k+1)-0.3393*x(k);
end

//Graficamos.
scf();
sci_stem(0:N,y,'red')
xgrid


//Utilizando el m todo de ecuaciones en diferencias para (10z+5) / (z^2 -1.02z  0 .2)
N = 50;    //Cantidad de muestras.
//Se al de entrada.
x = [1,zeros(1,N)];
//Se al de entrada(solo una prealocaci n de memoria).
y = zeros(1,length(x));
//Sabiendo que el adelanto mayor era de 2 muestras, introducimos a mano los valores de las primeras dos muestras de salida.
//Salida y(0).
y(1) = 0;
//Salida y(1);
y(2) = 10;
//Calculamos el resto de las muestras de forma iterativa.
for k=1:N-1
    y(k+2) = 10*x(k+1) + 5*x(k) + 1.02*y(k+1) -0.2*y(k);
end

//Graficamos.
scf();
sci_stem(0:N,y,'red')
xgrid
```

Bibliografía

[1] Katsuhiko Ogata, Sistemas de Control en Tiempo Discreto, Segunda Edición, Ed. Prentice Hall, 1996.

[2] Docencia: la transformada Z inversa.
Visto por última vez: 27/enero/2013.
http://docencia.udea.edu.co/SistemasDiscretos/contenido/inversa.html