PROCESAMIENTO DIGITAL DE SEÑALES

PROFESOR: HUMBERTO LOAIZA C., Ph.D.

LABORATORIO No. 3. (software)

Recursividad - Análisis de Sistemas

Fracciones Parciales – Test de Schür-Cohn – Sistemas Recursivos – Localización de Polos y Ceros –Respuesta de sistemas de 1^{er} y 2^{do} orden

I. Objetivos

- *Implementar herramientas para la descomposición en fracciones parciales de funciones de transferencia racionales.*
- *Implementar el algoritmo de Schür-Cohn para la verificación de estabilidad de sistemas.*
- *Implementar y analizar sistemas recursivos.*
- *Graficar e interpretar los polos y ceros de un sistema en el plano z.*
- *Analizar la respuesta de los sistemas de primer y segundo orden en función de la ubicación de polos y ceros.*

II. Introducción

1. Procedimiento de Matlab cuando ejecuta funciones – scripts

Cuando se envía un comando, el interpretador de Matlab realiza las siguientes acciones:

i. *Verifica si el nombre es una variable.*
ii. *Verifica que el nombre es una función interna o propia que no fue sobrecargada.*
iii. *Verifica que el nombre es una función local en el sentido de múltiples archivos.*
iv. *Verifica que el nombre es una función de un directorio privado.*
v. *Localiza cualquier ocurrencia de una función en directorios de métodos y sobre la ruta. El orden no es tenido en cuenta.*

2. Durante la ejecución Matlab

vi. *Verifica si el nombre está ligado a una función específica (ii, iii, iv)*
vii. *Utiliza reglas de precedencia para determinar cual de las instancias i a v debe invocar.*

Cuando se llama a una función en un archivo .M desde la línea de comandos o desde otro archivo .M, Matlab *desagrega* la función y la almacena en memoria hasta que sean desactivadas con el comando *clear* o se salga de Matlab. El comando *pcode* (pseudo código) realiza la etapa de desagregación y almacena el resultado en un archivo .P para ser cargado después.

3. Variables Globales

Para permitir que una variable sea compartida por varias funciones, esta debe declararse como ***global*** en todas las funciones. Si se desea que la variable sea compartida en el espacio de trabajo debe realizarse la misma operación desde la línea de comandos. La declaración como global debe realizarse antes que la variable se utilice en una función.

Aunque no es necesario, se recomienda utilizar letras mayúsculas en los nombres de las variables globales para facilitar su diferenciación con las variables locales.

Por ejemplo, la función almacenada en el archivo caída .m,

```
function recorrido= distancia(t)
global VELOCIDAD;
recorrido= VELOCIDAD * t;
```

Para su utilización se requiere introducir desde la línea de comandos,

```
>> global VELOCIDAD;
>> d= distancia(0: 0.5: 10);
```

III. Funciones prácticas de Matlab

- *Funciones para conversión y generación de respuestas en varios dominios*

- **impz.** Calcula la respuesta impulsional de sistemas digitales.
- **freqzplot.** Grafica la respuesta en frecuencia.
- **polyscale**. Escala las raíces de un polinomio en el plano z.
- **sos2zp.** Convierte funciones digitales expresadas en secciones de segundo orden a forma de ceros y polos.
- **tf2zp.** Convierte funciones de transferencia a forma de ceros y polos.
- **zp2sos.** Convierte sistemas expresados en forma de ceros y polos a secciones de segundo orden.
- **zp2tf.** Convierte sistemas expresados en forma de ceros y polos a funciones de transferencia.

- *Funciones para la manipulación de funciones*

- **str2func**: construye un manejador (handle) de una función a partir de un nombre indicado en una cadena de caracteres.
- **func2str**: realiza la operación contraria a str2func.
- **@** function_handle (manejador de funciones): retorna un manejador de una función especificada. Se emplea con la función *feval*.
- **feval**: evalúa la función especificada por su handle.

eval: ejecuta una expresión, o una cadena que contenga cualquier expresión válida de Matlab. Puede recibir como argumento expresiones construidas como resultado de concatenaciones de subcadenas y variables.

IV. Procedimiento

1. Descomposición en fracciones parciales

1.1. Elabore una función que descomponga en fracciones parciales una función de transferencia H(z), con base en el siguiente *script*,

```
% H(z)= (-15+8z^-1+8z^-2 )/(1+ 12z^-1+ 3z^-2 –8z^-3)

b= [-15  8  8 ];
a= [  1  12  3  -8];
[residuos, polos, directos ]= residuez (b,a)
```

Consigne el código y explique el algoritmo del programa.

1.2. Pruebe el programa para por lo menos cuatro funciones de transferencia. Verifique los resultados con la función **roots( )**. Pruebe también un ejemplo donde el orden del numerador es mayor que el denominador.

1.3. Analice el algoritmo utilizado por *residuez* y explique en forma práctica el procedimiento implementado para obtener las fracciones parciales. Que diferencias existen en comparación con el método clásico referenciado en clase?

1.4. Implemente una función en Matlab para verificar el Teorema de Superposición. La función debe obtener la respuesta al impulso utilizando la función racional y la suma de las respuestas de cada fracción parcial de H(z) sobre una cantidad suficiente de instantes de tiempo que permita comprobarlo gráficamente. La función debe visualizar las respuestas totales y parciales. También debe calcular y graficar el error obtenido, así como desplegar las ecuaciones de las fracciones parciales. Utilice la función *filter*( ).

Consigne el código y explique el algoritmo del programa.

1.5. Pruebe la función para por lo menos cuatro funciones de transferencia H(z) distintas que presenten tanto polos reales como complejos y un orden superior a cuatro.

Verifique analíticamente que las respuestas obtenidas para cada fracción parcial son válidas (Incluya un solo análisis en el informe).

Consigne y analice los resultados obtenidos.

2. Test de estabilidad de Schür-Cohn

2.1. Implemente en Matlab el algoritmo de Schür-Cohn para determinar la estabilidad de sistemas definidos por su función de transferencia racional H(z). La función debe evitar producir una falla para coeficientes iguales a 1 y debe poseer el siguiente encabezado:

function [flag, coef] = test_sc (b,a)

donde,

b → coeficientes polinomio numerador de H(z)
a → coeficientes polinomio denominador de H(z)
flag → mensaje indicando si el sistema es estable o inestable.
coef → vector que contiene los coeficientes de Schür-Cohn

Consigne el código y explique el algoritmo del programa.

2.2. Pruebe la función implementada en §2.1 para por lo menos cuatro diferentes funciones de transferencia de orden 4 y 5 (dos estables y dos inestables). Verifique la respuesta obtenida calculando las raíces del denominador de H(z) con ayuda de la función **roots ( )**.
Consigne y analice los resultados obtenidos.

2.3. Indique las ventajas y desventajas (velocidad, memoria, algoritmo,…) de los métodos de S-C y del cálculo directo de los polos para determinar estabilidad de sistemas discretos cuando se implementan en aplicaciones embebidas.

3. Generación de funciones recursivas

3.1. Implemente una función en Matlab que permita evaluar sistemas recursivos y visualizar las secuencias de entrada y salida en una sola ventana. El programa debe también graficar el error obtenido en cada iteración (al compararlo con el resultado de una calculadora). La función debe presentar el siguiente formato (ver funciones *eval* y *feval*):

function [y] = sist_recur (f_sist, y_ini, x, n)

donde,

f_sist → cadena de caracteres con el nombre de ***cualquier función*** en Matlab que contiene la ecuación recursiva. (esta función solo debe tener la definición de la función recursiva)
y_ini → vector de condiciones iniciales
x → vector que contiene la secuencia de entrada
n → vector que contiene los ***instantes*** consecutivos donde se evalúa la ecuación.
Ejemplo, n=[2 3 4 5] indica que se evalúe desde el instante dos hasta el cinco incluidos.
y → vector que contiene la secuencia de salida

Consigne el código y explique el algoritmo.

3.2. Pruebe la función de §3.1 con el algoritmo para el cálculo de la raíz cuadrada de un número A.

$$y(n) = 0.5 * [y(n-1) + \frac{x(n)}{y(n-1)}]$$
$$x(n) = A * u(n)$$
$$y(-1) = \textit{condición inicial (estima de la raíz)}$$

Verifique el buen funcionamiento del programa para los siguientes valores de A y de iteraciones (tamaño diferente del vector n).

a- A= 100, size(n) = 20 instantes
b- A= 25, size(n) = 15 instantes
c- A= 15, size(n) = 10 instantes
d- A= 7.5, size(n) = 5 instantes

3.3. Consigne y analice los resultados. Indique, si es el caso, como se puede disminuir el error y a la vez el tiempo de ejecución.

3.4. Explique cómo se podría establecer el número de iteraciones para obtener una precisión predeterminada (número de dígitos significativos), en el cálculo recursivo de la raíz cuadrada?

3.5. Pruebe la función de §3.1 con el algoritmo para el cálculo del promedio acumulativo de una secuencia de números.

$$y(n) = \frac{n}{n+1} y(n-1) + \frac{1}{n+1} x(n)$$
$$x(n) = \{\textit{muestras para evaluar la media acumulativa}\}$$
$$y(-1) = 0 \qquad \textit{condición inicial}$$

Verifique el buen funcionamiento del programa para cuatro diferentes secuencias de longitud diferente y mayor a 4.

3.6. Consigne y analice los resultados. Indique, si es el caso, como se puede disminuir el error y a la vez el tiempo de ejecución.

3.7. Modifique la función desarrollada en §3.1 de forma tal que no se necesite el parámetro n y se detenga automáticamente cuando alcance una buena aproximación del resultado. Pruebe para las funciones recursivas de §3.2 y §3.5. Explique claramente su procedimiento.

3.8. Para las siguientes series de Taylor obtenga la función recursiva y(n). Plantee las condiciones iniciales necesarias.

a- $e^A = 1 + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \ldots$

$x(n) = A * u(n)$

b- $sen(A) = A - \frac{A^3}{3!} + \frac{A^5}{5!} - \frac{A^7}{7!} + \ldots$

$x(n) = A * u(n)$

Verifique el buen funcionamiento en §a y en §b para por lo menos dos valores de A y calcule el error.

4. Visualización de polos y ceros

4.1. Implemente una función en Matlab que permita visualizar en el *plano z* los polos y ceros de una función de transferencia racional H(z). También debe visualizar el círculo unitario y los ejes. Estudie la función *zplane* y *pzmap* de Matlab.

La función debe presentar el siguiente encabezado:

function [polos, ceros] = graph_z (b,a)

donde,

b → coeficientes polinomio numerador de H(z)
a → coeficientes polinomio denominador de H(z)
polos → vector que contiene los polos del sistema
ceros → vector que contiene los ceros del sistema

Consigne el código y explique el algoritmo utilizado.

4.2. Con las mismas funciones empleadas para el análisis de estabilidad de Schür-Cohn (§2.2) verifique el buen funcionamiento de la función **graph_z**.

4.3. Indique si los resultados sobre estabilidad obtenidos en §2.2 y en §4.2 son diferentes. Se esperaba el resultado obtenido? Consigne y analice los resultados obtenidos. Compare la utilidad de los dos métodos.

5. Análisis de un sistema de *primer* orden

5.1. Realice una función en Matlab que permita graficar en tiempo-discreto la respuesta de un sistema de *primer* orden ante una entrada impulso, $\delta(n)$, y una entrada escalón unitario, $u(n)$, para un número amplio de muestras. La función debe presentar adicionalmente la gráfica de

los polos en el plano z, todo en una misma ventana. El sistema está determinado por las siguientes expresiones:

$$h(n) = a^n\, u(n) \quad \leftrightarrow \quad H(z) = \frac{1}{1 - a\, z^{-1}}$$

Consigne el código y explique el algoritmo del programa implementado.

5.2. Pruebe la función para los siguientes valores de *a*:

i.	$0 < a < 1$	iv.	$-1<a<0$
ii.	$a=1$	v.	$a= -1$
iii.	$a>1$	vi.	$a < -1$

Consigne y analice los resultados obtenidos.

5.3. Obtenga analíticamente la respuesta h(n) para cada caso. Son coherentes la ubicación del polo y las respuestas en el tiempo obtenidas en §5.2 ?

6. Análisis de un sistema de *segundo* orden

6.1. Realice una función en Matlab que permita graficar en tiempo-discreto la respuesta de un sistema de *segundo* orden ante una entrada impulso, $\delta(n)$, y una entrada escalón unitario, $u(n)$, para un número amplio de muestras. La función debe presentar adicionalmente la gráfica de los polos en el plano z, todo en una misma ventana. El sistema está determinado por las siguientes expresiones:

$$y(n) = -a_1\, y(n-1) - a_2\, y(n-2) + b_0\, x(n) \quad \leftrightarrow \quad H(z) = \frac{b_0\, z^2}{z^2 + a_1\, z + a_2}$$

Consigne el código y explique el algoritmo del programa implementado.

6.2. Pruebe la función para diferentes valores de a_1 y a_2 tal que:

i. $a_1^2 > 4a_2$ ii. $a_1^2 = 4a_2$ iii. $a_1^2 < 4a_2$

Consigne y analice los resultados obtenidos.

6.3. Obtenga analíticamente la respuesta h(n) para cada caso. Son coherentes la ubicación del polo y las respuestas en el tiempo obtenidas en §6.2 ?

6.4. Compare las respuestas del sistema de segundo orden con las de primer orden obtenidas en §5.2 considerando oscilación, límites de estabilidad, respuesta con polos sobre la circunferencia unidad,… justifique sus respuestas.

7. Consulta

7.1. Realice el resumen de un artículo que describa una aplicación de procesamiento digital de señales: Señales fisiológicas, señales sísmicas, señales sensoriales o aplicación en comunicaciones.
Consigne la referencia bibliográfica.

8. Informe

8.1. Presente un informe escrito claro en donde se consigne el procedimiento, las gráficas, los programas, las respuestas, las justificaciones y los resultados obtenidos. Además, consigne el análisis e interpretación de los resultados. Igualmente incluya conclusiones, observaciones, y la literatura consultada. El informe debe utilizar la misma numeración de la guía.

8.2. Elabore información de ayuda para cada una de las funciones implementadas. Estas ayudas deberán consultarse con el comando o botón *help* de Matlab.

Nota:

a) ***Para facilitar la sustentación ante el profesor, realice scripts donde se definan los datos y se invoquen las funciones desarrolladas.***
b) La omisión de alguno de los ítems en el informe representa una disminución de la nota.
c) Las funciones elaboradas deberán conservarse para su utilización en prácticas posteriores.