UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO

FACULTAD DE INGENIERÍA ELÉCTRICA

ELABORACIÓN DE HERRAMIENTA GRÁFICA DIDÁCTICA PARA DISEÑO DE FILTROS DIGITALES MEDIANTE EL MÉTODO DE MANIPULACIÓN DE POLOS Y CEROS EN LA PLATAFORMA SCILAB.

TESIS

Que para obtener el título de:

INGENIERO EN ELECTRÓNICA

Presenta:

GUILLERMO CENDEJAS ZAVALA

Asesor de tesis:

M.C. ANTONIO ULISES SÁENZ TRUJILLO

Morelia Michoacán, marzo 2017

Agradecimientos:

Al M.C Ulises saenz por su asesoría siempre dispuesta y el tiempo dedicado.

A la Universidad Michoacana de San Nicolás de Hidalgo por brindarme la oportunidad de prepararme en la misma.

A la mesa sinodal por sus ideas y recomendaciones respecto a ésta tesis.

A mi madre y padre por todo el apoyo brindado y la pasiencia que tuvieron en el transcurso de mi desarrollo académico.

A mis amigos con quienes compartí momentos de desesperación y satisfacción, así como momentos de diversión en el transcurso de mi formación académica.

Dedicatorias:

A mi madre por haberme apoyado en todo momento, por sus consejos, sus valores, por la motivación constate que me ha permitido seguir adelante, pero más que nada, por su amor.

A mi padre por los ejemplos de perseverancia, ejemplos de humildad y carisma que lo caracterizan y que me ha infundido siempre en mí y mis hermanos, por su fortaleza para salir adelante a pesar de las adversidades y por su amor.

A mis amistades que aún en día seguimos siendo amigos, con los cuales mi vida tiene un toque de diversión y de convivencia.

A laura Gabriela quien me acompañó en la mayor parte de mi desarrollo académico alentándome con palabras de apoyo a seguir adelante en mis estudios.

Resumen:

En este trabajo se presenta la elaboración de una herramienta para diseño de filtros digitales en la plataforma SCILAB®, la cual consiste de una interfaz gráfica de usuario, la herramienta permite modificar las características del filtro mediante el método de manipulación de polos y ceros, obteniendo la gráfica de respuesta en frecuencia en base al filtro diseñado.

La herramienta se compone básicamente de cuatro módulos:

- Interfaz gráfica de usuario para el diseño del filtro: ventana gráfica que permite ingresar las características del filtro a diseñar.

- Interfaz gráfica la cual permite manipular el polo o cero en base a sus coordenadas mediante cuadros de texto.

- Gráfica del círculo unitario: permite manipular en base a movimientos del puntero la ubicación del polo o cero seleccionado.

- Gráfica de respuesta en frecuencia: muestra la respuesta en frecuencia del filtro diseñado.

Operación básica: A través de una ventana gráfica o por medio de colocación de polos y ceros se adquieren los datos de diseño del filtro deseado, posteriormente se realiza el proceso de mostrar en el plano z complejo todos los ceros y polos del filtro, así como la gráfica de su respuesta en frecuencia; una vez realizado, se podrá seleccionar los polos y ceros para su manipulación. Estos datos pueden ser modificados de forma gráfica con el puntero, o por medio del cambio de coordenadas del polo o cero seleccionado, mostrando su respuesta en frecuencia en base a la posición de los polos y ceros.

Palabras clave: Filtros digitales, herramienta gráfica, polos y ceros, SCILAB,respuesta en frecuencia.

Abstract:

This work describes the development of a tool for digital filter design in SCILAB ® platform, which is represented by a graphical user interface, the tool allows to change the filter characteristics by the method of handling poles and zeros, plotting the frequency response based on the designed filter.

The tool has four modules:

- Graphical user interface for filter design: a graphic windows that allows to enter the characteristics of the filter to be designed.

- Graphical interface which allows to manipulate the pole or zero based on its coordinates using text boxes.

- Graph of the unit circle: allows to manipulate the location of the pole or zero selected based on pointer position.

- Frequency Response Graph: shows the frequency response of the filter designed.

Basic Operation: Through a graphical window or through placement of poles and zeros data are acquired to design the desired filter, then the process is executed to display on the unit circle all zeros and poles of the designed filter, and the graph of the frequency response, once that operation has been performed, poles and zeros could be manipulated. These data can be modified graphically by the cursor, or changing ubication of the pole or zero selected, frequency response can be displayed.

Índice

Lista de Figuras

Lista de Ecuaciones

Lista de Pseudocódigos

Capítulo 1
Introducción

Un filtro digital es un sistema que opera sobre señales digitales. Es una operación matemática que toma una secuencia de números (la señal de entrada) y la modifica produciendo otra secuencia de números (la señal de salida) con el objetivo de resaltar o atenuar ciertas características de la señal[1].

1.1 Antecedentes

1.1.1 FDA Tool

Es una herramienta de la plataforma MATLAB®, la cual consiste en una interfaz gráfica para el diseño y análisis de filtros digitales. FDA tool permite diseñar de manera rápida filtros digitales FIR (*Finite impulse response*) o IIR (*Infinite impulse response*) mediante el ingreso de especificaciones del filtro importado desde la ventana de trabajo de MATLAB® o agregando, moviendo o borrando polos y ceros. FDA tool también proporciona herramientas para el análisis de filtros, como su respuesta en magnitud y fase o la gráfica de polos y ceros en el círculo unitario[2].

En la Figura 1.1 se muestra un ejemplo de un módulo de FDA tool el cual permite mover y borrar polos o ceros del filtro diseñado (en este caso un sistema FIR), obteniendo su respuesta en frecuencia en base a los cambios realizados.

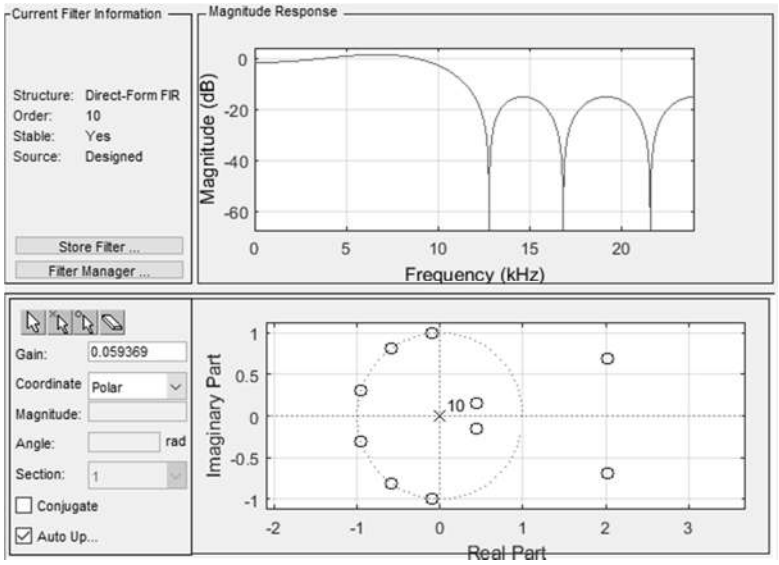

Figura 1.1 FDA tool ejemplo de filtro pasa bajas [2]

1.1.2 Wfir_GUI

Interfaz gráfica de usuario del software SCILAB® que se utiliza para diseñar de forma interactiva filtros FIR[3]. Esta interfaz permite ingresar los parámetros para el filtro a diseñar, a su vez contiene una pre visualización de la respuesta en frecuencia que se obtendrá con los parámetros del filtro a diseñar.

Para los sistemas FIR existe una herramienta de diseño gráfico que se ejecuta con el comando *wfir* como se observa en la Figura 1.2, sin embargo, carece de una interfaz para diseñar sisemas IIR.

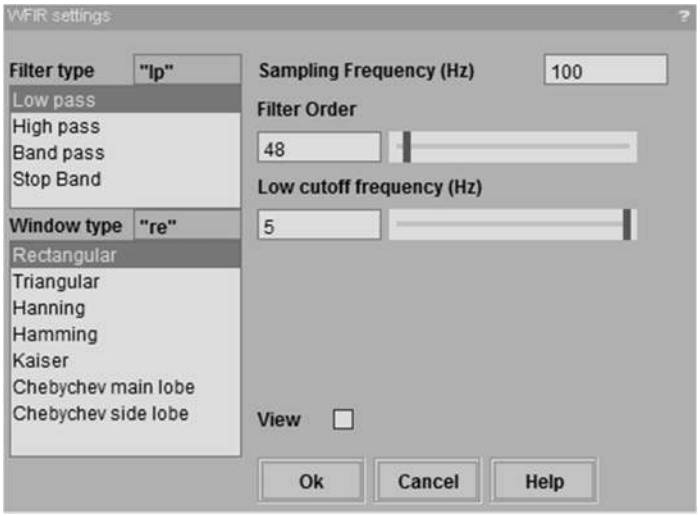

Figura 1.2 Ventana gráfica del comando wfir [3]

1.1.3 SciPy

SciPy es una biblioteca *open source* de herramientas y algoritmos matemáticos para Python que nació a partir de la colección original de Travis Oliphant que consistía de módulos de extensión para PythonSciPy que contiene módulos para optimización, álgebra lineal, integración, interpolación, funciones especiales, FFT, procesamiento de señales y de imagen, resolución de ODEs y otras tareas para la ciencia e ingeniería[4].

Esta herramienta permite crear filtros digitales únicamente por medio de comandos, no cuenta con una interfaz gráfica que facilite el diseño del filtro, o permita comprobar de forma interactiva la respuesta del mismo.

1.2 Objetivo

1.2.1 Objetivo general

Implementar una interfaz gráfica que sirva como herramienta para diseñar filtros digitales y permitir manipular sus polos y ceros de manera didáctica, con el fin de que el usuario observe la importancia de la posición que tienen los polos y ceros en el plano Z, al igual de poder implementar el filtro digital para fines particulares.

1.2.2 Objetivos particulares

- Diseñar una ventana gráfica para ingresar los parámetros del filtro.
- Implementar un algoritmo que a partir de los coeficientes del filtro muestre en el plano z complejo la ubicación de sus respectivos polos y ceros.
- Diseñar una ventana gráfica para permitir la manipulación de los polos y ceros colocados, ya sea por medio del cambio de parámetros o por manipulación en la gráfica del círculo unitario.
- Graficar la respuesta en frecuencia del sistema obtenido, y actualizar de manera automática en base a los cambios que se realicen.
- Elaborar un manual para el manejo de la herramienta.

1.3 Justificación

- Debido a que SCILAB® no cuenta con una interfaz gráfica que permita crear filtros FIR e IIR mediante la ubicación de sus polos y ceros, se tiene la necesidad de crear una herramienta que lo permita de una manera didáctica.
- Esta herramienta permite la práctica del diseño de filtros de manera didáctica, logrando diseñar el filtro digital en un menor tiempo y de una manera sencilla.
- El hecho de que SCILAB® es un software libre multiplataforma (Windows®, Mac®, Linux®) permite que todo usuario con una computadora que tenga los requerimientos mínimos de SCILAB® pueda tener acceso a esta herramienta.
- El poder manipular los polos y ceros facilita el análisis y relevancia que tienen en el sistema y en su respuesta en frecuencia.

1.4 Metodología

Se realiza una investigación bibliográfica de filtros digitales, al igual que se obtiene conocimiento previo del FDA tool como base para la creación de la herramienta. Se lleva cabo una investigación para el manejo adecuado del software SCILAB® y se hace consulta de libros en internet para diseño de filtros digitales en SCILAB®. Posteriormente se realiza una revisión del script *wfir_gui.sce* para conocimiento más profundo.

Se desarrolla la herramienta para el software SCILAB® en la cual previamente se realizan pruebas para evaluar la eficiencia del programa, y determinar las especificaciones mínimas y máximas para el orden del filtro.

1.5 Contenido de la tesis

En el Capítulo 1 se da una breve introducción a este trabajo, se muestran los antecedentes históricos de herramientas para diseñar filtros digitales. Se describe el objetivo, se explica por qué la decisión de desarrollar este tema y por último el marco teórico.

En el Capítulo 2 se muestra el desarrollo de la herramienta, la plataforma usada para la herramienta, explicación de una interfaz gráfica de usuario, las dos formas implementadas para diseñar filtros, así como la construcción de la interfaz gráfica principal y secundaria.

En el Capítulo 3 se presentan las pruebas realizadas de la herramienta diseñada, así como los resultados obtenidos.

En el Capítulo 4 se exponen las conclusiones generales, conclusiones particulares y trabajos futuros.

1.6 Marco teórico

1.6.1 Señal discreta en el tiempo

Una señal discreta es una señal que toma valores dentro un conjunto finito de posibles valores. Una señal discreta en el tiempo $x[n]$ es una función de una variable independiente que es un entero. Una señal discreta en el tiempo *no está* definida en los instantes entre dos muestras sucesivas [5].

1.6.2 Señal digital

Una señal discreta en el tiempo que tiene un conjunto de valores discretos (números enteros) es una señal digital. La Figura 1.3 muestra un ejemplo de una señal digital.

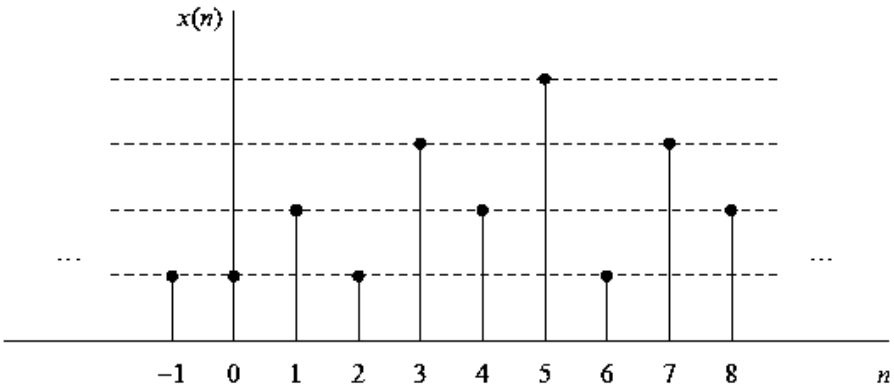

Figura 1.3 Señal digital con cuatro valores de amplitud [5]

1.6.3 Sistemas en tiempo discreto

Un sistema discreto es una operación o un conjunto de operaciones que se realizan sobre la señal de entrada $x[n]$ para producir una señal de salida $y[n]$ [5]. La señal de salida está relacionada con la entrada mediante una relación de transformación definida en la ecuación (1.1):

$$y[n] = T[x[n]] \tag{1.1}$$

Donde T denota la operación de transformación o procesado que el sistema realiza sobre $x[n]$ para producir una señal de salida $y[n]$. La Figura 1.4 muestra el diagrama general de un sistema discreto:

Figura 1.4 Diagrama de bloque de un sistema discreto.

1.6.4 Linealidad

Un sistema lineal es aquel que cumple con el Principio de Superposición [5]. Esta propiedad se refiere a que si una entrada a un sistema, es la combinación lineal (suma ponderada) de varias señales, entonces la salida correspondiente del mismo, es la combinación lineal de las salidas correspondientes a cada una de dichas entradas. Dicha propiedad se representa mediante la ecuación (1.2)

$$T[\alpha x_1[n] + \beta x_2[n]] = \alpha T[x_1[n]] + \beta T[x_2[n]] \tag{1.2}$$

Donde α y β son constantes cualquiera.

La Figura 1.5 muestra la representación del principio de superposición:

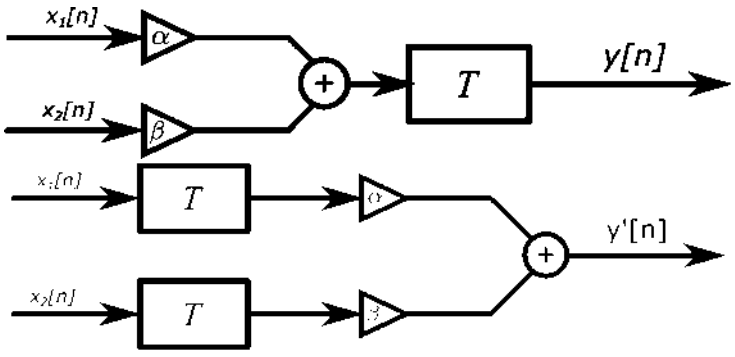

Figura 1.5 Representación gráfica del principio de superposición [5]

1.6.5 Sistema invariante en el tiempo

Se dice que un sistema es invariante en el tiempo si su característica de entrada–salida no cambia con el tiempo [5]. Es decir, *si* $y[n]$ es la salida correspondiente a la entrada $x[n]$, en un sistema invariante en el tiempo, la entrada $x[n - n_0]$ producirá la salida $\mathrm{y[n - n_0]}$. La expresión (1.3) representa la propiedad de invariancia en el tiempo:

$$T\big[x[n]\big] = y[n] \;\Rightarrow\; T\big[x[n-k]\big] = \; y[n-k] \qquad Para\ todo\ n\ y\ k \qquad (1.3)$$

1.6.6 Estabilidad

Un sistema es estable si toda secuencia de entrada limitada (Bounded Input), produce una secuencia limitada de salida (Bounded Output). De aquí que comúnmente se le llame BIBO estable. La entrada $x[n]$ está limitada si existe un valor finito positivo Mx tal que cumpla con la expresión (1.4), la cual representa los límites para que el sistema sea estable:

$$|x[n]| < Mx < \infty, \qquad |y[n]| < My < \infty \qquad \text{Para todo } n \qquad (1.4)$$

Si para determinada secuencia de entrada acotada $x[n]$, la salida no está acotada (es infinita), el sistema se clasifica como *no estable*[5].

1.6.7 Sistemas discretos lineales invariantes en el tiempo (DSLIT)

Cuando un sistema cumple con las propiedades de linealidad e invariancia al tiempo, se dice que son Sistemas Discretos Lineales e Invariantes en el Tiempo (DSLIT)[5][6].

La propiedad de invariación en tiempo implica que si $h[n]$ es la respuesta del impulso $\delta[n]$, entonces la respuesta a $\delta[n-k]$ es $h[n-k]$. Así la salida es representada por la ecuación (1.5) :

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] \qquad (1.5)$$

Como consecuencia de la ecuación (1.5) un sistema lineal invariante en tiempo se caracteriza completamente por su respuesta al impulso $h[n]$. Así teniendo una $h[n]$ determinada, es posible calcular la salida $y[n]$ provocada por una entrada $x[n]$ [5][6].

Se dice que $y[n]$ es la convolución de $x[n]$ con $h[n]$ [6]. Y se representa por la ecuación (1.6):

$$y[n] = x[n] * h[n] \qquad (1.6)$$

Por lo que:

"La respuesta de un DSLIT a una entrada arbitraria $x[n]$ es la convolución de la entrada $x[n]$ con la respuesta al impulso $h[n]$ [5]".

1.6.8 Transformada Z.

La transformada Z de una señal discreta $x[n]$ se define como la serie de sumas o potencias [7][8], como se muestra en la ecuación (1.7):

$$X(z) = \sum_{n=-\infty}^{\infty} x[n]z^{-n} \qquad (1.7)$$

Donde z es una variable compleja.

La transformada Z también puede indicarse por medio de la ecuación (1.8) :

$$X(z) = \mathcal{Z}\{x[n]\} \qquad (1.8)$$

Debido a que la transformada z está definida mediante una serie infinita de potencias de z, puede ocurrir que la serie sea divergente, en ese caso $X(z)$ *no existe*; y solo existirá para aquellos valores de z para los cuales la serie converge.

La transformada z es evaluada en el círculo unitario como se muestra en la Figura 1.6.

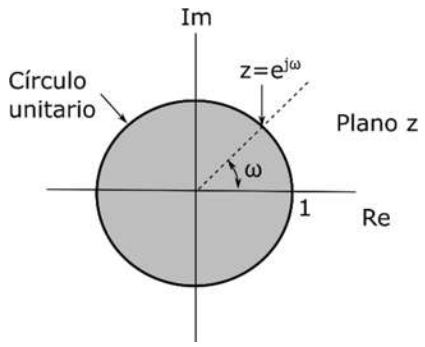

Figura 1.6 El plano z complejo [8].

La transformada z puede ser caracterizada por sus **ceros** (las raíces del polinomio en el numerador) y sus **polos** (las raíces del polinomio en el denominador) [8].

1.6.9 Región de convergencia

El conjunto de valores de z de alguna secuencia para los cuales la transformada $\mathcal{Z}$ converge, se les da el nombre de región de convergencia ROC, llamada así por sus iniciales en inglés [6] [8].

Como ejemplo se tiene la siguiente secuencia definida por $x[n] = a^n u[n], \quad a > 0$

La ROC es el rango de valores de Z para el cual $|az^{-1}| < 1$, o de manera equivalente $|z| > |a|$, determinados por la solución en la ecuación (1.9):

$$X(z) = \frac{1}{1 - az^{-1}} = \frac{z}{z - a} \qquad donde\ |z| > |a|. \qquad (1.9)$$

Donde su ROC se aprecia en la Figura 1.7 :

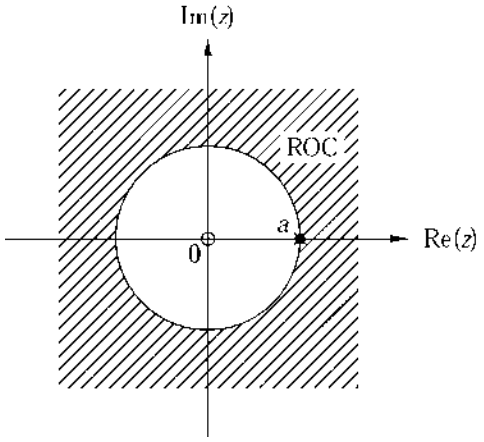

Figura 1.7 ROC para $|z| > |a|$ [5]

Por otro lado, si el sistema es causal, la ROC de $H(z)$ es el exterior de algún círculo de radio r, por lo tanto r debe ser menor que 1. Además, como la ROC no puede contener ningún polo de $H(z)$ se concluye que:

"Un sistema DSLIT causal es BIBO estable si y solo si todos sus polos están en el interior del círculo unitario[5]*."*

1.6.10 Polos y ceros

Todas las transformadas racionales de z^{-1}, son divisiones de polinomios en z^{-1}. En general, una transformada z racional se forma como se ve en la ecuación (1.10):

$$X(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + \ldots + b_M z^{-M}}{a_0 + a_1 z^{-1} + \ldots + a_N z^{-N}} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{\sum_{k=0}^{N} a_k z^{-k}} \qquad (1.10)$$

Donde $a_0, a_1, \ldots a_N$ y $b_0, b_1, \ldots b_N$ son constantes.

En el caso de que $a_0 \neq 0$ y $b_0 \neq 0$ se puede evitar las potencias negativas de z sacando en factor común los términos $b_0 z^{-M}$ y $a_0 z^{-N}$ resultando la ecuación (1.11):

$$X(z) = \frac{B(z)}{A(z)} = \frac{b_0 z^{-M}}{a_0 z^{-N}} * \frac{z^M + \left(\frac{b_1}{b_0}\right) z^{M-1} + \ldots + \left(\frac{b_M}{b_0}\right)}{z^N + \left(\frac{a_1}{a_0}\right) z^{N-1} + \ldots + \left(\frac{a_N}{a_0}\right)} \qquad (1.11)$$

Dado que B(z) y A(z) son polinomios en z, se puede expresar en forma de factores como la ecuación (1.12):

$$X(z) = \frac{B(z)}{A(z)} = \frac{b_0}{a_0} z^{N-M} * \frac{(z - c_1)(z - c_2) \ldots (z - c_M)}{(z - p_1)(z - p_2) \ldots (z - p_N)} \qquad (1.12)$$

$c_1, c_2, \ldots c_M$ son las raíces del numerador.

$p_1, p_2, \ldots p_N$ son las raíces del denominador.

Se puede representar $X(z)$ gráficamente mediante un diagrama de polos y ceros en el plano complejo, que muestra la posición de los polos mediante cruces (×) y la posición de los ceros mediante círculos (○), como ejemplo en la Figura 1.8:

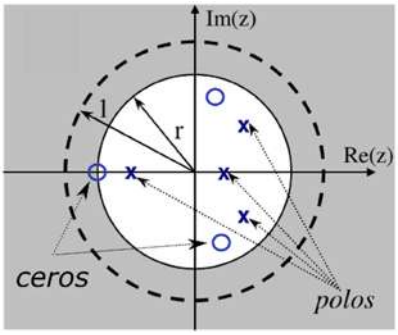

Figura 1.8 Polos y ceros en el plano z complejo

1.6.11 Clasificación de los Sistemas Según su Respuesta al Impulso

La respuesta de un sistema a la cual se le aplica a la entrada un impulso unitario se le conoce como respuesta al impulso.

Un sistema DSLIT queda completamente caracterizado por su respuesta a impulso; esto debido a que cualquier señal puede ser representada como una suma de impulsos retrasados y escalados[9].

En otras palabras, al multiplicar una señal $x[n]$ por $\delta[n-k]$ se extrae la muestra $x[k]$. Si repetimos este proceso para todos los valores de ***k*** se puede expresar la señal original $x[n]$.

La función de un sistema general se rige por la ecuación (1.13):

$$H(z) = \sum_{k=0}^{\infty} h[k]z^{-n} = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}} \qquad (1.13)$$

Los Sistemas DSLIT se pueden clasificar en dos tipos de acuerdo a la duración de su respuesta al impulso:

- **Sistema FIR** (Finite Impulse Response) si su respuesta al impulso es de longitud finita, cuyos polos están en el origen o en el infinito.
- **Sistema IIR** (Infinite Impulse Response) si su respuesta al impulso es de longitud infinita, los polos estarán en ubicaciones diferentes del origen o en el infinito.

1.6.12 Sistema FIR

La respuesta al impulso de un sistema FIR es cero fuera de un intervalo finito de valores de *n*, como se muestra en la ecuación (1.14):

$$h[n] = 0 \; para \; n < 0 \;\; y \;\; n \geq N \qquad (1.14)$$

Por lo tanto, la convolución para un sistema FIR toma la forma de la ecuación (1.15):

$$y[n] = \sum_{k=0}^{N-1} h[k]x[n-k] \qquad (1.15)$$

Donde (N - 1) es el orden del filtro.

Por lo tanto, la función para el sistema tomando la transformada z de $h[k]$ es la ecuación (1.16):

$$H(z) = \sum_{k=0}^{N-1} h[k]z^{-k} \qquad (1.16)$$

Los filtros FIR sólo presentan polos en el origen, por lo que siempre son estables. Sobre el posicionamiento de los ceros, en los filtros de fase lineal los ceros se dan en pares recíprocos, es decir, si z_0 es una raíz del polinomio $H(z)$, también lo será ${z_0}^{-1}$ [10]. En la Figura 1.9 se aprecia el diagrama de bloque de un sistema FIR:

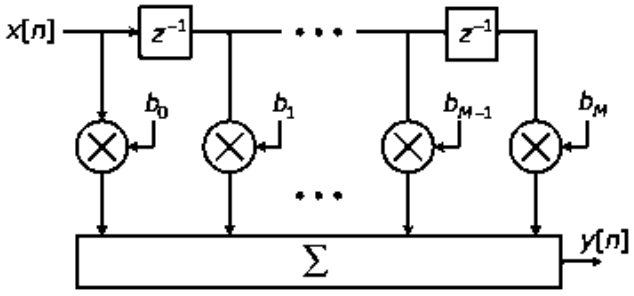

Figura 1.9 Diagrama de bloques de un sistema FIR [10]

1.6.13 Sistema IIR

Un sistema lineal e invariante en el tiempo IIR tiene una respuesta al impulso de duración infinita[10]. Su salida se basa en la fórmula de la convolución de la respuesta al impulso con la señal de entrada como se muestra en la ecuación (1.17):

$$y[n] = \sum_{k=0}^{\infty} h[k]x[n-k] \tag{1.17}$$

Dado que la sumatoria implica todas las muestras de entrada actual y de todas las pasadas, se dice que el sistema tiene una memoria infinita. La función del sistema se rige por la ecuación (1.13), donde su numerador son los ceros y su denominador son los polos del sistema, por lo cual:

Si los coeficientes a_k y b_k de $H(Z)$ son reales, los polos y los ceros aparece por parejas conjugadas o son reales. Para que el sistema sea estable y causal, todos los polos deben situarse exactamente en el interior del círculo unitario, no habiendo restricción para los ceros[5].

Capítulo 2
Desarrollo

En el presente capítulo se muestra el desarrollo de la herramienta, donde se aplica dos formas para la creación de filtros digitales:

1.- Método de manipulación de polos y ceros.

2.- Mediante el uso de los comandos *wfir* e *iir* de SCILAB® en una interfaz gráfica de usuario.

Enfatizando que la segunda forma de diseñar los filtros se hace uso de comandos ya existentes en la plataforma SCILAB®, una vez diseñado el filtro de la segunda manera, se muestran los polos y ceros y estos a su vez se pueden manipular integrando así el método de colación de polos y ceros como se muestra en la Figura 2.1, que representa el esquema general de la herramienta.

Figura 2.1 Esquema general de la herramienta para diseñar filtros digitales.

2.1 Método de manipulación de polos y ceros

Un cero es un punto del plano z en el que la función del sistema se hace nula, mientras que en un polo el valor de dicha función tiende a infinito.

Este método consiste en colocar los polos y ceros en el plano z de manera que se obtenga un módulo de la respuesta en frecuencia aproximado al que se desea. Como la respuesta en frecuencia se evalúa en el círculo de radio unitario (llamado también círculo unitario) del plano z, si existe un polo próximo a dicha circunferencia, el valor de su magnitud será elevado en aquellos puntos cercanos a dicho polo. Si se coloca un cero cercano a dicha circunferencia, se conseguirá reducir el valor de la magnitud. Así se puede diseñar filtros de una manera sencilla (filtro pasa bajas, pasa altas ,pasa banda, rechaza banda) [11].

2.2 Uso de la plataforma SCILAB®

Figura 2.2 Logotipo de SCILAB ® [12]

SCILAB® es una plataforma de software libre de código abierto para computación científica, orientado al cálculo numérico, a las operaciones matriciales y especialmente a las aplicaciones científicas y de ingeniería, la Figura 2.2 representa su logotipo característico.

Su valor principal radica en los cientos de funciones tanto de propósito general como especializadas que posee, así como en sus posibilidades de interfaz gráfica.

SCILAB® posee además un lenguaje de programación propio, muy próximo a los habituales en cálculo numérico (Fortran, C, etc.) que permite al usuario escribir sus propios *scripts (*conjunto de comandos escritos en un archivo que se pueden ejecutar con una única orden) para resolver un problema concreto y también escribir nuevas *funciones*, por ejemplo, sus propios algoritmos [12].

Las características de SCILAB® incluyen análisis numérico, visualización 2D y 3D, optimización, análisis estadístico, diseño y análisis de sistemas dinámicos, procesamiento de señales, e interfaces con Fortran, Java, C y C++. Mientras que la herramienta *Xcos* permite una interfaz gráfica para el diseño de modelos [13].

2.3 Interfaz gráfica de usuario (GUI).

La interfaz gráfica de usuario GUI (por sus siglas en inglés *Graphical User Interface*) es una forma en que el usuario interactúa con el programa o el sistema operativo de una computadora. Consiste en el diseño de una herramienta que permita la interacción con el sistema de una manera gráfica a través del uso de un conjunto de imágenes y objetos pictóricos (iconos, ventanas, etc.). Tiene que ser lo más simple posible y fácil de utilizar.

Para este trabajo se realizó una GUI que permite colocar polos y ceros dentro del plano z por medio del puntero, a su vez permite el ajuste de parámetros del mismo, observando la respuesta en frecuencia en una gráfica. Así mismo, se implementó otra ventana para el diseño de manera sencilla de filtros FIR e IIR.

2.4 GUI builder

A manera de referencia, existen diferentes lenguajes de programación que permiten crear una interfaz gráfica (C, visual Basic, etc.), los cuales permiten usar herramientas como botones, botón deslizante, etc. SCILAB® nos permite realizar interfaces de manera sencilla usando GUI builder.

GUI builder es un *toolbox* de SCILAB® que consiste en un entorno de programación gráfica, el cual permite realizar y ejecutar programas de simulación de forma simple, y genera el código de manera automática [14].

Para el manejo de la herramienta guibuilder se puede consultar por medio de la referencia [15].

2.5 Construcción de la GUI en SCILAB®

La forma de implementar las interfaces gráficas de usuario en SCILAB® es crear objetos y definir las acciones que cada uno va a realizar. Al usar GUI builder se obtiene un archivo:

- Archivo con extensión ***sce***: Contiene las funciones y los manejadores de la interfaz gráfica.

Los manejadores y funciones están unidas a través de subrutinas *(callback)*. Un *callback* se define como la acción que llevará a cabo un objeto de la interfaz gráfica cuando el usuario lo active. Como ejemplo, si en la ventana existe un botón, al presionarlo se ejecutarán las instrucciones de la función asociada con ese botón, a eso se le conoce como la función callback.

Al crear el archivo con extensión ***sce*** se necesita seleccionarlo en la ventana de *SciNotes* y presionar el botón de *ejecutar.*

Todas las propiedades de cada elemento (posición, color, estilo, etc.) se almacenan en una estructura nombrada *handles*, cada objeto creado tendrá su propio manejador (*handler)* que a su vez son parte del manejador de la figura (Ventana gráfica). Con la asignación de manejador individual se garantiza que cualquier cambio o asignación de propiedades o variables quede almacenado.

Para esta herramienta se hizo uso de dos Figuras (ventanas gráficas) que consisten en lo siguiente:

- Interfaz gráfica principal para la colocación de polos y ceros en el plano z, así como los parámetros (ganancia, coordenadas polares y rectangulares), y gráfica de respuesta en frecuencia.
- Interfaz gráfica para ingresar los parámetros del filtro a diseñar (tipo de filtro, frecuencia de muestreo (Hz), orden del filtro, frecuencia baja de corte, frecuencia alta de corte, sistema FIR o IIR).

2.6 Primer forma para creación de filtros digitales

En esta sección se hace uso del método de ubicación de polos y ceros en la ventana gráfica principal, donde se ubican los polos y ceros por medio del puntero, permitiendo modificar sus parámetros una vez se hayan colocado y seleccionado previamente, mostrando su respuesta en frecuencia del filtro diseñado.

Dentro de esta sección se mostrará las funciones que constituyen la ventana principal:

2.6.1 Creación del plano z complejo

Para la creación del plano z se utiliza una gráfica en 2d (comando *plot2d*), introduciendo un círculo de radio unitario con origen en cero como se muestra en el Pseudocódigo 1.

Existe el comando *plzr* el cual muestra los polos y ceros dentro del plano z, pero estos polos y ceros no pueden ser modificados, únicamente se muestra la gráfica del círculo unitario por lo cual no cumple con el objetivo de esta herramienta.

Pseudocódigo 1 Gráfica del plano z complejo

```
Nombre: Gráfica circulo
Inicio
    Manejador1 ← figura plano z
    Dibujar plano
    Dibujar un círculo en el origen.
    Colocar gráfica escalada.
Fin
```

2.6.2 **Colocación de polos y ceros**

En el caso de la ubicación de polos y ceros es necesario que las coordenadas que se obtengan sean en base a la ventana del plano z, esto es, que el origen (0,0) esté en el centro del círculo unitario y no en la ventana principal, por lo cual el comando que cumple con este requisito es *xchange*. Al momento de tomar el manejador de la gráfica del plano z, automáticamente las coordenadas son mostradas en base al origen de esta gráfica.

Para colocar un polo o cero es necesario tener capturado el manejador de la gráfica del plano z y tener el control de los eventos o click realizado con el ratón. Existen dos comandos para poder controlar el evento y el click:

xclick, Espera un click del ratón o un evento en una ventana gráfica, previniendo que el callback del objeto sea ejecutado.

xgetmouse, obtiene el evento del ratón, así como su posición dentro de la ventana gráfica, no previene que el callback del objeto sea ejecutado.

Debido a que se espera crear polos o ceros por cada click (izquierdo) sin detener el programa, existe otra forma de detectar los eventos, esto es mediante funciones para eventos del manejador:

event handler function

Mediante el uso de esta estructura de función cada vez que se realiza un evento la función proporciona el tipo de evento, las coordenadas y el número de ventana donde se está realizando el evento. Permitiendo así crear polos o ceros (marcadores) por cada click realizado, el Pseudocódigo 2 muestra la forma para colocar polos y ceros.

Pseudocódigo 2 Dibuja polo o cero

```
Inicio:
Si evento → click izquierdo
      Dibujar objeto
      Manejador ← objeto
      Crea estructura
      Dibuja conjugado
      Manejador2 ← conjugado
      Crea estructura
      Asigna propiedad conjugado en ambos
      Enlista estructuras
      Calcula polinomio
      Grafica respuesta en frecuencia
Fin
```

2.6.3 **Lista de estructuras**

Para enlistar los polos y ceros es necesario hacerlo de una manera ordenada, dividiendo en 4 cuadrantes el plano z como se muestra en la Figura 2.3, por lo cual son cuatro listas de una lista general. Esto provoca que al momento de hacer una búsqueda de la estructura del polo o cero, no tenga que buscar entre todas las estructuras almacenadas, sino únicamente de las del cuadrante en el que se está buscando.

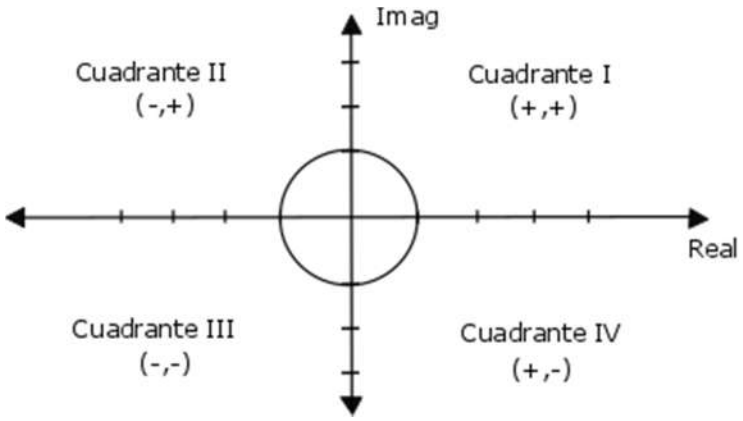

Figura 2.3 Distribución de cuadrantes para las listas.

El Pseudocódigo 3 muestra el proceso de la función que se generó para ubicar en el cuadrante adecuado:

Pseudocódigo 3 Ubicar cuadrante

```
Nombre: ubicaCuadrante
Entradas: coordenadas x, y
Inicio:
Si X>=0 & Y >= 0
      Cuadrante ← 1
```

```
Si   X < 0 & Y >= 0
      Cuadrante ← 2
Si X < 0 & Y < 0
      Cuadrante ← 3
Si X>0 & Y < 0
      Cuadrante ← 4
Fin programa
```

2.6.4 **Buscar polo o cero**

Para la búsqueda del manejador del polo o cero situado dentro de la gráfica del plano z se requiere de un rango a partir de las coordenadas del pixel seleccionado; esto es debido a que el polo o cero seleccionado consta de un rango de pixeles que contiene dentro de su figura (“x”, “o”) lo cual permite no tener que seleccionar el pixel exacto donde se colocó el polo o cero como se muestra en el Pseudocódigo 4 de la función para buscar.

Pseudocódigo 4 Buscar polo o cero

```
Nombre: Busca
Entrada: coordenadas x, y
Salida: Cuadrante, índice
Inicio:
 Crear límite superior e inferior de x
Ubica cuadrante ← número de cuadrante
Desde i=1 hasta cantidad de estructuras en cuadrante
      Si x está entre los límites
            Crear límites superior e inferior de y
            Si y está entre los límites
                  índice ←i
```

```
            finSi
        finSi
    Repetir
    Fin
```

2.6.5 **Movimiento de polo o cero en la gráfica del plano z**

Para la manipulación del polo o cero y su conjugado es necesario tener en cuenta el evento de sostener el click izquierdo, posteriormente detectar el movimiento del puntero, y finalmente detectar cuando se suelta el botón izquierdo.

Al sostener el botón izquierdo comenzará la búsqueda para saber si donde se hizo clic existe un polo o cero, una vez confirmado, si el usuario comienza a mover el puntero, de forma inmediata se actualiza la figura (“x” y “o”) junto a su conjugado si se encuentra en existencia, finalmente se detecta cuando se suelta el botón y se procede a calcular el nuevo polinomio y obtener su respuesta en frecuencia.

Para poder detectar el evento de movimiento junto a click izquierdo sostenido es necesario el comando *xgetmouse* junto una configuración previa del mismo mostrada a continuación:

```
rep=xgetmouse ([%T %T]);
```

Esta configuración es necesaria para poder detectar el evento -5 (botón izquierdo sostenido junto a movimiento).

Reubicación de cuadrante

Debido a que los manejadores de cada polo y cero están asignados a cuadrantes específicos (I, II, III, IV) si las coordenadas nuevas son pertenecientes a otro cuadrante produce un error puesto que el polo o cero necesita tener el número de cuadrante adecuado, para solucionar esto se requiere de los siguientes pasos:

1.- Encontrar su estructura en la lista y guardarlo en una variable temporal.

2.- Borrar su estructura de la lista.

3.- Dibujar en base a la estructura temporal.

4.- Al terminar de mover, enlista la estructura.

Estos pasos aplican de igual forma a su conjugado si es que existe. Con esto se evita el problema de los cuadrantes, y a su vez se reubica el polo o cero al cuadrante correcto, permitiendo manipular los polos y ceros en toda la gráfica del plano z. La función de movimiento se puede observar en el siguiente Pseudocódigo 5:

Pseudocódigo 5 Movimiento de polo o cero

```
Nombre: moverx
Entradas: Coordenadas x, y, evento
Inicio:
Si botón es dejado apretado
   Buscar el manejador en lista
   Si existe polo o cero
      Regresa el cuadrante y número en la lista
      Copia en un temporal
      Si existe conjugado
         Busca el manejador del conjugado en lista
         Copia conjugado en un temporal
      FinSi
   FinSi
   Borra de lista el polo o cero seleccionado
   Si existe conjugado
```

```
      Borrar conjugado
    Mientras se esté moviendo el puntero
      Redibuja el polo o cero de temporal
  Si Actualización está seleccionado
          Crear nuevo polinomio por cada movimiento de puntero
          Grafica respuesta en frecuencia
      FinSi
    Fin mientras
    Si el botón es soltado
      Enlista el polo o cero de temporal
      Si existe conjugado
        Enlista conjugado de temporal
      Fin si
      Crea nuevo polinomio
      Grafica respuesta en frecuencia
    Fin Si
Fin Si
Fin
```

2.6.6 **Actualización de respuesta en frecuencia en base al movimiento**

Para tener la respuesta en frecuencia en base a cada movimiento del polo o cero es necesario crear una función que permita obtener el sistema (filtro) en base a la multiplicación de las entidades (monomios) de cada estructura guardada en las listas y Añadiendo el valor actual del polo o cero que se encuentra en movimiento, esto debido a

que el polo o cero momentáneamente no se encuentra enlistado (Debido a la reubicación del polo o cero) hasta que se suelta el click izquierdo.

Esta función sólo es llamada dentro de la función *moverx* por lo cual sólo es funcional mientras exista movimiento del puntero. El Pseudocódigo 6 muestra la función de actualizar por movimiento:

Pseudocódigo 6 Actualización por movimiento

```
Nombre: auto_upd
Inicio
Desde cuadrante 1 hasta 4
       Desde 1 hasta cantidad de estructuras en lista
       Si es polo
              Si no está en el origen
                Polinomio polos ← polinomio polos  *  monomio
                Si existe conjugado
                 Polinomio polos ← polinomio polos  *  monomio conjugado
               FinSi
              FinSi
      Si no lo es
             polinomio de ceros  *  monomio
             Si existe conjugado
               Polinomio ceros ←polinomio ceros  *  monomio conjugado
             Finsi
      FinSi
     Repetir
Repetir
Si temporal es polo
      Polinomio polos ← polinomio  polos  *  monomio de temporal
```

```
      Polinomio polos ← polinomio polos  *  monomio conjugado de temporal
Si no lo es
      Polinomio ceros ← polinomio ceros  *  monomio del temporal
      Polinomio ceros ← polinomio ceros  *  monomio conjugado de temporal
finSi
Grafica respuesta en frecuencia
Fin
```

2.6.7 **Cálculo del sistema (Filtro)**

El cálculo del sistema se realiza en base a la ecuación (1.13) , el sistema se conforma de dos polinomios si su sistema tiene una respuesta IIR, esto es, que se ve en forma de polinomio del numerador (ceros) entre el polinomio del denominador (polos). En el caso de un sistema FIR se sabe que los polos se encuentran en el origen, por lo cual únicamente se calcula el polinomio de ceros.

Para obtener el sistema se calcula los polinomios. Cada polo y cero creado contiene su entidad como monomio, se identifica su respectivo polinomio (numerador o denominador) y se realiza la multiplicación para así obtener el nuevo sistema (Filtro) como se muestra en el siguiente Pseudocódigo 7:

Pseudocódigo 7 Cálculo del polinomio

```
 Nombre: polinomio
 Inicio
 Desde cuadrante 1 : 4
        Desde 1 : cantidad de estructuras en lista
        Si es polo
              Si no está en el origen
               Polinomio polos ← polinomio polos  *  monomio
```

```
                    Si existe conjugado
                     Polinomio polos ← polinomio polos * monomio
conjugado
                    FinSi
                  FinSi
              Si no lo es
                    Polinomio ceros ← polinomio ceros  *  monomio
                    Si existe conjugado
                     Polinomio ceros ← polinomio ceros * monomio
conjugado
                    FinSi
            FinSi
            Repetir
      Repetir
      Grafica respuesta en frecuencia
      Fin
```

2.6.8 **Gráfica de respuesta en frecuencia**

La gráfica de respuesta en frecuencia requiere del sistema obtenido (Filtro diseñado), esto es, en el caso de un sistema FIR los coeficientes del polinomio de ceros. En el caso de un sistema IIR necesita los coeficientes de dos polinomios

- Polinomio de polos (Denominador).
- Polinomio de ceros (Numerador).

Una vez diseñado el sistema, se procede a obtener la respuesta en frecuencia mediante el comando *repfreq*. Obtenida la respuesta en frecuencia se procede a calcular la frecuencia máxima basado en el teorema de muestreo de Nyquist - Shannon [5] visto en la ecuación (2.1) :

$$f_{máx} = \frac{f_{muestreo}}{2} \qquad (2.1)$$

Donde la frecuencia de muestreo es obtenida por medio del GUI para el diseño e filtros o por la ventana principal en la sección de frecuencia de muestreo. Ya realizado los cálculos el vector de la respuesta en frecuencia se transforma a decibeles (dB). Logrando así una gráfica con representación vertical en decibeles y el eje horizontal en frecuencia (Hz).

El Pseudocódigo 8 muestra el proceso para graficar su respuesta en frecuencia:

Pseudocódigo 8 Gráfica de respuesta en frecuencia

```
Nombre: gainfunction
Entradas: Numerador, Denominador.
Inicio:
     Genera el sistema (numerador entre denominador)
     Manejador ← Gráfica de respuesta en frecuencia
     Calcula escala de frecuencia (eje horizontal)
     Calcula respuesta en frecuencia
     Limpia gráfica
     Calcula frecuencia máxima
     Transforma a dB la respuesta en frecuencia
     Grafica frecuencia en respuesta
     Coloca nombre a ejes
Fin de programa
```

2.6.9 **Actualización del polo o cero mediante valor de coordenadas**

Esta herramienta cuenta con la opción de cambiar los parámetros del polo o cero mediante el ingreso de nuevas coordenadas polares o rectangulares siempre y cuando previamente se haya seleccionado un polo o cero. Debido a que SCILAB® no cuenta con un

comando que permita cambiar de coordenadas polares a rectangulares y viceversa se crearon dos funciones con extensión *sci* que permiten hacerlo, ver anexo A y B. Las dos formas para ingresar coordenadas son las siguientes:

***Coordenadas rectangulares*:** Esta opción consiste en ingresar un nuevo valor de coordenadas x o y, al momento de ingresar las coordenadas se actualiza la posición del polo o cero seleccionado en el plano z, posteriormente se calcula el nuevo sistema (filtro) y a su vez se muestra la nueva gráfica de respuesta en frecuencia como se muestra en el Pseudocódigo 9:

Pseudocódigo 9 Manipulación del polo o cero mediante valor de coordenadas rectangulares

```
Nombres: up_x, up_y
Inicio:
    Leer valor
    Establece límite de valor (máximo de 10)
    Temporal ← polo o cero
    Si existe conjugado
        Temporal_conj ← conjugado
    FinSi
    Realiza actualización de posición en plano z
    Borra el polo o cero de la lista.
    Enlista la estructura temporal
    Enlista su conjugado
    Calcula polinomios
    Grafica respuesta en frecuencia
Fin de programa
```

La diferencia entre las dos funciones es la asignación del manejador correspondiente, esto es, el objeto correspondiente al recuadro de texto (“coordenadas X”, “coordenadas y”).

Coordenadas polares: En esta opción los valores de coordenadas rectangulares son transformados a polares mostrando la magnitud y fase de la ubicación del polo o cero. Una vez transformado, se puede modificar sus valores de magnitud y fase (en grados), ya ingresado los nuevos parámetros se transforma nuevamente a coordenadas rectangulares (siendo transparente para el usuario), se actualiza su posición en el plano z y se procede a calcular el nuevo sistema (filtro), mostrando su respuesta en frecuencia. El Pseudocódigo 10 muestra el proceso para la manipulación por medio de coordenadas polares:

Pseudocódigo 10 Manipulación de polo o cero mediante coordenadas polares

```
Nombres: up_magnitud, up_ángulo
Inicio:
   Leer valor
   Transformar polar a rectangular
   Temporal ← polo o cero
   si existe conjugado
      Realiza actualización de posición en plano z
   FinSi
   Borra el polo o cero de la lista.
   Enlista la estructura temporal
   si existe conjugado
       enlista conjugado
   FinSi
   Calcula polinomio
   Grafica respuesta en frecuencia
```

```
   Busca el polo o cero recién enlistado
Fin programa
```

La diferencia entre las dos funciones es la asignación del manejador correspondiente. Esto es, el objeto correspondiente al recuadro de texto (Magnitud, Fase).

2.6.10 Ganancia

Esta herramienta permite calcular la ganancia necesaria para obtener una ganancia unitaria en la banda de paso. El proceso consiste en encontrar el punto máximo de la respuesta en frecuencia, una vez obtenido se obtiene su recíproco, y esa será la corrección de ganancia que se aplicará al sistema, esto se aprecia en la ecuación (2.2):

$$Ganancia = \frac{1}{Punto\ máximo} \qquad (2.2)$$

Así el proceso para la función es el siguiente Pseudocódigo 11:

Pseudocódigo 11 Ganancia necesaria

```
Nombre: ganan_cero
Inicio:
   Max ← máximo valor de respuesta en frecuencia
   Ganancia ← 1/Máximo valor
   Respuesta en frecuencia *  Ganancia
   Muestra la ganancia en ventana principal
Fin programa
```

2.7 Segunda forma para la creación de filtros digitales

Esta forma se basa en una interfaz gráfica de usuario para ingresar los parámetros del filtro, donde se usa los comandos ya existentes en SCILAB® para diseñar filtros *wfir* e *iir*, permitiendo al usuario diseñar en base a parámetros. Una vez obtenido el sistema se ubican los polos y ceros dentro de la ventana gráfica principal en la gráfica del plano z, y a su vez se grafica la respuesta en frecuencia.

Los polos y ceros ubicados pueden ser modificados mediante el uso de la primera forma e incluso agregar o quitar polos y ceros, esto con el objetivo de cumplir las expectativas de ser una herramienta didáctica.

Para la adquisición de los parámetros basta con adquirir el valor numérico escrito en el campo y así guardarlo en una variable para el uso posterior.

Las siguientes secciones muestran las funciones principales para el funcionamiento de la segunda forma de creación de filtros digitales:

2.7.1 Cálculo de la frecuencia normalizada

Una vez obtenido los parámetros del filtro que se desea diseñar es necesario calcular la frecuencia de corte, esto es debido a que ambos comandos (*wfir* e *iir*) necesitan la frecuencia normalizada que está en el rango de 0 a 0.5. Para el cálculo se aplica la división como se ve en la ecuación (2.3):

$$\frac{frecuencia\ \ corte}{frecuencia\ \ muestreo} \qquad (2.3)$$

Con esto se obtiene el valor en porcentaje de la frecuencia de corte con respecto a la frecuencia de muestreo. Una vez calculado, se debe determinar si es un pasa bajas o pasa altas los cuales sólo contienen una frecuencia de corte, pero si es un pasa bajas o rechaza banda necesitará dos frecuencias de corte (baja y alta), como se muestra en el Pseudocódigo 12 de la función para obtener el vector para la frecuencia de corte:

Pseudocódigo 12 Cálculo de la frecuencia de corte

```
Nombre: edit_frec_corte
Inicio:
   Frec_corte_baja / frecuencia de muestreo
   Frec_corte_alta / frecuencia de muestreo
   Si es filtro pasa bajas | es filtro pasa altas
          Frec_corte ← [frec 0]
   Si no lo es
          Frec_corte ← [frec_baja frec_alta]
Fin programa
```

2.7.2 **Colocar polos y ceros del filtro diseñado**

Una vez diseñado el filtro se obtiene sus raíces por medio del comando *roots* y se procede a colocar los marcadores de polos y ceros en la gráfica del plano z. Como primer paso se identifica si es un polo o cero, después se dibuja el polo o cero y se guarda su estructura. Para la segunda vez que se dibuja un polo o cero se pregunta si es conjugado al anterior, si lo es, se asigna su propiedad conjugado a ambos, y así se repite hasta terminar de colocar todos los marcadores.

En algunos casos el conjugado no se asigna correctamente al que le corresponde, esto debido a que encuentra en la lista otro que podría ser su conjugado, pero ya podría tener asignado otro conjugado al que se encontró. Para evitar que el conjugado se asigne a otro polo o cero el cual ya contiene su conjugado, se necesita buscar el polo o cero que no contenga propiedad de conjugado con otro, para esto es necesario una búsqueda uno por uno hasta encontrarlo.

El Pseudocódigo 13 muestra el proceso para colocar polos y ceros, así como asignar la propiedad de conjugado:

Pseudocódigo 13 Colocación de polos y ceros del filtro diseñado

```
Nombre: coloca
Inicio:
  Grafica marcador (polo o cero)
  Si es conjugado del anterior (x anterior y anterior)
         Busca el polo o cero anterior
         Busca polo o cero sin conjugado
         Crea estructura del conjugado
         Asigna a ambos la propiedad de conjugados
         Enlista conjugado
         Limpia variables
  Si no lo es
         Crea estructura del polo o cero
         x anterior ← x
         y anterior ← y
         enlista polo o cero
Fin programa
```

2.7.3 **Distribución de los polos y ceros**

Para ubicar de manera correcta los polos y ceros es necesario primero saber de qué tipo de sistema se trata (FIR o IIR), posteriormente se identifica qué tipo de marcador (polo o cero) es, y se indica su valor para finalmente colocarlo y realizar el cálculo del filtro obtenido, mostrando su respuesta en frecuencia en la ventana principal.

Para colocar los marcadores es necesario hacer un ciclo para ir colocando uno por uno, una vez se haya ubicado todos los polo y ceros, se procede a hacer el cálculo y mostrar la respuesta en frecuencia del filtro diseñado.

El Pseudocódigo 14 muestra el proceso para definir cada uno de los polos y ceros que se ubican en el plano z complejo:

Pseudocódigo 14 Distribución de polos y ceros

```
Nombre:dis_fil
Inicio:
  Si es sistema FIR
       Tipo ← ceros
       Desde i=1:cantidad de ceros
          X ← real de raíz
          Y ← Imaginaria de raíz
          Coloca
          i = i + 1
       Repetir
       Tipo ← polos
       Desde j=1: cantidad de polos
          X ← 0
          Y ← 0
          Coloca
          j=j + 1
     Repetir
  Si no lo es
      Tipo ← ceros
       Desde i=1:cantidad de ceros
          X ← real de raíz
          Y ← Imaginario de raíz
          Coloca
          i = i + 1
```

```
        Repetir
      Tipo ← polos
        Desde j=1: cantidad de polos
          X ← real de raíz
          Y ← imaginario de raíz
          Coloca
          j=j + 1
        Repetir
  Fin si
Fin programa
```

2.8 Interfaz gráfica principal

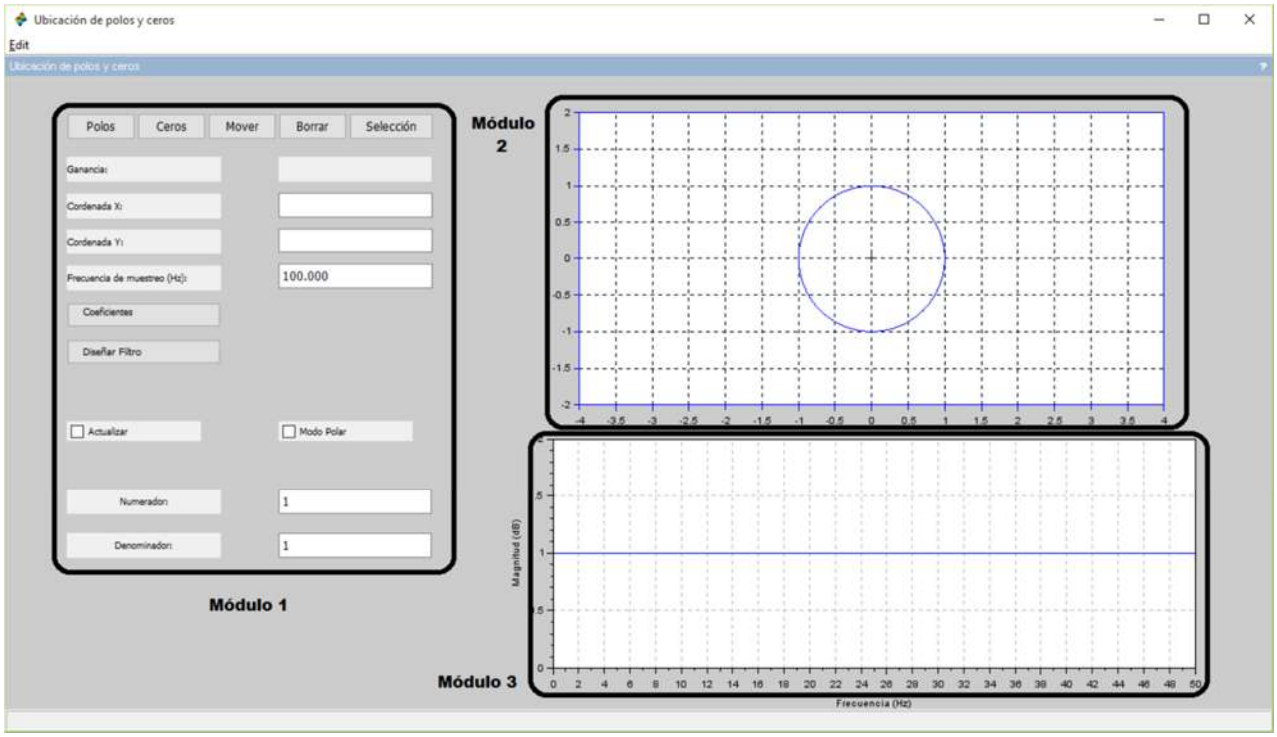

Figura 2.4 Ventana gráfica principal de la herramienta

La Figura 2.4 muestra la estructura de la GUI principal. Esta estructura está divida en 3 módulos:

- 1.- Parámetros del polo o cero seleccionado, sección donde se puede manipular el polo o cero en base a sus parámetros de manera polar o rectangular.
- 2.- Gráfica del círculo unitario, donde se puede manipular en base a movimientos dentro de la gráfica la ubicación del polo o cero seleccionado.
- 3.- Gráfica de respuesta en frecuencia, muestra de manera actualizable la respuesta que tiene el sistema creado.

Módulo 1, parámetros del filtro:

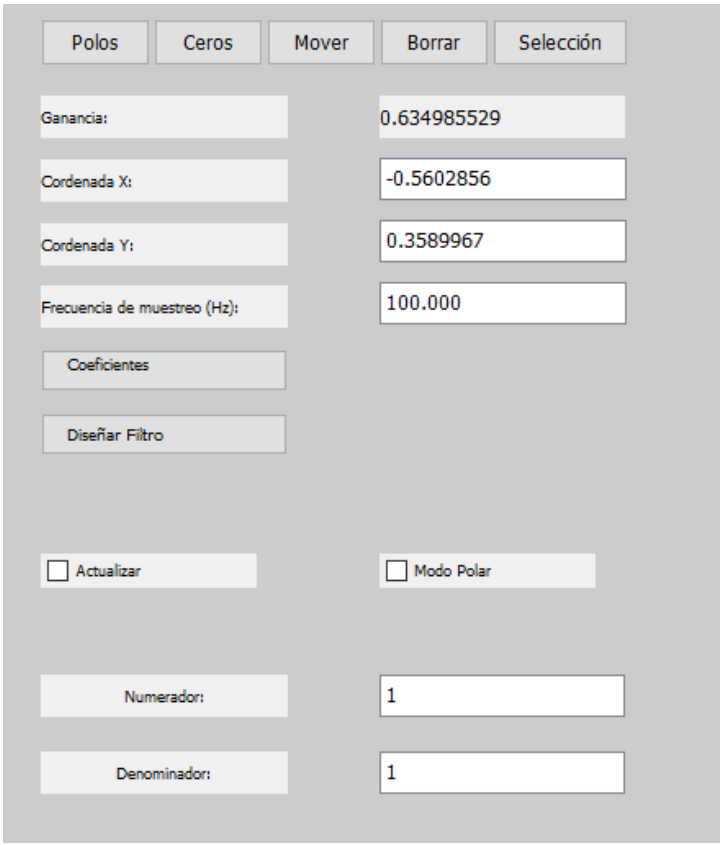

Figura 2.5 Módulo 1 parámetros del filtro digital

En el módulo 1 de parámetros del polo o cero mostrado en la Figura 2.5, se encuentra los siguientes recuadros:

- **Polos:** botón para crear marcador de polo (“x”) dentro del plano Z complejo.
- **Ceros:** botón para crear marcador de cero (“o”) dentro del plano Z complejo.
- **Mover:** Botón que permite cambiar de posición un marcador mediante el movimiento del puntero, así como a su conjugado si existe.
- **Borrar:** Permite eliminar un polo o cero, al igual que su conjugado si existe.
- **Selección:** Permite seleccionar un polo o cero y su conjugado si existe.
- **Ganancia:** Muestra la ganancia necesaria para que su magnitud en dB alcance la posición cero como punto máximo en la gráfica de respuesta en frecuencia.
- **Coordenadas x, y:** Muestran las coordenadas actuales del polo o cero seleccionado, permitiendo hacer cambios por medio de valores numéricos.
- **Frecuencia de muestreo:** Muestra la frecuencia de muestreo actual, permitiendo hacer cambios por medio de valores numéricos.
- **Coeficiente:** Botón que permite mostrar los coeficientes del filtro obtenido.
- **Diseñar Filtro:** Botón que muestra la ventana secundaria para el diseño de filtros mediante una interfaz gráfica.
- **Actualizar:** Selección que permite actualizar la respuesta en frecuencia por cada movimiento de un polo o cero en el plano z complejo.
- **Modo polar:** Permite intercambiar coordenadas rectangulares, por coordenadas polares.

- **Numerador:** Coeficientes del polinomio de ceros del filtro diseñado
- **Denominador:** Coeficientes del polinomio de polos del filtro diseñado.

Módulo 2, plano z complejo:

El módulo 2 es donde se encuentra el círculo unitario y son colocados los polos y ceros de manera manual o del filtro diseñado, la Figura 2.6 muestra un ejemplo de la primera forma para diseñar filtros digitales, donde son colocados los polos y ceros de manera manual.

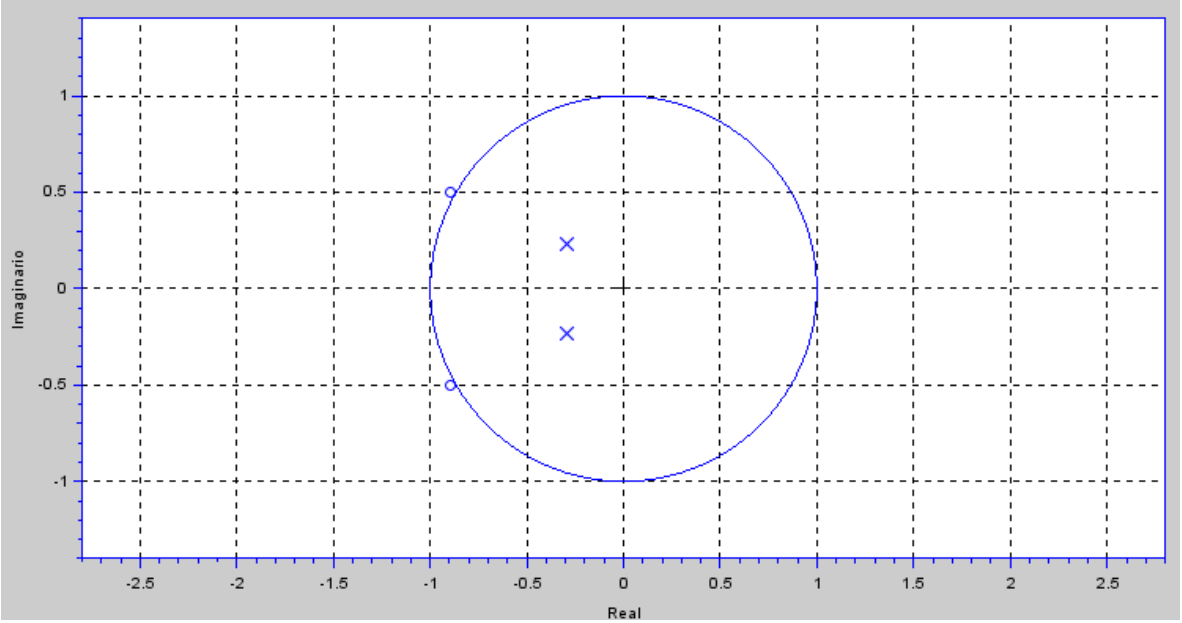

Figura 2.6 Módulo 2. Plano z complejo con polos y ceros

Módulo 3, respuesta en frecuencia:

El módulo 3 muestra la respuesta en frecuencia obtenida del filtro diseñado, a manera de ejemplo, la Figura 2.7 representa la respuesta en frecuencia cuando no se tiene ningún polo o cero en el plano z como viene por default al abrir la herramienta.

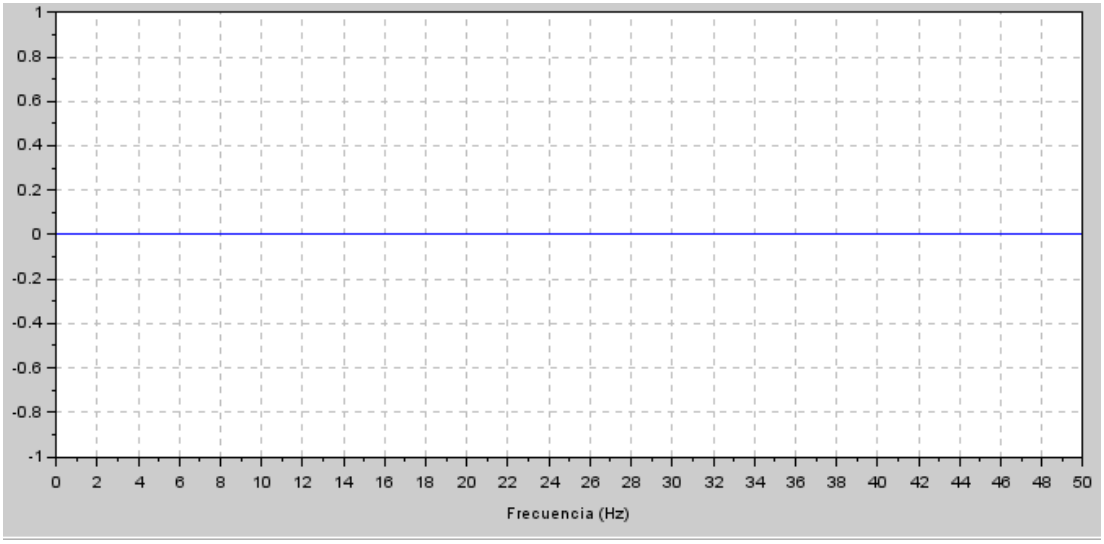

Figura 2.7 Módulo 3, Respuesta en frecuencia

2.9 Interfaz gráfica secundaria para el diseño de filtro digital

Figura 2.8 Interfaz gráfica para el diseño de filtros digitales

En la Figura 2.8 se muestra la ventana utilizada para la segunda forma de diseño de filtros digitales, esto es, mediante los parámetros específicos del filtro a diseñar.

La interfaz gráfica se compone de lo siguiente:

- **FIR:** Selección para determinar que sea un sistema FIR.
- **IIR:** Selección para determinar que sea un sistema IIR.
- **Tipo de filtro:** Lista de los filtros disponibles (pasa bajas, pasa altas, pasa banda y rechaza banda).

- **Frecuencia de muestreo:** Campo para ingresar la frecuencia de muestreo en Hertz.
- **Orden del filtro:** campo para ingresar el orden del filtro en valor positivo.
- **Frecuencia baja de corte:** Campo para ingresar la frecuencia de corte en pasaba bajas y pasa altas, así como definir la frecuencia baja en pasa banda y rechaza banda.
- **Frecuencia alta de corte:** Campo para definir la frecuencia de corte alta en pasa banda y rechaza banda.
- **Diseñar:** Botón para finalizar el diseño del filtro y regresar a la ventana principal.

Capítulo 3

Pruebas y Resultados

En esta sección se muestra diferentes pruebas para probar el funcionamiento de la herramienta, así como el resultado que se obtuvo de su implementación. Para tal fin se muestran ejemplos de filtros, con sistema FIR y con sistema IIR, con esto se comprobó si se logró el objetivo de la herramienta.

3.1 Diseño de filtro digital, primera forma para diseño de filtros digitales

En esta prueba se demuestra el uso de la primera forma para diseñar filtros digitales (Colocación de polos y ceros por medio del puntero), donde se hace uso únicamente de la ventana principal de la herramienta, tomando como referencia un filtro diseñado previamente en la plataforma MATLAB®. Para la prueba se diseñó un filtro con las siguientes características:

- Respuesta del sistema: IIR.
- Tipo de filtro: Pasa bajas
- Frecuencia de muestreo: 10000 hz.
- Orden del filtro: 4° orden.

Para su diseño es necesario colocar por medio del puntero los polos y ceros, primeramente, se selecciona el botón de polos o ceros que se encuentra en la parte de arriba en la ventana principal como se muestra en la Figura 3.1:

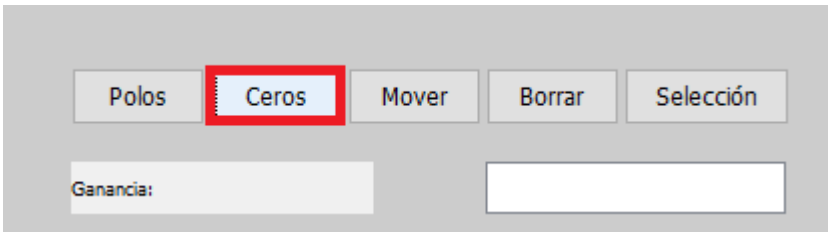

Figura 3.1 Botón de selección para crear ceros.

Una vez seleccionado se procede a colocar dentro de la ventana del plano z complejo, donde cada click dado genera un par de polos o ceros, esto es debido a que se crea de manera automática su conjugado, pudiendo apreciarse en la Figura 3.2:

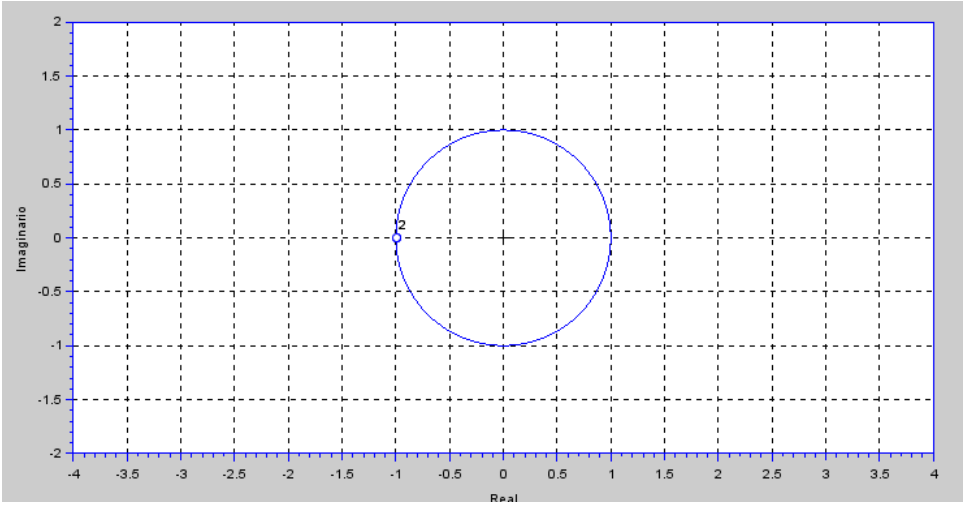

Figura 3.2 Colación de ceros en el plano z

En la Figura 3.2 se muestra un número "2" el cuál especifica que hay dos ceros en la misma posición, pero se esperaba tener cuatro debido a que fueron dos pares de ceros los creados que se esperaban estar todos en la misma posición, esto es debido a que su conjugado varía en las coordenadas Y en milésimas, por lo tanto, no se encuentran en el mismo lugar.

Si se desea tener un mayor control en las coordenadas se procede a seleccionar un par de ceros por medio del botón "Selección", el cual permite acceder a las coordenadas (X, Y) del polo o cero siempre y cuando esté desmarcada la casilla de Modo Polar, y así modificar sus parámetros de coordenadas como se muestra en la Figura 3.3:

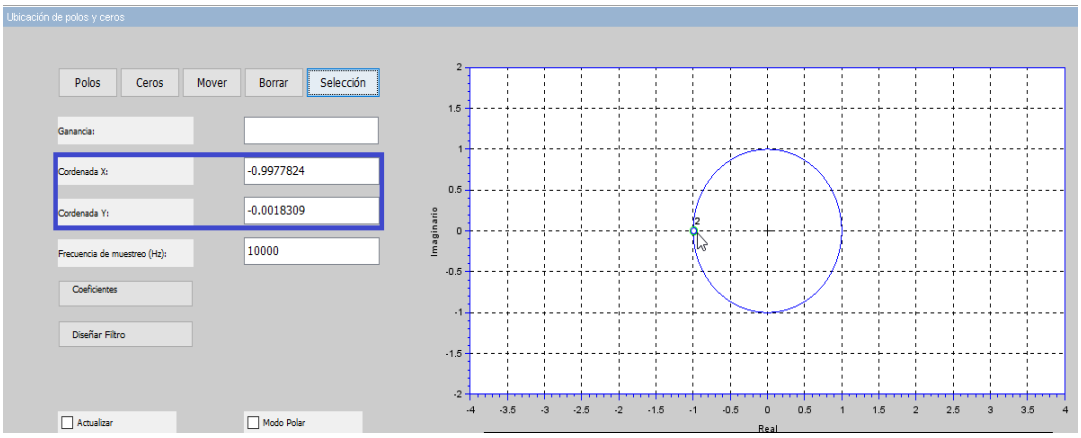

Figura 3.3 Selección de ceros

Una vez modificado los valores coordenadas X = -1, Y= 0, en ambos pares de ceros se obtiene los cuatros ceros en la misma posición mostrada en la Figura 3.4:

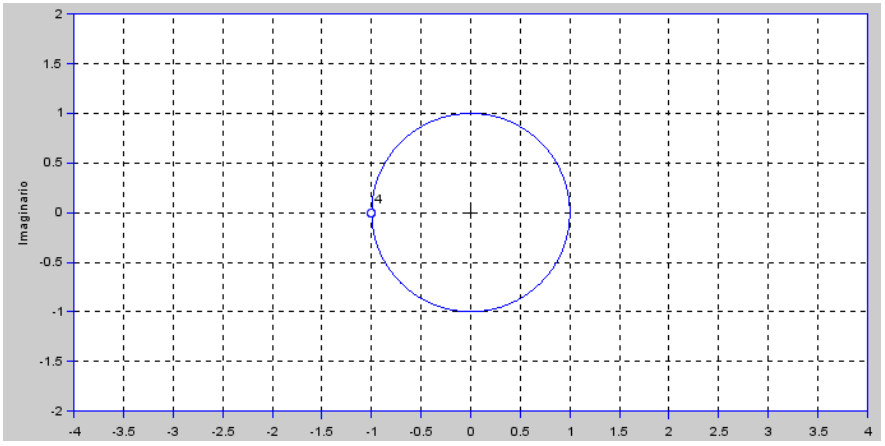

Figura 3.4 Cuatro ceros en la misma posición

Ya colocado los ceros se procedió a colocar los polos, esto se realiza seleccionando el botón de “Polos” y colocándolos en las posiciones correspondientes por medio del puntero mostrando el resultado en la Figura 3.5:

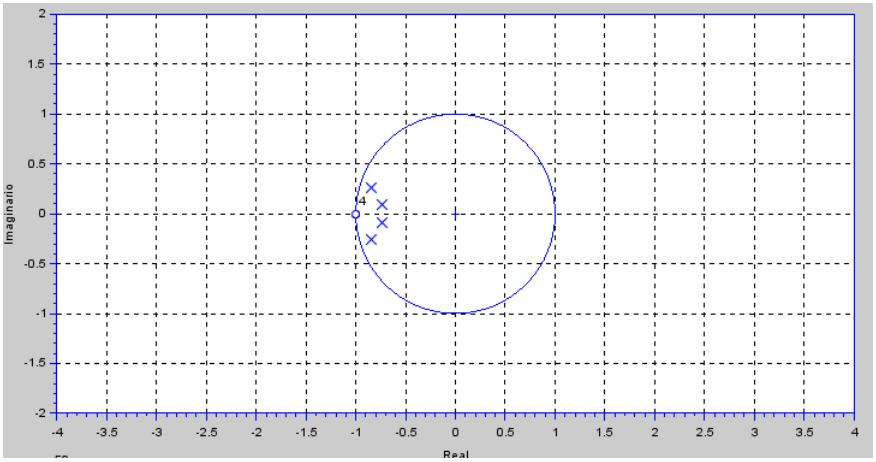

Figura 3.5 Colocación de los polos en el plano z

Finalmente, una vez colocado los polos y ceros se obtiene su respuesta en frecuencia como se obvserva en la Figura 3.6, demostrando que efectivamente se comporta como un filtro pasa altas con frecuencia en corte en 4500.

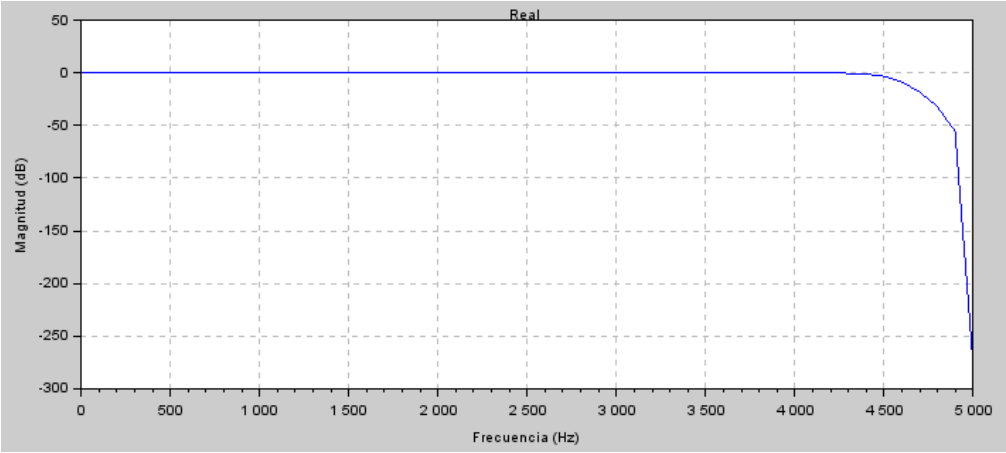

Figura 3.6 Respuesta en frecuencia del filtro diseñado

En base a 50 pruebas y el ejemplo se comprueba que el método de colocación de polos y ceros funcionó como se esperaba, y dando opción a que estos parámetros puedan modificados tanto por el puntero, como por el ingreso de coordenadas polares y rectangulares.

3.2 Diseño con sistema FIR, segunda forma para diseño de filtros

El objetivo de esta prueba es diseñar un filtro mediante un sistema FIR, con esto se espera que los polos únicamente estén situados en el origen, mientras que los ceros no tengan restricción en su posición, obteniendo su respuesta en frecuencia, así como sus coeficientes para poder implementar el filtro diseñado. Para esta prueba se diseñó un filtro con las siguientes características:

- Respuesta del sistema: FIR.
- Tipo de filtro: Pasa bajas.
- Frecuencia de muestreo: 10000 hz.
- Orden del filtro: 10° orden.
- Frecuencia de corte: 1700hz.
- Tipo de ventana: rectangular (default).

Mediante el cual se muestra la posición de sus polos y ceros en la Figura 3.7, donde se aprecia que sus polos se encuentran en el origen y sus ceros no tienen restricción, comportándose como se esperaba.

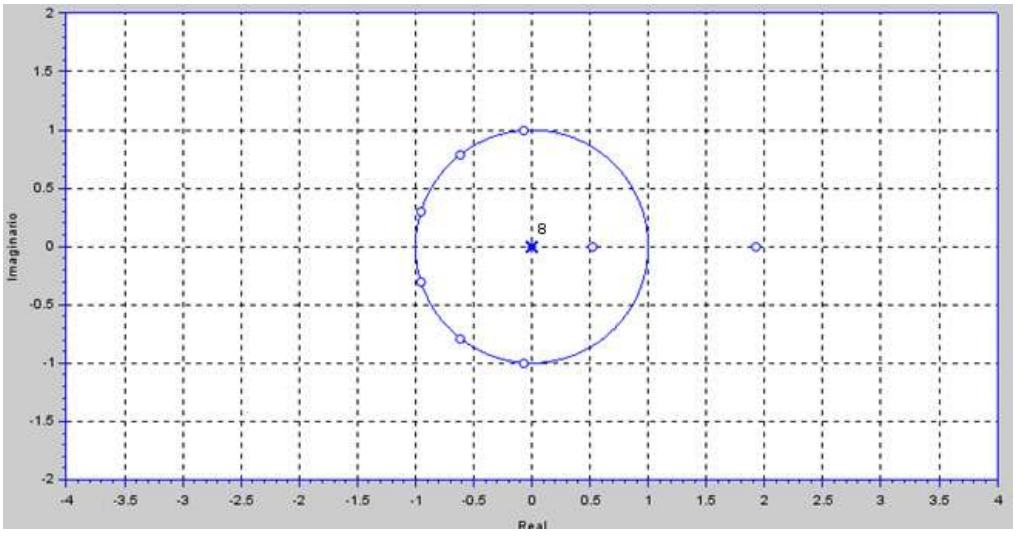

Figura 3.7 Ubicación de polos y ceros del filtro pasa bajas, sistema FIR

En base a la colocación de polos y ceros de la Figura 3.7 referente al filtro diseñado, se generó la gráfica de su respuesta en frecuencia mostrada en la Figura 3.8, donde se aprecia que efectivamente se trata de un filtro pasa bajas con frecuencia de corte en 1700hz.

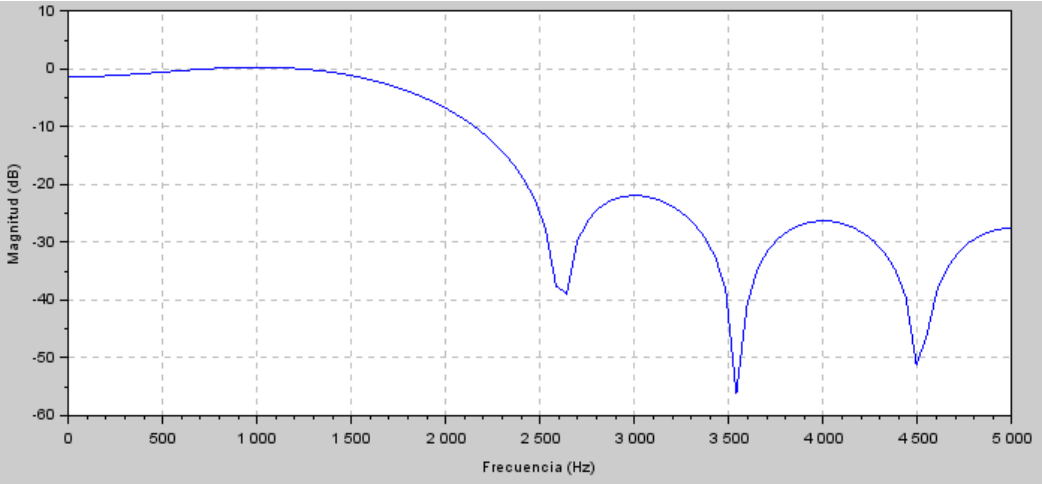

Figura 3.8 Respuesta en frecuencia del filtro pasa bajas, sistema FIR

Donde se muestra el factor de corrección en la Figura 3.9:

Figura 3.9 Ganancia del filtro pasa bajas, sistema FIR

De igual manera los coeficientes generados para implementar el filtro diseñado son los siguientes:

- Numerador: [1, 0.824045, -1.23606, -4, -5.285225, -4, -1.23606, 0.824045, 1]
- Denominador: 1

Validación:

Para validar el resultado obtenido se hizo uso del *toolbox* FDA tool de MATLAB®, donde se espera tener una respuesta en frecuencia muy cercana en valores al obtenido por la herramienta diseñada, así como la posición de polos y ceros en el mismo sitio.

La Figura 3.10 muestra la ubicación de los polos y ceros en el plano z complejo del FDA tool, donde se comprueba que efectivamente se obtuvo lo esperado en cuanto a posición de polos y ceros de la Figura 3.7 comparada con la Figura 3.10.

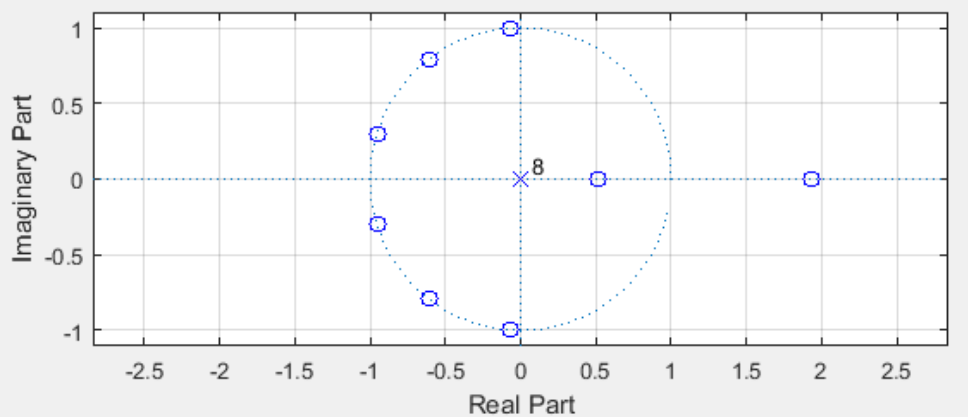

Figura 3.10 Ubicación de polos y ceros del filtro pasa bajas en FDA tool

Finalizado lo anterior se procedió a comparar su respuesta en frecuencia. La Figura 3.11 muestra repuesta en frecuencia hecha en FDA tool, donde se observó que su respuesta es muy parecida a la de la Figura 3.8, donde se apreció que la banda de paso y la frecuencia de corte coinciden, por lo cual se determina que es correcto el funcionamiento de la herramienta para diseñar filtros con sistema FIR.

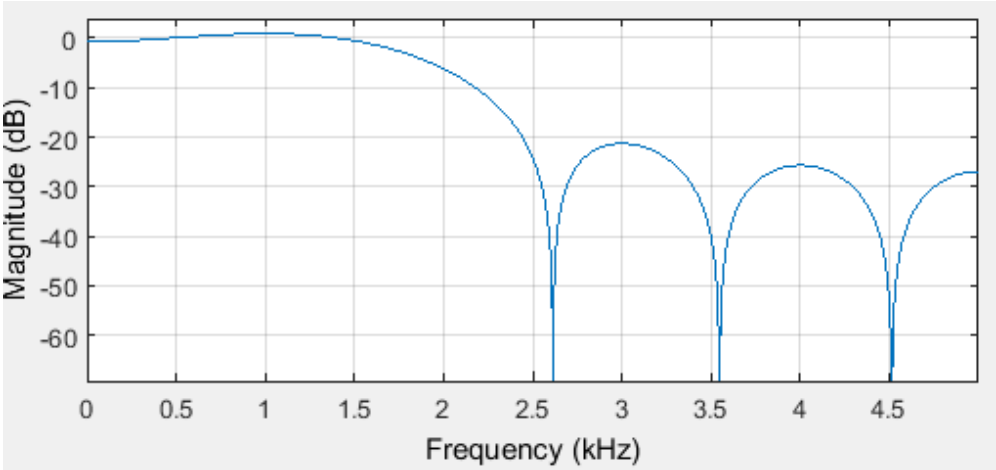

Figura 3.11 Respuesta en frecuencia de filtro pasa bajas en FDA tool

3.3 Diseño con sistema IIR, segunda forma para diseño de filtros digitales

Esta prueba tiene como objetivo diseñar un filtro mediante un sistema IIR, en el cual se espera que los polos estén dentro del círculo unitario para ser un sistema estable, no habiendo restricción para los ceros, una vez colocados los polos y ceros, obtener su respuesta en frecuencia, así como sus coeficientes para poder implementar el filtro diseñado. Para esta prueba se diseñó un filtro con las siguientes características:

- Respuesta del sistema: IIR.
- Tipo de filtro: Pasa bajas
- Frecuencia de muestreo: 15000 hz.
- Orden del filtro: 10° orden.
- Frecuencia de corte: 5000 hz.
- Butterworth (default).

Mediante el cual se muestra la posición de sus polos y ceros en la Figura 3.12, donde se observa que sus polos se encuentran en el origen y sus ceros no tienen restricción, y la posición de los ceros son muy parecidas.

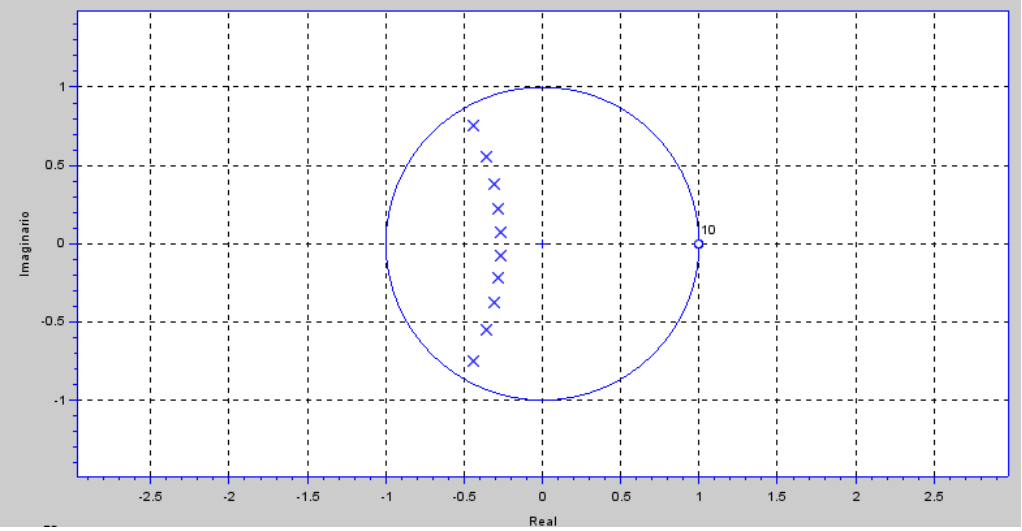

Figura 3.12 Ubicación de polos y ceros filtro pasa altas, sistema IIR

En base a la colocación de polos y ceros de la Figura 3.12 referente al filtro diseñado, se generó la gráfica de su respuesta en frecuencia mostrada en la Figura 3.13 donde se observa que efectivamente se trata de un filtro pasa altas con frecuencia de corte en 5000 hz.

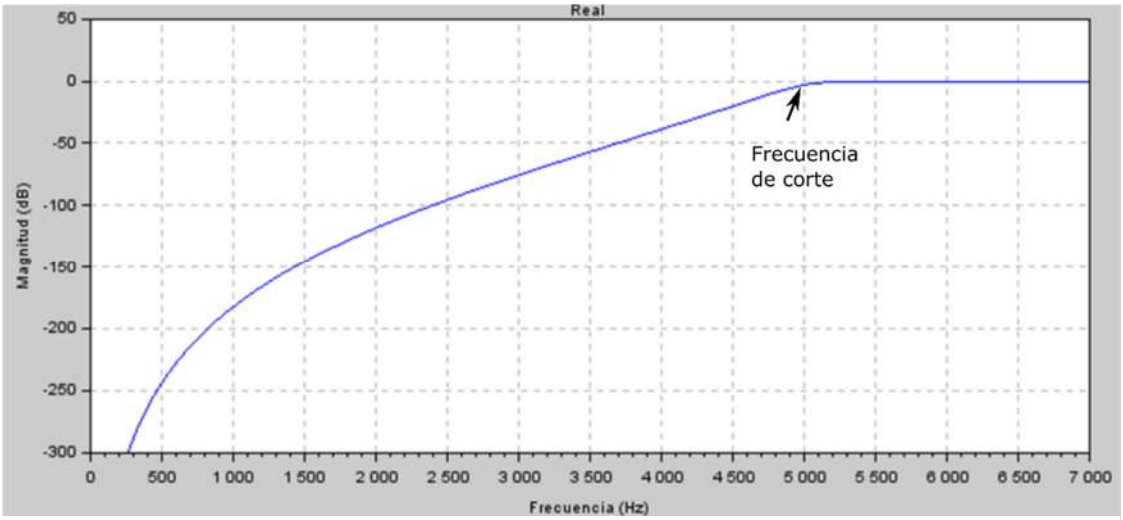

Figura 3.13 Respuesta en frecuencia de filtro pasa altas, sistema IIR

Donde se muestra su factor de correción en la Figura 3.14:

Figura 3.14 Ganancia del filtro pasa altas, sistema IIR

De igual manera sus coeficientes generados para poder ser implementar el filtro diseñado son los siguientes:

- Numerador: [0, -0.07568, 0.06236, 0.09354, -0.302, 0.4, -0.30273, 0.09354, 0.06236, -0.07568, 0]
- Denominador: [0.00014, 0.00266, 0.02249, 0.11571, 0.41444, 1.05454, 2.03872, 2.81852, 3.01948, 1.99240, 1]

Validación:

Para validarel resultado obtenido se realizó el mismo procedimiento que en sistema FIR, mediante el FDA tool de MATLAB®, donde se espera tener una respuesta en frecuencia muy cercana en valores al obtenido por la herramienta diseñada, así como la posición de polos y ceros en el mismo sitio.

La Figura 3.15 muestra la ubicación de los polos y ceros en el plano z complejo del FDA tool, donde se comprueba que efectivamente se obtuvo lo esperado en cuanto a la posición de polos y ceros de la Figura 3.12 comparada con la Figura 3.15.

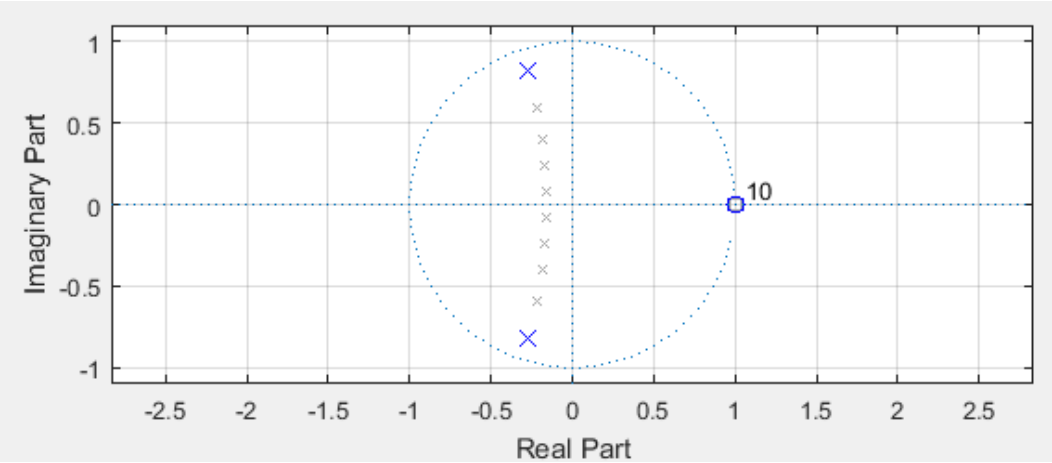

Figura 3.15 Ubicación de polos y ceros del filtro pasa altas, sistema IIR

Realizado lo anterior se procede a comparar su respuesta en frecuencia. La Figura 3.16 muestra la repuesta en frecuencia hecha en FDA tool, donde se observa que su respuesta son parecidas con ligera diferencia, esto debido a la diferencia de escalas en la gráfica y por la resolución que tienen ambas, por lo cual se determina que es correcto el funcionamiento de la herramienta para diseñar filtros con sistema IIR.

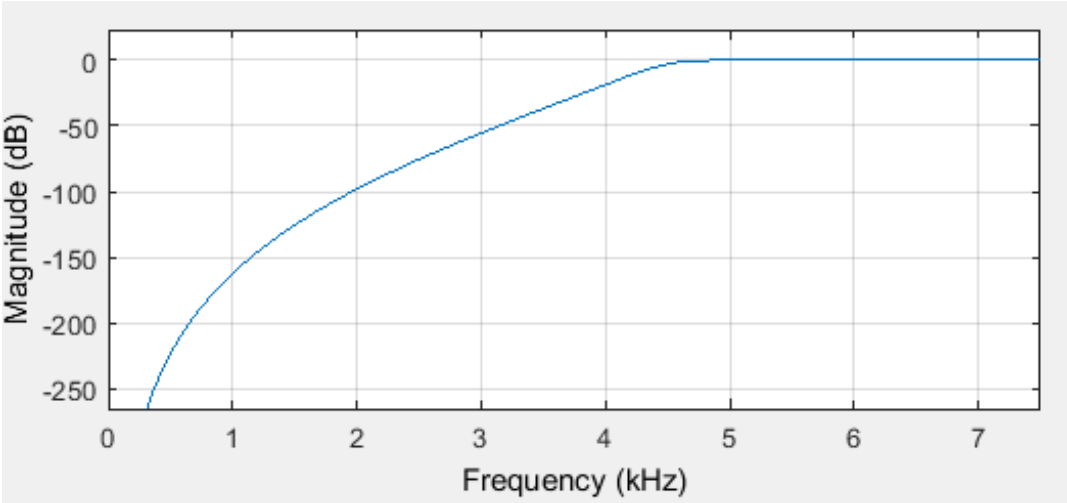

Figura 3.16 Respuesta en frecuencia del filtro pasa altas, sistema IIR

Al hacer las comparaciones en referencia entre la herramienta y FDA tool se observa que efectivamente el diseño de filtros digitales se está realizando de manera correcta.

En resumen, en este capítulo se observó que el funcionamiento de la herramienta elaborada para el diseño de filtros digitales en la plataforma SCILAB® resultó satisfactoria. Esta herramienta permite crear filtros digitales que pueden ser implementados de forma matemática (mediante los coeficientes obtenidos) en la plataforma SCILAB®.

A su vez, la herramienta permite modificar los parámetros del filtro por medio del reubicamiento de sus polos y ceros, o modificando las coordenadas de forma polar o rectangular de sus polos y ceros respectivos.

Capítulo 4
Conclusiones y trabajos futuros

4.1 Conclusiones

Como se demostró en el capítulo 3, esta herramienta cumple con el objetivo de diseñar filtros digitales, así como se logró manipular sus polos y ceros para observar su respuesta en frecuencia en base a la posición final en el plano z complejo.

En este trabajo desarrollaron dos formas para el diseño de filtros digitales, en el cual la segunda forma basada en el uso de una interfaz con la cual se ingresan los parámetros del filtro, se encuentra asociada a la primera forma que consiste en le ubicación de polos y ceros en el plano z complejo, logrando una mejora en tiempo requerido para el diseño de filtros digitales.

Así mismo, se logró ubicar los polos y ceros del filtro diseñado con el fin de tener una estructura didáctica, que permite al usuario observar los cambios de respuesta en frecuencia debido a la posición de los polos y ceros.

Una vez obtenido la respuesta en frecuencia, se permite al usuario tomar los coeficientes del filtro diseñado para implementar el filtro digital en una señal deseada.

4.2 Trabajos futuros

Es del interés del autor continuar con el desarrollo de la herramienta para diseño de filtros digitales, Añadiendo nuevas características y mejoras al actual.

Utilizar un método más eficiente para calcular los polinomios con el fin de que la herramienta sea menos pesada en cálculos matemáticos y se refleje en mayor velocidad de proceso.

Permitir al usuario implementar de manera automática el filtro diseñado en la interfaz *xcos* de la plataforma SCILAB®.

Añadir un marcador que se presente cuando el sistema es inestable, esto es, cuando los polos se encuentren fuera del círculo unitario, con el fin de ser más didáctico para el usuario.

Mejorar el aspecto de la interfaz gráfica principal separando con márgenes los módulos, así como imágenes que reemplacen los textos de los botones para colocar polos y ceros.

Implementar la herramienta como un *toolbox* nativo de la plataforma SCILAB® que pueda ejecutar la herramienta por medio de un comando.

Anexo A. Código fuente convertidor de coordenadas rectangulares a polares

```
function rect2pol (handles)   //Funciòn la cual copia lo que existe en ángulo y magnitud a la variable temporal
   global var;
   global handles;
   if handles.select_polar.value == 1 then
      if var.atrapado_c == %T then
         handles.value_magnitude.Value = sqrt ((var.temp.x)^2  +  (var.temp.y)^2 );//La raíz cuadrada de la sumatoria al cuadrado
         handles.value_magnitude.String = msprintf ( '%f' , handles.value_magnitude.Value );
         handles.value_magnitude.Value = handles.value_magnitude.String;
         var.v_mag = handles.value_magnitude.Value; //Asignar para poder hacer pol2rect

         //Para las opciones de ángulo se tiene que considerar el cuadrante, se pondrá en forma de negativos
         handles.value_angle.Value = atand ( var.temp.y / var.temp.x ); //el arctang de y/x

         //la seleccion de diferentes formas de cuadrantes
         //cuadrante 1
         if (var.temp.x > 0 & var.temp.y > 0) then
            handles.value_angle.String = msprintf ('%5.7f',handles.value_angle.Value);
            var.v_ang = handles.value_angle.Value; //valor de ángulo para pol2rect
         end
         if (var.temp.x>0 & var.temp.y < 0) then
            handles.value_angle.String = msprintf ('%5.7f',360 + (handles.value_angle.Value));
            var.v_ang = 360 + handles.value_angle.Value; //valor de ángulo para pol2rect
         end
         if (var.temp.x < 0 & var.temp.y > 0) then
            handles.value_angle.String = msprintf ('%5.7f',180 + (handles.value_angle.Value));
            var.v_ang = 180 + handles.value_angle.Value; //valor de ángulo para pol2rect
         end
         if (var.temp.x < 0 & var.temp.y < 0) then
            handles.value_angle.String = msprintf ('%5.7f',180 + (handles.value_angle.Value));
            var.v_ang = 180 + handles.value_angle.Value; //valor de ángulo para pol2rect
         end
      end

   else
      if var.atrapado_c == %T then
         handles.value_angle.Value = var.temp.y;
         handles.value_angle.String = msprintf ('%5.7f', handles.value_angle.Value);
         handles.value_angle.Value = handles.value_angle.String;

         handles.value_magnitude.Value = var.temp.x;
         handles.value_magnitude.String = msprintf ('%5.7f', handles.value_magnitude.Value);
         handles.value_magnitude.Value = handles.value_magnitude.String;
      end
   end
endfunction
```

Anexo B. Código fuente convertidor de coordenadas polares a rectangulares

```
function pol2rect (handles)
   global var;
   global handles;

   if handles.select_polar.value == 1 then
      if var.atrapado_c == %T then
         var.temp.x = var.v_mag * cosd (var.v_ang);
         var.temp.y = var.v_mag * sind (var.v_ang);
         if var.temp.conjugado < > - 1 then
            var.temp_conj.x = var.temp.x;
            var.temp_conj.y = - var.temp.y;
         end
      end
   end
endfunction
```

Anexo C. Código fuente de ventana secundaria para diseño de filtros

```
f1 = figure ('figure_position',[400,50],'figure_size',[656,482],'auto_resize','on','figure_name','GUI para diseño
de filtro');
f1.visible = 'off';
f1.closerequestfcn = "myclose"
//////////
delmenu (f1.figure_id, gettext ("&File"));
delmenu (f1.figure_id, gettext ("&Tools"));
delmenu (f1.figure_id, gettext ("&Edit"));
delmenu (f1.figure_id, gettext ("&?"));
toolbar (f1.figure_id, "off");
handles.dummy = 0;
handles.c_fir = uicontrol (f1,'unit','normalized','BackgroundColor',[ - 1, - 1, - 1]);
global var;
global handles;

//Variables a guardar para la respuesta al impulso (fir o iir)
var.f_m = 100; //frec muestreo
var.tip_filt = 'lp'; //tipo de filtro
var.ord_filt = 4;  //orden del filtro
var.frec_c_b = 5; //frec de corte baja
var.frec_c_a = 45; //frec de corte alta
var.t_ventana = 're'; //Tipo de ventana
var.r_i = 'f'; //Respuesta del sistema (f =  FIR , i = IIR)
var.ceros = 1;  //ceros creados en base al filtro Ya realizado
var.polos = 1;  //polos creados en base al filtro Ya realizado
var.y_anterior = '';
var.x_anterior = '';
```

```
function c_fir_callback (handles)
   global handles;
   global var;
   //Write your callback for  c_fir  here
   if handles.c_fir.Value == 1  then
      handles.c_iir.Value = 0;
      var.r_i='f';
   else
      handles.c_fir.Value = 1
      var.r_i = 'f';
   end
   ord_filt (handles)
endfunction

function c_iir_callback (handles)
   global handles;
   global var;
   //Write your callback for  c_iir  here
   if handles.c_iir.Value == 1  then
      handles.c_fir.Value = 0;
      var.r_i = 'i';
   else
      handles.c_iir.Value = 1;
       var.r_i = 'i';
   end
  ord_filt (handles)
endfunction

function lista_filtros_callback (handles)
   global handles;
   global var;
   //Write your callback for  lista_filtros  here
   //Visibilidad de pasa bajas
   if handles.lista_filtros.Value == 1 then
      handles.t_frec_alt.Visible = 'off';
      handles.frec_alta.Visible = 'off';
      handles.t_Frec_baja.Visible = 'on';
      handles.frec_baja.Visible = 'on';
      handles.a_filtro.String = 'lp';
      var.tip_filt = 'lp';
   end
   //Visibilidad para pasa altas
   if handles.lista_filtros.Value  == 2 then
      handles.t_frec_alt.Visible = 'on';
      handles.frec_alta.Visible = 'on';
      handles.t_Frec_baja.Visible = 'off';
      handles.frec_baja.Visible = 'off';
      handles.a_filtro.String = 'hp';
      var.tip_filt = 'hp';
      //forzar orden impar
      if 2 * int (var.ord_filt/2) == var.ord_filt then var.ord_filt = var.ord_filt + 1,end ///tomado del scrpit
wfir_gui.sce
      handles.orden_filtro.String = string (var.ord_filt);
   end
```

```
  //Visibilidad para pasa banda
    if handles.lista_filtros.Value == 3 then
       handles.t_frec_alt.Visible = 'on';
       handles.frec_alta.Visible = 'on';
       handles.t_Frec_baja.Visible = 'on';
       handles.frec_baja.Visible = 'on';
       handles.a_filtro.String = 'bp';
       var.tip_filt = 'bp';
    end
    //Visibilidad para rechaza banda
    if handles.lista_filtros.Value == 4 then
       handles.t_frec_alt.Visible = 'on';
       handles.frec_alta.Visible = 'on';
       handles.t_Frec_baja.Visible = 'on';
       handles.frec_baja.Visible = 'on';
       handles.a_filtro.String = 'sp';
       var.tip_filt = 'sb';
       //impar orden
       if 2 * int (var.ord_filt/2) == var.ord_filt then var.ord_filt = var.ord_filt + 1,end ///tomado del scrpit
wfir_gui.sce
       handles.orden_filtro.String = string (var.ord_filt);
    end
endfunction

function ok_callback (handles)
    global var;
    global handles;
    global f1;
    var.cantidad_marcadores = 0;


    limpiar_imag ()
    handles.value_Frec_m.String = msprintf ('%5.3f',handles.value_Frec_m.Value); //graficar el valor de frec
de muestreo en la ventana principal
    edit_frec_corte ()//convertimos en porcentaje la frec de corte

    if var.r_i == 'f' then
       //pasa bajas
       wft = wfir (string (var.tip_filt),var.ord_filt,var.f_c,string (var.t_ventana),[ -  1  -  1]);
       var.wft = wft;
    else
       wft = iir (var.ord_filt,string (var.tip_filt),'butt',var.f_c,[0 0]);
       var.wft = wft;
    end
    raíces () //Generar las raíces en base al sistema creado
    dis_filt (handles) //Colocar los ceros y polos
    var.gan  = %T; //para colocar la ganancia necesaria
    polinomio ()
    f1.visible = 'off';

endfunction

function refr_frec (handles)
    global var;
    global handles;
    var.f_m = eval (handles.f_muestreo.String);
```

```
    if var.f_m <=  0 then
        handles.f_muestreo.String = '1';
        var.f_m = 1;
    end
    handles.value_Frec_m.Value =  var.f_m;
    act_para (handles)
endfunction

function act_para (handles)
    global var
    global handles;
    if var.frec_c_b  > var.f_m * 0.45  then
        var.frec_c_b = var.f_m * 0.05;
        handles.frec_baja.String = string (var.frec_c_b);
    end

    if var.frec_c_a>=  var.f_m * 0.49 then
        var.frec_c_a = var.f_m * 0.45;
        handles.frec_alta.String = string (var.frec_c_a);
    end
endfunction

function ed_frec_b (handles)
    global var
    global handles;
    var.frec_c_b = eval (handles.frec_baja.String);
    //Debug;
    if var.frec_c_b <  0  then
        var.frec_c_b = var.f_m * 0.05;
        handles.frec_baja.String = string (var.frec_c_b);
    end
    if var.frec_c_b  > (var.f_m * 0.49)  then
        var.frec_c_b = var.f_m * 0.4;
        handles.frec_baja.String = string (var.frec_c_b);
    end
    //chequeo
    no_mayor (handles)
endfunction

function ed_frec_a (handles)
    global var;
    global handles;
    var.frec_c_a = eval (handles.frec_alta.String);
    //Debug;
    if var.frec_c_a <  0  then
        var.frec_c_a = var.f_m * 0.45;
        handles.frec_alta.String = string (var.frec_c_a);
    end
    if var.frec_c_a > (var.f_m * 0.5)  then

        var.frec_c_a = var.f_m * 0.5;
        handles.frec_alta.String = string (var.frec_c_a);
    end
    //checar que no es mayor la frec baja
    no_mayor (handles)
endfunction
```

```
function no_mayor (handles)
   global var
   global handles;
   if var.frec_c_b >=  var.frec_c_a then
      var.frec_c_a = var.f_m * 0.49;
      handles.frec_alta.String = string (var.frec_c_a);
      //Anuncio
   end
endfunction


function ord_filt (handles)
   global var;
   global handles;
   var.ord_filt = eval (handles.orden_filtro.String);
   if var.ord_filt < =  1 then
      handles.orden_filtro.String = '2';
      var.ord_filt = 2;
   end
   //impar para hb
   if handles.c_fir.Value  == 1 then  //debug para impar en respuesta FIR
      if handles.lista_filtros.Value == 2 then
         if 2 * int (var.ord_filt/2) == var.ord_filt then var.ord_filt = var.ord_filt + 1,end ///tomado del script
                                                                                              wfir_gui.sce
         handles.orden_filtro.String = string (var.ord_filt);
      end
      //impar para sb
      if handles.lista_filtros.Value  == 4 then
         if 2 * int (var.ord_filt/2) == var.ord_filt then var.ord_filt = var.ord_filt + 1,end ////tomado del scrpit
                                                                                              wfir_gui.sce
         handles.orden_filtro.String = string (var.ord_filt);
      end
   end
endfunction

function edit_frec_corte ()
   global var;
   global handles;
   fm = var.f_m
   var.lf = (var.frec_c_b)/fm;
   var.hf = (var.frec_c_a)/fm;

   //Diferentes frecuencia de corte dependiendo de qué tipo de filtro que sea
   if var.tip_filt == 'lp' then
      var.f_c = [var.lf,0];
   elseif var.tip_filt == 'hp' then
      var.f_c = [var.hf,0];
   else
      var.f_c = [var.lf,var.hf]
   end

endfunction
```

```
function raíces ()
   global var
   global handles
   if var.r_i == 'f' then
      var.ceros = roots (var.wft);
      var.polos = 1;
   else
      var.ceros = roots (numer (var.wft));
      var.polos = roots (denom (var.wft));
   end
endfunction


function coloca (X, Y)                    //coloca los ceros y polos en el círculo unitario
   global var
   global handles

   sca (handles.imag_circUnit);       //Asignar el manejador al axes del círculo unitario
   plot (X,Y,var.Tipo)                //Dibuja con las coordenadas X,Y una figura de identidad "tipo", donde
puede ser un 'o' o 'x'.


   if  -  Y == var.y_anterior & X == var.x_anterior  then      //identificar que son polos o ceros ne el mismo
lugar
      //primero borrar el s_raiz anterior para poderlo poner con conjugado
      [cuadrante, indice] = busca (var.x_anterior,var.y_anterior);
      var.c1 = cuadrante;          //Asignar las variables que seran usadas en la modificación por texto
      var.in_1 = indice;

      //Ahora  crear sus valores
         s = 1
         while (s < > 0)
         if var.cuadrante (var.c1)(var.in_1).conjugado < >  -  1 then         //Esto es para encontrar el correcto
                                                                              indice para poner su conjugado
            var.in_1 = var.in_1 + 1
         else
            s = 0;
        end
        end
      hc_raiz = get ("current_entity");  //Capturar el manjeador de la figura creada (de su conjugaado)
      var.sc_raiz = struct ('figura',hc_raiz.children,'tipo',var.Tipo,'entidad',(%z -  (X + (%i * Y))),
'x',X,'y',Y,'conjugado',var.s_raiz);
      var.cuadrante (var.c1)(var.in_1).conjugado  =  var.sc_raiz;//le asigna una referencia del conjugado a la
                                                                  raíz primeramente creada
      var.sc_raiz.conjugado = var.cuadrante (var.c1)(var.in_1);
      var.cuadrante (var.c1)(var.in_1).conjugado  =  var.sc_raiz;//le asigna una referencia del conjugado a la
                                                                  raíz primeramente creada
      enlista (var.sc_raiz);
      //limpear variables
      var.s_raiz = ''; //Borrar valores de estructuras
      var.sc_raiz = '';//Borrar valores de estructura
      var.y_anterior = '';
      var.x_anterior = '';
      var.c1 = '';
      var.in_1 = '';
   else
```

```
      h_raiz = get ("current_entity");      //Capturar el manejador de la figura creada con el plot, y se
                                                guarda en una variable
      var.s_raiz = struct ('figura',h_raiz.children,'tipo',var.Tipo,'entidad',(%z -  (X + (%i * Y))),
                           'x',X,'y',Y,'conjugado', -  1);// Crear una estructura de la raíz creada, con los dato
      var.y_anterior = Y;
      var.x_anterior = X;
      enlista (var.s_raiz);
      var.s_raiz = ';//Borrar valores de estructura
   end
endfunction


function dis_filt (handles)
   global var
   global handles
   ////////////////////sistema FIR//////////////////////
   if var.r_i == 'f' then
      //ceros
      var.Tipo = 'o';
      r = size (var.ceros);
      for i = 1:r (1) //Seleccionar solo la parte de columnas
         var.X = real (var.ceros (i));
         var.Y = imag (var.ceros (i));
         coloca (var.X,var.Y);
         i = i + 1;
      end
      //polos
      var.Tipo = 'x'
      for j = 1:r (1) //Seleccionar solo la parte de columnas
         var.X = 0;
         var.Y = 0;
         coloca (var.X,var.Y);
         j = j + 1;
      end

   else
      /////////////// sistema IIR//////////////////////
      //ceros

      var.Tipo = 'o';
      r = size (var.ceros);
      if string (var.tip_filt) == 'lp' then
         for i = 1:r (1) //Seleccionar solo la parte de columnas
            var.X = 1;
            var.Y = 0;
            coloca (var.X,var.Y);
            i = i + 1;
         end
      elseif string (var.tip_filt) == 'hp' then
         for i = 1:r (1) //Seleccionar solo la parte de columnas
            var.X =  -  1;
            var.Y = 0;
            coloca (var.X,var.Y);
            i = i + 1;
         end
```

```
    elseif  string (var.tip_filt) == 'bp' then
        L = r (1)/2;
        for i = 1:L //Seleccionar solo la parte de columnas
            var.X =  -  1;
            var.Y = 0;
            coloca (var.X,var.Y);
            i = i + 1;
        end

        for k = 1:L //Seleccionar solo la parte de columnas
            var.X = 1;
            var.Y = 0;
            coloca (var.X,var.Y);
            k = k + 1;
        end
     elseif    string (var.tip_filt) == 'sb' then
    //                      var.Tipo = 'o';
            r = size (var.ceros);
            for i = 1:r (1) //Seleccionar solo la parte de columnas
                var.X = real (var.ceros (i));
                var.Y = imag (var.ceros (i));
                coloca (var.X,var.Y);
                i = i + 1;
            end
    end

    //polos
    var.Tipo = 'x';
    for j = 1:r (1) //Seleccionar solo la parte de columnas
        var.X = real (var.polos (j));
        var.Y = imag (var.polos (j));
        coloca (var.X,var.Y);
    end
  end
endfunction


function limpiar_imag (handles)
global var
global handles
// Borrar todos los marcadores creados ("x" o "y") y vuelve a dibujar el círculo unitario
delete (handles.imag_circUnit.children) ;
sca (handles.imag_circUnit);
xset ("color",2);
xarc ( -  1,1,2,2,0,23040) ;  //generar un círculo (23040 = 360 * 64)
plot (0,0,' + ');//cruz central

//Borrar todas las estructuras antes almacenadas en las listas
var.cuadrante = list ();
var.cuadrante (1) = list ();
var.cuadrante (2) = list ();
var.cuadrante (3) = list ();
var.cuadrante (4) = list ();

endfunction
```

```
function myclose ()
 global f1
 f1.visible = 'off';
 endfunction
```

Anexo D. Código fuente de ventana principal para diseño de filtros

```
// This GUI file is generated by guibuilder version 3.0
//\\\\\\Se limpia todo el registro para evitar problemas\\\\\\\\\\\\\
close
clear
clearglobal
clc

exec (SCI + '/proyecto/macros/filtro_gui.sce', -  1);
exec (SCI + '/proyecto/macros/rect2pol.sci', -  1);
exec (SCI + '/proyecto/macros/pol2rect.sci', -  1);

f = figure ('figure_position',[ -  8, -
8],'figure_size',[1260,800],'auto_resize','on','background',[33],'figure_name','Ubicación de polos y ceros');  //se
crea la figura (ventana)
f.closerequestfcn = "Cerrar" //provoca que al momento de hacer click en la X del la ventana, primero realice
un proceso antes de cerrar la ventana

delmenu (f.figure_id,gettext ('File'))
delmenu (f.figure_id,gettext ('?'))
delmenu (f.figure_id,gettext ('Tools'))
toolbar (f.figure_id,'off')

handles.dummy  =  0;
// Crear los manejadores de botones, axes, etc (Creado por GUI builder)

handles.imag_respFreq =  newaxes ();handles.imag_respFreq.margins  =  [ 0 0 0
0];handles.imag_respFreq.axes_bounds = [0.432194,0.57346,0.5066513,0.361481];
handles.imag_circUnit =  newaxes ();handles.imag_circUnit.margins = [ 0 0 0
0];handles.imag_circUnit.axes_bounds = [0.3510929,0.0575285,0.6652459,0.4619011];

// Creación de una variable global  la cual contendrá las demás variables que se manejarán dentro del
programa
global var;
global handles;
global f1;
var.cuadrante = list ();
var.cuadrante (1) = list ();
var.cuadrante (2) = list ();
var.cuadrante (3) = list ();
var.cuadrante (4) = list ();
var.list_numer = list (); //lista de numeración de polos y ceros en la misma coordenada
var.Tipo = '';
var.fig = gcf ();
var.temp = '';   //Es para guardar una raíz (movimiento)
```

```
var.temp_conj = ''; // Viarable para el temporal conjugado
var.atrapado = %F;  //Bandera para el movimiento
var.atrapado_c = %F; //es el atrapado para modificación visual
var.polipolos = 1; //Variable
var.policeros = 1;
var.co_polos = 1;//Coeficientes de los polos
var.co_ceros = 1;//Coeficientes de los ceros
var.temp_mod = ''; //temporal secundario, para poder hacer la modificación visual sin recurrir al temp
normal
var.temp_conj_mod = '';
var.c1 = '';
var.c2 = '';
var.in_1 = '';
var.in_2 = '';
var.mag = '';
var.ang = '';
var.v_mag = ''; //se guarda el valor de magnitud para poder hacer pol2rect
var.v_ang = '';
var.f_m = 100;//Frec de muestreo y la freq para intervalo
var.gan = %f;
var.cantidad_marcadores = 0; //cantidad de marcadores de números


//Creación de la gráfica por default del círculo unitario
plot2d (0,0, -  1,"031"," ",[ -  4, -  2,4,2])
xset ("color",2)
xarc ( -  1,1,2,2,0,23040)   //generar un círculo (23040 = 360 * 64)
xgrid
handles.imag_circUnit.x_label.text =("Real")
handles.imag_circUnit.y_label.text =("Imaginario")

//Asignar el manejador al actual (Axes) creado
sca (handles.imag_respFreq);
//gainplot ();    //graficar (por defaul crea unos ejemplos gráficos)
handles.imag_respFreq.x_label.text = _("Frecuencia (Hz)")
handles.imag_respFreq.y_label.text = _("Magnitud (dB)")
handles.imag_respFreq.grid = ones (1,2) * color ("gray")


   plot ([0 var.f_m/2],[1 1])  //gráfica por default (linea en cero)

//Se asigna ahora de nuevo el manejador al círculo unitario
sca (handles.imag_circUnit);

funcprot (0)   //evitar que nos salgan errores con respecto a los manejadores

function btn_select_callback (handles)
   global var;
   global handles;
   var.fig.event_handler_enable = 'off';   // Apagar por seguridad cualquier manejador antes existente
   var.fig.event_handler  =  'Select_p_z' ;   // Mandar llamar la función Genraiz
   var.fig.event_handler_enable  =  "on";  // Activar el manejador (con esto provocar que sea el manejador
correcto
   //Select_p_z ();
endfunction
```

```
function btn_ceros_callback (handles)
   global var;
   global handles;                    // Le decimos a la función que nuestra variable es la global, y no una Local
   var.fig.event_handler_enable = "off"; //Apagar por seguridad cualquier manejador antes existente
   var.fig.event_handler = 'Genraiz' ;   // Mandar llamar la función Genraiz
   var.fig.event_handler_enable = "on";  // Activar el manejador (con esto provocar que sea el manejador
correcto)
   var.Tipo = 'o';                    // Asignar el tipo (todo esta dentro de una variable general "var")
endfunction

function btn_polos_callback (handles)
   global var;                        // Le decimos a la función que nuestra variable es la global, y no una Local
   global handles;
   var.Tipo = 'x';                    // Asignar el tipo (todo esta dentro de una variable general "var"), Polo
   var.fig.event_handler_enable  =  "off";   // Apagar por seguridad cualquier manejador antes existente
   var.fig.event_handler = 'Genraiz' ;     // Mandar llamar la función "Genraiz"
   var.fig.event_handler_enable = "on";    // Asignar el tipo (todo esta dentro de una variable general "var")
endfunction

function btn_clear_callback (win, x, y, ibut)
   global var;
   global handles;                    // Le decimos a la función que nuestra variable es la global, y no una Local
   var.fig.event_handler_enable = 'off';    // Apagar por seguridad cualquier manejador antes existente
   var.fig.event_handler = 'borrar';        // Mandar llamar la función "Borrar"
   var.fig.event_handler_enable = 'on';     // Activar el manejador (con esto provocar que sea el manejador
correcto)

endfunction




function btn_move_callback (handles)
   global var;                        // Le decimos a la función que nuestra variable es la global, y no una Local
   global handles;
   var.fig.event_handler_enable = 'off';   // Apagar por seguridad cualquier manejador antes existente
   var.fig.event_handler = 'moverx';
   var.fig.event_handler_enable = 'on';     // Activar el manejador (con esto provocar que sea el manejador
correcto)
endfunction


function select_polar_callback (handles)
   global handles
   if handles.select_polar.Value == 1 then
      handles.txt_magnitud.String = 'Magnitud:';
      handles.txt_angle.String = 'Ángulo:';
      handles.value_magnitude.Callback = 'up_magnitud ()';//Equivalente a la
      handles.value_angle.Callback =  'up_ángulo ()';
      rect2pol ();

   else
      handles.txt_magnitud.String = 'Cordenada X:';
      handles.txt_angle.String = 'Cordenada Y:';
      handles.value_magnitude.Callback = 'up_x ()';//Equivalente a la coordenada X
```

```
    handles.value_angle.Callback =  'up_y ()';
    rect2pol ();
  end
endfunction


function Genraiz (win, x, y, ibut)
  global var
  global handles
  if ibut ==  -  1000 then return,end     //Concluimos el evento "Genraiz" si se cierra le ventana   (figura)
  [X,Y] = xchange (x,y,'i2f')          //Nos regresa las coordenadas x,y de pixeles a escala del AXES que se
está manejando, y X Y en coord de pixeles
  xinfo (msprintf ('Event code %d at mouse position is (%f,%f)',ibut,X,Y)) //Pone dentro de la Ventana, en
la posicion de abajo, los datos de que evento de click se dió y las coord x,y

  if ibut == 3 then                   //ibut 3, representa el click izquierdo (un click, sin dejar apretado el botón)
    sca (handles.imag_circUnit);       //Asignar el manejador al axes del círculo unitario
    plot (X,Y,var.Tipo)                //Dibuja con las coordenadas X,Y asigandas por el ratón, una figura de
identidad "tipo", donde puede ser un 'o' o 'x'.
    h_raiz = get ("current_entity");     //Capturar el manejador de la figura creada con el plot, y se guarda
en una variable
    s_raiz = struct ('figura',h_raiz.children,'tipo',var.Tipo,'entidad',(%z -  (X + (%i * Y))),
'x',X,'y',Y,'conjugado', -  1);         // Crear una estructura de la raíz creada, con los dato
s necesarios
    if handles.select_conjugate.value == 1 then      //Esto ya no es neceario pero sirve de debug
      //Crear su conjugado si está marcada la opción de conjugado
      plot (X, -  Y,var.Tipo)            //Dibujar el conjugado del polo o cero
      hc_raiz = get ("current_entity");  //Capturar el manjeador de la figura creada (de su conjugaado)
      //Seguro contra polos en el mismo lugar

      sc_raiz = struct ('figura',hc_raiz.children,'tipo',var.Tipo,'entidad',(%z -  (X + ( -  %i * Y))), 'x',X,'y', -
Y,'conjugado', s_raiz); //crear la estructura de la raíz creada
      s_raiz.conjugado = sc_raiz;     //le asigna una referencia del conjugado a la raíz primeramente creada
(con esto ambas estarán entre sí referenciadas).
      enlista (sc_raiz);
    end
    enlista (s_raiz);                 //mandar enlistar la raíz y su conjugado si está en existencia con la función
"enlista".

    //Generar los polinomios
    polinomio (); // La multiplicacion  de los polinomios que a su ve grafica RESPUESTA EN
FRECUENCIA
  end
endfunction


function cuadrante = UbicaCuadrante (x, y)

  if x >= 0 & y >= 0 then      // Esta parte asigna el primer cuadrante .
    cuadrante = 1;
  elseif x  <  0 & y >= 0 then    //Nuestra variable se tome en segundo cuadrante.
    cuadrante = 2;       //Agregar al segundo cuadrante lo contenido en "cosa".
  elseif x  <  0 & y  <  0 then    //Nuestra variable se tome el tercer cuadrante.
    cuadrante = 3;       //Agregar al tercer cuadrante lo contenido en "cosa".
  elseif x>0 & y <  0 then
    cuadrante = 4;      //Agregar al cuarto cuadrante lo coneido en "cosa".
```

```
   end
endfunction


function enlista (cosa)
   global var;                        // Dterminar a la función que nuestra variable es la global, y no una Local.
   i = UbicaCuadrante (cosa.x,cosa.y)
   var.cuadrante (i)($ + 1) = cosa;
endfunction


// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * Movimiento * * * * * * * * * * * * * * * * * * *
function mod_raíz_temp (x, y)
   global var;
   var.temp.figura.data = [x y];
   var.temp.x = x;
   var.temp.y = y;

endfunction


function mod_raíz_temp_conj (x, y)
   global var;
   var.temp_conj.figura.data = [x y];
   var.temp_conj.x = x;
   var.temp_conj.y = y;

endfunction


function moverx (win, x, y, ibut)
   global var;
   global handles
   indice  =  -  1; //indicador de que no se han encontrado ningun polo o cero al dar click

   if ibut ==  -  1000 then return,end      //Concluimos el evento "Genraiz" si se cierra le ventana   (figura)
   [X,Y] = xchange (x,y,'i2f')           //Nos regresa las coordenadas x,y de pixeles a escala del AXES que se
                                                 está manejando, y X Y en coord de pixeles
   xinfo (msprintf ('Event code %d at mouse position is (%f,%f)',ibut,X,Y)) //Pone dentro de la Ventana, en
            la posicion de abajo, los datos de que evento de click se dió y las coord x,y

   if ibut~ = 0  then  //(ibut cero es dejar el botón apretado)
      return;
   end
   // busqueda

   if ibut == 0 then
      var.atrapado_c = %F;
      Mod_visual ();

      //ahora borrar los temporales
      var.temp = '';
      var.temp_conj = '';                        //Para que no tome el conjugado de otro ya existente si
seleccionar solo un polo o cero

      [cuadrante, indice] = busca (X,Y);
```

```
        var.c1 = cuadrante;
        var.in_1 = indice;
        if indice~ =  - 1 then
            var.temp = var.cuadrante (cuadrante)(indice);   //Crear un temporal  struct

            if var.temp.conjugado < > - 1 then
                [cuadrante2,indice2] = busca (var.temp.x, - var.temp.y);                    //busca el conjugado
                var.c2 = cuadrante2;
                var.in_2 = indice2;
                if var.in_2 < > - 1 then //Debug
                    if var.in_1 == var.in_2 & var.c1 == var.c2 then        //esto es para evitar si existen muchos polos
o ceros en ese lugar, poder tener el siguiente de la lista y no tome el mismo
                        var.in_2 = var.in_2 + 1;
                        var.temp_conj = var.cuadrante (var.c2)(var.in_2);     //Crear la variable temporal conjugada
que es con la cual ge

                else

                    var.temp_conj = var.cuadrante (var.c2)(var.in_2);

                end
            end
        end
    end
    if indice  == - 1 then   //Si no encontro nada en donde dimos click , simplemente no hace nada
        var.c1 = '';         //Asignar las variables que seran usadas en la modificación por texto
        var.c2 = '';
        var.in_1 = '';
        var.in_2 = '';
        handles.value_magnitude.String = '';
        handles.value_angle.String = '';
        return;
    end
    //Borrar temporal mientra se mueve la raíz con su temporal
    rep = [X,Y, - 1];                          //Obliga a entrar en el while
    var.atrapado = %T;                         //Pon en true la bandera "atrapado"
    var.atrapado_c = %T;                       //Pon el atrapado de color
    if var.temp.conjugado ~ =  - 1 then //preguntar que existe conjugado, para así borrarlo //////////
        if indice2 < > - 1 then
            var.cuadrante (var.c2)(var.in_2) = null ();
        end
    end

    var.cuadrante (cuadrante)(indice)  =  null ();   //Borrar el actual en lista
    Mod_visual ()
end

while rep (3) == - 1 do  //mientras no suelte el click
    rep = xgetmouse ([%T %T]);                    // el doble true es para poder obtener ibut - 5
    mod_raíz_temp (rep (1),rep (2));              //Modifica la Raíz
    if handles.select_conjugate.value == 1 then   //esto si (pregunta) tiene su conjugado
        if var.temp.conjugado  ~= - 1 then
            if var.temp_conj  <  > '' then          //proteccion contra no existente conjugado
                mod_raíz_temp_conj (rep (1), - rep (2));
            end
        end
```

```
    end
    if handles.select_autUpd.value == 1 then
      auto_upd ()
    end

  end

  if (rep (3) ~= - 1) &(var.atrapado) then
    rect2pol ();      //Se encarga tambien de mostar en la pantalla el valor del polo o cero que recién se
movió

    if var.temp.conjugado < > - 1 then   //si tiene su conjugado
      if var.temp_conj < > '' then
        var.temp.conjugado = var.temp_conj // Asignar de nuevo la propiedad conjugado del
temporal_conj
        var.temp_conj.conjugado = var.temp;     //Asignar de nuevo propiedad conjugado del temp
        enlista (var.temp_conj);
        enlista (var.temp);
      end
    else           //no tiene conjugado pues solo enlista
      enlista (var.temp);
    end

   //volver a buscar el lugar en la lista del cero o polos que se quedó seleccionado para poderse modificar
en valores coordenadas
    var.atrapado = %F;
    [cuadrante, indice] = busca (var.temp.x,var.temp.y);
    var.c1 = cuadrante;
    var.in_1 = indice;

     if var.temp.conjugado < > - 1 then
        [cuadrante2,indice2] = busca (var.temp.x, - var.temp.y);                  //busca el conjugado
        var.c2 = cuadrante2;
        var.in_2 = indice2;
     end
    polinomio ()
  end
endfunction

function [nc, indice] = busca (X, Y)
  global var
  global handles
  xlim_s = X + 0.09;  //Crear límite superior de X
  xlim_i = X - 0.09;  //Crear límie inferior de X
  indice = - 1;
  nc = UbicaCuadrante (X,Y);
  for i = 1 : size (var.cuadrante (nc))
    //  printf ('%d',i)
    if var.cuadrante (nc)(i).x < xlim_s & var.cuadrante (nc)(i).x>xlim_i then
      ylim_s = Y + 0.09;                  //Crear límite superior de Y
      ylim_i = Y - 0.09;                  //Crear límite inferior de Y
      if var.cuadrante (nc)(i).y>ylim_i & var.cuadrante (nc)(i).y < ylim_s then //Checar que esté
                                                                                dentro de los límites de y
        indice = i;
        return;
      end
```

```
        end
        i = i + 1;
    end
endfunction


function polinomio ()
    global var
    global handles
    //Buscar todas las X y Y de cada polo o cero creado , esto dentro de la lista de lista
    //Iniciar al búsqueda para las multiplicacion de polinomios
    z = %z;             //definimos Z como transformada Z
    var.polipolos = 1;
    var.policeros = 1;
    for cuadra = 1:4
        for i = 1:size (var.cuadrante (cuadra))
            if var.cuadrante (cuadra)(i).tipo == 'x' then
                if var.cuadrante (cuadra)(i).x ~ =  0 & var.cuadrante (cuadra)(i).y ~= 0 then
                    var.polipolos = var.polipolos * (z -  (var.cuadrante (cuadra)(i).x + (%i * var.cuadrante
(cuadra)(i).y)));
                end
            else
                var.policeros = var.policeros * (z -  (var.cuadrante (cuadra)(i).x + (%i * var.cuadrante
(cuadra)(i).y)));
            end
        end
    end
    //Tomar los coeficientes del polinomio
    var.co_polos = coeff (real (var.polipolos));
    var.co_ceros = coeff (real (var.policeros)); //esto seria los coeficientes de un sistema FIR

    //Realizar la gràfica de magnitud
    //gainfunction (real (var.policeros),real (var.polipolos))

    gainfunction (var.co_ceros,var.co_polos)

endfunction


function gainfunction (num, den)
    global var
    global handles

//Declarar el sistema con uso de constantes (z).

    a = sca (handles.imag_respFreq);
    frq = linspace (1d -  6,0.5 -  1d -  6,10 * size (var.co_ceros," * "));
    [frq,repf] = repfreq (syslin ("d",poly (num,"z","c"),poly (den,"z","c")),frq);
    var.repf = repf;
    delete (a.children) //limpear la pantalla de imag frec

    fmax = frq (:) * var.f_m;   //Frec de muestreo y la freq para intervalo
    var.fmax = fmax;
    if var.gan == %T then
        [ma] = ganan_cero (); //Calcular para siempre llegar a cero su ganancia
        m = 20 * log10 (abs (ma (:))) //pasar a decibeles
```

```
        var.m = m;
        a.data_bounds = [0.001 min (m) * .98;max (fmax) max (m)] //  tamaño de gráfica de manera manual
                                                                    para cuando es mas pequeña
        var.gan = %f;
    else
        m = 20 * log10 (abs (var.repf (:)));
        var.m = m;
        a.data_bounds = [0.001 min (m) * .98;max (fmax) max (m)] //Auto  tamaño de gráfica de manera
manual                                                              para cuando es mas pequeña
        var.gan = %f;
    end

    plot (fmax,m)
    sca (handles.imag_circUnit);  //Devolver el manejador del pntero a la grafica del círculo unitario.
    Numeracion ()
endfunction


function borrar (win, x, y, ibut)
    //funcion para borrar un cero o polo
    global var;
    global handles;

    //indice  =  -  1; //indicador de que no se han encontrado ningun polo o cero al dar click

    if ibut ==  -  1000 then return,end      //Concluimos el evento "Genraiz" si se cierra le ventana   (figura)
    [X,Y] = xchange (x,y,'i2f')          //Nos regresa las coordenadas x,y de pixeles a escala del AXES que se
                                                está manejando, y X Y en coord de pixeles
    xinfo (msprintf ('Event code %d at mouse position is (%f,%f)',ibut,X,Y)) //Pone dentro de la Ventana, en
                                    la posicion de abajo, los datos de que evento de click se dió y las coord x,y

    if  ibut  <> 3 then
        return;
    end

    if ibut  == 3 then   //Evento de un click y soltar
        [cuadrante, indice] = busca (X,Y);
        var.c1 = cuadrante;
        var.in_1 = indice;
        if indice  <>  -  1 then
            var.temp = var.cuadrante (cuadrante)(indice);    //Crear un temporal  struct
            if var.temp.conjugado <>  -  1 then
                [cuadrante2,indice2] = busca (var.temp.x, -  var.temp.y);                  //busca el conjugado
                var.c2 = cuadrante2;
                var.in_2 = indice2;
                var.temp_conj = var.cuadrante (cuadrante2)(indice2)

                if var.in_1 == var.in_2 & var.c1 == var.c2 then        //esto es para evitar si existen muchos polos o
                                                ceros en ese lugar, poder tener el siguiente de la lista y no tome el mismo
                    var.in_2 = var.in_2 + 1;
                    var.temp_conj.figura.visible = 'off';
                    var.cuadrante (var.c2)(var.in_2) = null ();   //Borrar el conjugado
                else
                    var.temp_conj.figura.visible = 'off';
                    var.cuadrante (var.c2)(var.in_2) = null ();   //Borrar el conjugado
                end
```

```
        end
        var.temp.figura.visible = 'off';
        var.cuadrante (var.c1)(var.in_1) = null ();   //Borrar el polo o cero
        var.temp  =  '';
        var.temp_conj = '';
        handles.value_magnitude.String = '';
        handles.value_angle.String = '';
      end
    end
    polinomio ()
endfunction


function auto_upd ()
    global var
    global handles
    //Buscar todas las X y Y de cada polo o cero creado , esto dentro de la lista de lista
    //Iniciar al búsqueda para las multiplicacion de polinomios
    z = %z;             //definimos Z como transformada Z
    var.polipolos = 1;
    var.policeros = 1;

    for cuadra = 1:4
      for i = 1:size (var.cuadrante (cuadra))
        if var.cuadrante (cuadra)(i).tipo  == 'x' then
          if var.cuadrante (cuadra)(i).x ~= 0 & var.cuadrante (cuadra)(i).y ~= 0 then
            var.polipolos = var.polipolos * (z -  (var.cuadrante (cuadra)(i).x + (%i * var.cuadrante
(cuadra)(i).y)));
          end
        else
          var.policeros =  var.policeros * (z -  (var.cuadrante (cuadra)(i).x + (%i * var.cuadrante
(cuadra)(i).y)));
        end
      end
    end
    if var.temp.tipo == 'x' then
      var.polipolos = var.polipolos * (z -  (var.temp.x + (%i * var.temp.y)));
      if var.temp.conjugado < > - 1 then
        var.polipolos  =  var.polipolos * (z -  (var.temp_conj.x + (%i * var.temp_conj.y)));
      end
    else
      var.policeros  =  var.policeros * (z -  (var.temp.x + (%i * var.temp.y)));
      if var.temp.conjugado < > - 1 then
        var.policeros  =  var.policeros * (z -  (var.temp_conj.x + (%i * var.temp_conj.y)));
      end
    end
    var.co_polos = coeff (real (var.polipolos));
    var.co_ceros = coeff (real (var.policeros)); //esto seria los coeficientes de un sistema FIR

    //Realizar la gràfica de magnitud
    gainfunction (var.co_ceros,var.co_polos)

endfunction
```

```
function Mod_visual () //Esta función se encargará de al momento de seleccionar se modifique el  tamaño del
polo o ceroi
   global var
   global handles
   if var.atrapado_c == %T then
      if var.temp ~= '' then
         var.temp.figura.mark_size = 8;
         var.temp.figura.mark_foreground = 3;
      end

      if handles.select_conjugate.value == 1 then
         if var.temp.conjugado < > - 1 then
            if var.temp_conj < > '' then
               var.temp_conj.figura.mark_size = 8;
               var.temp_conj.figura.mark_foreground = 3;
            end
         end
      end
   end
   var.temp_mod = ''; //temporal secundario, para poder hacer la modificación visual sin recurrir al temp
                                                                                                    normal
   var.temp_mod = '';

   if var.atrapado_c == %F then
      if var.temp ~= '' then
         var.temp.figura.mark_size = 6;
         var.temp.figura.mark_foreground = 2;
         if handles.select_conjugate.value == 1 then
            if var.temp.conjugado < > - 1 then
               if var.temp_conj < > '' then
                  var.temp_conj.figura.mark_size = 6;
                  var.temp_conj.figura.mark_foreground = 2;
               end
            end
         end
      end
   end
endfunction

function up_magnitud (handles)
   global handles;
   global var;
   //Convertimos X y Y en forma polar desde inicio en el string
   if (handles.value_magnitude.Value)~= eval (handles.value_magnitude.String) then   //Checar que no
                                        sea el mismo dato sin modificar para evitar crear otro polo o cero
      //Evaluar lo que hay dentro de el string de magnitud
      if (handles.value_magnitude.String)~='' then
         handles.value_magnitude.Value  = eval (handles.value_magnitude.String);
         var.v_mag = handles.value_magnitude.Value;
      end

      //Asignar los valores modificados al temporal
      if var.temp ~ =  '' then
         pol2rect ();
         //Asignar el valor de magnitud a conjugado
```

```
        var.temp.figura.data = [var.temp.x var.temp.y];                    //Añdimos el cambio visual

        if var.temp.conjugado <> - 1 then
            var.temp_conj.figura.data = [var.temp_conj.x var.temp_conj.y];
        end
    else
        handles.value_magnitude.String = '';

    end

    if (var.atrapado_c) == %T then
        var.cuadrante (var.c1)(var.in_1) = null ();   //Borrar el polo o cero
        if var.temp.conjugado <> - 1 then
            var.cuadrante (var.c2)(var.in_2) = null ();   //Borrar el conjugado
            enlista (var.temp_conj);
        end
        enlista (var.temp);
        polinomio ();

        [cuadrante, indice] = busca (var.temp.x,var.temp.y);
        var.c1 = cuadrante;
        var.in_1 = indice;
        if var.temp.conjugado <> - 1 then
            [cuadrante2,indice2] = busca (var.temp.x, - var.temp.y);
            //Asignar las variables que seran usadas en la modificación por texto
            var.c2 = cuadrante2;
            var.in_2 = indice2;
        end
    end
  end
endfunction


function up_x (handles)   //Funcion para crear una relacion entre el texto de coord X y modificación
    global handles;
  global var;

  if (handles.value_magnitude.Value)~= eval (handles.value_magnitude.String) then   //Checa que no sea el
                                                    mismo dato sin modificar para evitar crear otro polo o cero
    //Evaluar lo que hay dentro de el string de magnitud
    if (handles.value_magnitude.String)~ = '' then
        handles.value_magnitude.Value = eval (handles.value_magnitude.String);//eval
(handles.value_magnitude.String);
        if (handles.value_magnitude.Value < - 10) then
            disp ('La cordenada X es muy pequeÑa');
            handles.value_magnitude.Value = - 10;
            handles.value_magnitude.String = msprintf ('%f',handles.value_magnitude.Value);
        else if (handles.value_magnitude.Value>10)
                disp ('La coordenada y es muy grande');
                handles.value_magnitude.Value = 10;
                handles.value_magnitude.String = msprintf ('%f',handles.value_magnitude.Value);
            end
        end
    end
    //Asignar los valores modificados al temporal
    if var.temp ~= '' then
```

```
        var.temp.x  = eval (handles.value_magnitude.String);          // Asignar el valor de magnitud value
                                                                                                    a X
        if var.temp.conjugado < > - 1 then
            var.temp_conj.x = eval (handles.value_magnitude.String);       //Asignar el valor de magnitud a
                                                                                                    conjugado
            var.temp_conj.figura.data = [var.temp_conj.x var.temp_conj.y];
        end

        var.temp.figura.data = [var.temp.x var.temp.y];                    //AÑadimos el cambio visual
    else
        handles.value_magnitude.String  = '';
    end

    if (var.atrapado_c) == %T then
        var.cuadrante (var.c1)(var.in_1) = null ();   //Borrar el polo o cero
        if var.temp.conjugado < > - 1 then
            var.cuadrante (var.c2)(var.in_2) = null ();   //Borrar el conjugado
            //provocar que sean los conjugados nuevos
            var.temp.conjugado = var.temp_conj;
            var.temp_conj.conjugado =  var.temp;
            enlista (var.temp_conj);
        end
        enlista (var.temp);
        polinomio ();
        //Ahora asignar el nuevo valor de de cuadrante e indice
        [cuadrante, indice] = busca (var.temp.x,var.temp.y);
        var.c1 = cuadrante;//Asignar las variables que seran usadas en la modificación por texto
        var.in_1 = indice;
        if var.temp.conjugado < > - 1 then
            [cuadrante2,indice2] = busca (var.temp.x, - var.temp.y);
            var.c2 = cuadrante2;
            var.in_2 = indice2;
        end

    end
  end
endfunction


function up_y (handles)//Funcion para crear una relacion entre el texto de coord  y modificación
  //Evaluar lo que hay dentro del string de angle
  global handles;
  global var;

  if (handles.value_angle.Value) ~=(eval (handles.value_angle.String)) then   //Checar que no sea el
                                          mismo dato sin modificar para evitar crear otro polo o cero
    //Evaluar lo que hay dentro de el string de magnitud
    if (handles.value_angle.String)~= '' then
        handles.value_angle.Value  = eval (handles.value_angle.String);//eval
        if (handles.value_angle.Value < - 10) then
            disp ('La cordenada Y es muy pequeÑa');
            handles.value_angle.Value  =  - 10;

            handles.value_angle.String = msprintf ('%f',handles.value_angle.Value);
        else if (handles.value_angle.Value>10)
                disp ('La cordenada Y es muy grande');
```

```
            handles.value_angle.Value = 10;
            handles.value_angle.String = msprintf ('%f',handles.value_angle.Value);
          end
        end
      end
      //Asignar los valores modificados al temporal
      if var.temp ~= '' then
        var.temp.y = handles.value_angle.Value;          // Asignar el valor de magnitud value a X
        if var.temp.conjugado <> -1 then
          var.temp_conj.y = -(var.temp.y);       //Asignar el valor de magnitud a conjugado
          var.temp_conj.figura.data = [var.temp_conj.x var.temp_conj.y];              //Añadimos cambio
                                                                                       visual al conjugado
        end
        var.temp.figura.data = [var.temp.x var.temp.y];                        //AÑadimos el cambio visual

      else
        handles.value_angle.String = '';

      end

      if (var.atrapado_c) == %T then
        var.cuadrante (var.c1)(var.in_1) = null ();  //Borrar el polo o cero
        if var.temp.conjugado <> -1 then
          var.cuadrante (var.c2)(var.in_2) = null ();  //Borrar el conjugado
          enlista (var.temp_conj);
        end
        enlista (var.temp);
        polinomio ();


        [cuadrante, indice] = busca (var.temp.x,var.temp.y);
        var.c1 = cuadrante;          //Asignar las variables que seran usadas en la modificación por texto
        var.in_1 = indice;

        if var.temp.conjugado <> -1 then
          [cuadrante2,indice2] = busca (var.temp.x, -var.temp.y);
          var.c2 = cuadrante2;
          var.in_2 = indice2;
        end
      end
    end
endfunction


function up_ángulo (handles)
  global handles;
  global var;

  //Convertimos X y Y en forma polar desde inicio en el string
  if (handles.value_angle.Value) ~= eval (handles.value_angle.String) then
    //Evaluar lo que hay dentro de el string de magnitud
    if (handles.value_angle.String)~='' then
      handles.value_angle.Value = eval (handles.value_angle.String);
      var.v_ang = handles.value_angle.Value;
    end
```

```
    //Asignar los valores modificados al temporal
    if var.temp ~= '' then
       pol2rect ();
       //Asignar el valor de magnitud a conjugado
       var.temp.figura.data = [var.temp.x var.temp.y];                    //Añadimos el cambio visual
       if var.temp.conjugado < > - 1 then
          var.temp_conj.figura.data = [var.temp_conj.x var.temp_conj.y];
       end
    else
       handles.value_angle.String = '';
       //   var.atrapado_c = %F;
    end

    if (var.atrapado_c) == %T then
       var.cuadrante (var.c1)(var.in_1) = null ();  //Borrar el polo o cero
       if var.temp.conjugado < > - 1 then
          var.cuadrante (var.c2)(var.in_2) = null ();  //Borrar el conjugado
          enlista (var.temp_conj);
       end
       enlista (var.temp);
       polinomio ();

       [cuadrante, indice] = busca (var.temp.x,var.temp.y);
       var.c1 = cuadrante;          //Asignar las variables que seran usadas en la modificación por texto
       var.in_1 = indice;
       if var.temp.conjugado < > - 1 then
          [cuadrante2,indice2] = busca (var.temp.x, - var.temp.y);
          var.c2 = cuadrante2;
          var.in_2 = indice2;
       end
    end
  end
endfunction


function Select_p_z (win, x, y, ibut)// Con esta funciòn seleccionar un par de polos o ceros modificar
visualmente para poder diferenciar  y conoces su coordenadas X y Y asi como tambien se  pueden modificar
  global var;
  global handles;

  indice = - 1; //indicador de que no se han encontrado ningun polo o cero al dar click

  if ibut == - 1000 then return,end      //Concluimos el evento "Genraiz" si se cierra le ventana   (figura)
  [X,Y] = xchange (x,y,'i2f')          //Nos regresa las coordenadas x,y de pixeles a escala del AXES que se
                                                                  está manejando, y X Y en coord de pixeles
  xinfo (msprintf ('Event code %d at mouse position is (%f,%f)',ibut,X,Y)) //Pone dentro de la Ventana, en
                                  la posicion de abajo, los datos de que evento de click se dió y las coord x,y

  if ibut == 3 then   //Evento de un click y soltar
     var.atrapado_c = %F;
     Mod_visual ();          //Provocar que los polos o ceros seleccionados se vean de forma que se
                                                                        diferencìe de los demas

     [cuadrante, indice] = busca (X,Y);
     var.c1 = cuadrante;          //Asignar las variables que seran usadas en la modificación por texto
     var.in_1 = indice;
```

```
    if indice  ==  -  1 then    //Si no encontro nada en donde dimos click , simplemente no hace nada
       var.c1 = '';          //Asignar las variables que seran usadas en la modificación por texto
       var.c2 = '';
       var.in_1 = '';
       var.in_2 = '';
       handles.value_magnitude.String = '';
       handles.value_angle.String = '';
       var.temp = '';
       var.temp_conj = '';
       return;
    end

    var.temp = var.cuadrante (cuadrante)(indice);    //Crear un temporal  struct
     if var.temp.conjugado < > -  1 then
          [cuadrante2,indice2] = busca (var.temp.x, -  var.temp.y);                    //busca el conjugado
          var.c2 = cuadrante2;
          var.in_2 = indice2;
          if var.in_2 < > -  1 then //Debug
             if var.in_1 == var.in_2 & var.c1 == var.c2 then        //esto es para evitar si existen muchos polos
                                        o ceros en ese lugar, poder tener el siguiente de la lista y no tome el mismo
                var.in_2 = var.in_2 + 1;
                var.temp_conj = var.cuadrante (var.c2)(var.in_2);      //Crear la variable temporal
                                                                        conjugada que es con la cual ge

             else

                var.temp_conj = var.cuadrante (var.c2)(var.in_2);

             end
          end
       end

    var.atrapado_c = %T;
    Mod_visual ();          //Provocar que los polos o ceros seleccionados se vean de forma que se
                                                                        diferencìe de los demas
    rect2pol ();          //este comando se encarga tambien de mostar en los reecuadros las coordenadas, no
                                                                        nadamas calcular
  end

endfunction


function filt (handles)
  global f1
  global handles
  f1.visible = 'on';
endfunction


function [ma] = ganan_cero (handles)
  global var
  global handles

  m = max (real (var.repf));
```

```
    x = 1/m;
    handles.value_gain.Value = x; //Ganancia necesaria
    ma = var.repf * x;
    handles.value_gain.String =  msprintf ('%5.9f',handles.value_gain.Value);
endfunction


function frec_muestreo (handles)
    global var;
    global handles;
    if (handles.value_Frec_m.Value) < >(eval (handles.value_Frec_m.String)) then   //Checar que no sea el
                                                        mismo dato sin modificar para evitar crear otro polo o cero
        var.f_m = eval (handles.value_Frec_m.String);
        if var.f_m < =  0 then
            var.f_m = 1;
            handles.value_Frec_m.Value  =  1;
            handles.value_Frec_m.String  =  msprintf ('%f',handles.value_Frec_m);
        end
        polinomio ()
    end
endfunction


function coficiente (handles)
    global var
    global handles
    handles.txt_num.Visible = 'on'
    handles.txt_den.Visible = 'on'
    handles.value_num.Visible = 'on'
    handles.value_den.Visible = 'on'

    handles.value_num.String = string (var.co_ceros);
    handles.value_den.String = string (var.co_polos);
endfunction


function logarit (handles)
    global var
    global handles

    if handles.select_logarit.Value  == 1 then
        handles.imag_respFreq.log_flags = 'lnn';
    else
        handles.imag_respFreq.log_flags = 'nnn';
    end

endfunction


function Cerrar ()
global f1
global f

    delete (f1);
    delete (f);
    clear;
```

```
    clearglobal;
    clc;
endfunction


function val_gan (handles)
    global var
    global handles
    a = sca (handles.imag_respFreq); //limpear la pantalla de imag frec

    if (handles.value_gain.Value) ~= eval (handles.value_gain.String) then   //Checar que no sea el mismo
                                                                                dato sin modificar
        //Evaluar lo que hay dentro de el string de magnitud
        if (handles.value_gain.String) ~= '' then
            delete (a.children);

            handles.value_gain.Value  = eval (handles.value_gain.String;
            var.v_gan = handles.value_gain.Value;
            ma = var.repf * var.v_gan;
            m = 20 * log10 (abs (ma (:))) //pasar a decibeles
            var.m = m;
            a.data_bounds = [0.001 min (m) * .98;max (var.fmax) max (m) * 1.25] //Auto  tamaño de gráfica de
                                                                    manera manual para cuando es mas pequeña
            plot (var.fmax,m)
            sca (handles.imag_circUnit);  //Devolver el manejador del pntero a la grafica del círculo unitario.
        end
    end
endfunction


function Numeracion (handles)
    global var
    global handles
    a = sca (handles.imag_circUnit);
    var.coincide = %f;

    //limpear marcadores de número
    k = var.cantidad_marcadores;
    i = 1;
    while (k>0)
        if f.children (1).children (i).tag == "text" then
            delete (f.children (1).children (i))
            i = 1; //reiniciar contador pues al eliminar un manejador la lista cambia
            k = k -  1;
        else
            i = i + 1;
        end
    end
        var.cantidad_marcadores = 0;

    for cuadra = 1:4  //checar en cada cuadrante
        for i = 1:size (var.cuadrante (cuadra))

            if i ==  1 then //Solo para la primera estructura de la lista
                var.list_numer (1) =  var.cuadrante (cuadra)(1);
                var.list_numer (1).cantidad = 1;
```

```
        end

        if i>1 then
            [enc] = busca_coinc (cuadra,i)

            if enc == %f then  //No encontró coincidencia
                var.list_numer ($ + 1) =  var.cuadrante (cuadra)(i);  //agregar a la lista para comparar
                var.list_numer ($).cantidad = 1;//Asignar su cantidad de repeticiones
            end
        end
        i = i + 1;
    end //Hasta aqui ya se llenaron la cantidad de repetidos

    //Hacer un ciclo para dibujar sus repetidos
    if var.list_numer < > list () then
    for k = 1:size (var.list_numer)
        if var.list_numer (k).cantidad>1 then
            xstring (var.list_numer (k).x -  0.01,var.list_numer (k).y + 0.009,string (var.list_numer
(k).cantidad)) //tomado de la escala por default
            h = gce ();
            h.tag = "text"; //este marcador nor permite diferenciarlo para poderse eliminar cada que se
modifique los polos y ceros
            var.coincide = %f;
            contador = 1;
            var.cantidad_marcadores =  var.cantidad_marcadores + 1;
            k = k + 1;
        end
    end
    end
    //limpear la lista para el siguiente cuadrante
    var.list_numer = list ();
  end
endfunction


function [enc] = busca_coinc (cuadra, indice)
  global var
  global handles
  i = indice;
  enc = %F;

  for j = 1:size (var.list_numer)
    if var.cuadrante (cuadra)(i).x == var.list_numer (j).x & var.cuadrante (cuadra)(i).y == var.list_numer
(j).y then
        if var.cuadrante (cuadra)(i).tipo == var.list_numer (j).tipo then //checar que sean del mismo tipo
            var.coincide=%T;//Denotar que si existen varios en el mismo punto
            var.list_numer (j).cantidad = var.list_numer (j).cantidad + 1;    //aumentar el número
            enc = %T;
        end
    end
    j = j + 1
  end
endfunction
```

Referencias:

[1] "SISTEMAS DE COMUNICACION I /SISTEMAS TELEMATICOS II Filtro de señal FM." [Online]. Available: https://www.academia.edu/9594083/SISTEMAS_DE_COMUNICACION_I_SISTEMAS_TELEMATICOS_II_Filtro_de_se%C3%B1al_FM. [Accessed: 28-Apr-2016].

[2] "Introduction to the Filter Design and Analysis Tool (FDATool)." [Online]. Available: https://www.mathworks.com/examples/signal/mw/signal_product-FDAToolExample-introduction-to-the-filter-design-and-analysis-tool-fdatool. [Accessed: 28 - Apr - 2016].

[3] "Scilab Help." [Online]. Available: https://help.scilab.org/docs/5.5.2/en_US/index.html. [Accessed: 28-Apr-2016].

[4] "SciPy.org—SciPy.org." [Online]. Available: http://www.scipy.org/index.html. [Accessed: 28-Apr-2016].

[5] J. G. Proakis and D. G. Manolakis, *Tratamiento digital de señales*. Pearson Educación, 2007.

[6] "Toría del Procesado Digital de Señales" [Online]. Available: http://catarina.udlap.mx/u_dl_a/tales/documentos/lem/sandino_p_ma/capitulo2.pdf. [Accessed: 06-May-2016].

[7] A. V. Oppenheim, J. R. Buck, M. M. Daniel, A. C. Singer, S. H. Nawab, and A. S. Willsky, *Signals Systems Pie and Computer Explorations in Signals*. Pearson Education, Limited, 2003.

[8] M. J. Roberts, *Signals and Systems: Analysis of Signals Through Linear Systems*. McGraw-Hill Companies,Incorporated, 2003.

[9] "Sistemas LTI-sistemasLTI.pdf." [Online]. Available: http://www.sc.ehu.es/acwarila/PDI/Tema%204/sistemasLTI.pdf. [Accessed: 12-May-2016].

[10] C. B. Rorabaugh, *Notes on Digital Signal Processing: Practical Recipes for Design, Analysis, and Implementation*. Prentice Hall, 2011.

[11] "prácticas de tratamiento digital de señales.pdf." [Online]. Avaiable: http://agamenon.tsc.uah.es/Asignaturas/it/tds/apuntes/practica_sfd_2008_09.pdf [Accesed: 16-May-2016].

[12] "Microsoft Word - Apuntes Scilab.doc - ApuntesScilab.pdf." [Online]. Available: http://personal.us.es/echevarria/documentos/ApuntesScilab.pdf. [Accessed: 17-May-2016].

[13] "About Scilab/Scilab/Home-Scilab." [Online]. Available: http://www.scilab.org/scilab/about. [Accessed: 17-May-2016].

[14] “Scilab Module : GUI Builder.” [Online]. Available: https://atoms.scilab.org/toolboxes/guibuilder. [Accessed: 17-May-2016].

[15] Chin Luh Tan, *Scilab GUI Part 1*. 2011.