DISEÑO DE SOFTWARE EN GUI DE MATLAB PARA EL ANALISIS Y DISEÑO DE SISTEMAS DE CONTROL EN BASE A ESPACIO DE ESTADOS DISCRETO

EDUARD ARMANDO RODRIGUEZ VASQUEZ

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÌA
PROGRAMA DE INGENIERÌA ELECTRÒNICA
NEIVA
2007

DISEÑO DE SOFTWARE EN GUI DE MATLAB PARA EL ANALISIS Y DISEÑO DE SISTEMAS DE CONTROL EN BASE A ESPACIO DE ESTADOS DISCRETO

EDUARD ARMANDO RODRIGUEZ VASQUEZ

Monografía presentada como requisito para optar el título de Ingeniero Electrónico

Director
AGUSTIN SOTO OTALORA
Ingeniero Electrónico
Especialista en Automatización Industrial

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÌA
PROGRAMA DE INGENIERÌA ELECTRÒNICA
NEIVA
2007

NOTA DE ACEPTACION

_____

_____

_____

_____

Firma del presidente del Jurado

_____

Firma del Jurado

_____

Firma del Jurado

Neiva, _____de 2007

Dedico este trabajo a mis padres, a mi esposa, a mi hija María José y a mis hermanos; los cuales han sido fuente de motivación constante en todo mi proceso de formación y hoy están conmigo celebrando la culminación de este ciclo de mi vida.

AGRADECIMIENTOS

Hoy quiero decir gracias…. Gracias a ustedes compañeros por estar conmigo en el largo camino del éxito, encontrando tropezones y gozando de alegrías. Gracias a mis docentes por enseñarme a explorar la mente y ayudarme a escudriñar cada rincón de mi cerebro; Gracias a mis padres quienes me brindaron su apoyo e hicieron de mi una persona integra, a mi esposa que con su compresión y cariño me ha acompañado.

Pero muy especialmente gracias a ti Dios y padre celestial, porque me orientaste en la búsqueda de una mejor formación, recta, llena de grandes valores, principios que muy seguramente serán el cimiento del profesional que hoy necesita y quiere ésta sociedad.

CONTENIDO

LISTA DE FIGURAS

Pág.

RESUMEN

El software diseñado esta compuesto por iconos y botones colocados de forma adecuada en una ventana de manera que el usuario pueda acceder de forma clara y fácil a cualquiera de ellos sin irse a confundir y conservando el orden. Es así como necesariamente el programa requiere parámetros para poder iniciar con el análisis, uno de estos es necesariamente la función de transferencia de la planta a controlar y para ello la interfaz posee tres opciones una en formato TF otro en ZPK y por ultimo en espacio de estado continuo. Luego, como se trabaja el sistema en discreto, se debe realizar el respectivo mapeo al plano Z, para ello se debe seleccionar un tiempo de muestreo T, esta opción se encuentra contenida en el software.

Posteriormente, el programa se encargará de generarle al usuario las matrices G, H, C, D las cuales son conocidas como el espacio de estado en discreto de la planta a analizar. Con esta información se procede a hacer el análisis como tal y para ello se le da la posibilidad de evaluar la ubicación de polos y ceros, y observar la posición de estos con respecto al círculo unitario, también se puede mirar sobre la grafica del circulo la posición de la ganancia crítica del sistema. Por otra parte se observan la respuesta en frecuencia en los diagramas de Bode de magnitud y fase, las respuestas impulso y paso las cuales son muy significativas y dicientes para el control de sistemas. Por último, el usuario tiene la posibilidad de determinar por medio de la interfaz la observabilidad y controlabilidad de la planta, los cuales son requisitos necesarios para culminar satisfactoriamente el proceso de análisis del sistema y poder pasar al diseño del regulador.

Ahora para el diseño del controlador, el software esta configurado de manera que requiere de unos parámetros iniciales tales como: El sobreimpulso deseado para la respuesta paso **(SI)** y el tiempo de establecimiento **(t_s)**; la segunda forma de ingresar los parámetros es por medio de los polos deseados del sistema.

Con los anteriores parámetros, el software procede a calcular el observador y la matriz de realimentación **(matriz K)** de tal manera que se obtenga el regulador completo y se cerrará el lazo de control.

Al final de todo el proceso se obtendrá el mismo sistema con el lazo de control. Se podrá validar nuevamente las gráficas de análisis antes mencionadas para constatar el correcto diseño del regulador.

INTRODUCCIÒN

Para los sistemas de control existen diferentes formas de representación en cuanto a su respectivo análisis y diseño, es por ello que siempre que se considera un sistema se debe determinar su forma mas aplicada y correcta de simular debido a que los prototipos y pruebas se hacen en software en donde fácilmente se puedan hacer cambios sin afectar el proceso real. Pero para estas analogías se debe escoger parámetros que reflejen el verdadero comportamiento de la planta real y para lo cual se debe proceder de la forma más sencilla de implementar de manera que se vean reflejadas la mayor cantidad de dinámicas y comportamientos del sistema en ecuaciones matemáticas manipulables para el sistema de cómputo.

Es de allí que una de las formas mas representativas en la actualidad son las matrices en espacio de estados, ya que como su nombre lo indica su estructura matricial facilita la manipulación adecuada en el software sin necesidad de hacer demasiadas transformaciones ni cálculos extensos.

Por ello he creado gracias a las herramientas gráficas del matlab (GUI) una aplicación en donde se crea una interfaz para facilitar las tareas del usuario y dar una ambientación en rutinas de análisis y diseño de este tipo de sistemas mucho más fácil y amenas para este.

Esta aplicación basa todo su análisis en sistemas discretos, más aun en evaluar parámetros de estabilidad, observabilidad y controlabilidad los cuales son requisitos necesarios para el posterior diseño e implementación de reguladores discretos para los sistemas de control de este tipo y la obtención de parámetros y respuestas deseadas de la planta a controlar.

1. MARCO TEÓRICO

1.1 DEFINICIONES

1.1.1 Estado. El conjunto más pequeño posible de variables se conoce como estado de un sistema dinámico tales que al conocer estos estados en $t = t0$ y la entrada en $t_0 \geq t$ determinaran el comportamiento en cualquier tiempo dentro del rango.

1.1.2 Variables de Estado. Variables de estado se conocen como un conjunto pequeño de estas que determinan el comportamiento del sistema dinámico; de manera que para un sistema de "n" variables se tienen "n" estados, expresados así: $x_1, x_2, x_3, \ldots\ldots, x_n$.

1.1.3 Vector de Estado. La reunión de "n" variables que describen el comportamiento del sistema dentro de un mismo vector se denominara **Vector de Estado** e irá representado como el **vector X.** Este vector se encargará de determinar la forma única del estado **X(t)** en cualquier tiempo y dependerá de las condiciones iniciales y la entrada del sistema.

1.1.4 Espacio de Estados. Es un espacio compuesto con todos los puntos de estados posibles dentro de "n" dimensiones que se determinan según la cantidad de estados para el sistema en particular y tendrá como ejes coordenados: $X_1, X_2, X_3, \ldots X_n$.

1.1.5 Ecuaciones en el Espacio de Estados. Para el modelamiento de sistemas dinámicos utilizando espacios de estado se trabajan con tres tipos de variables, entre las que están: variables de entrada, variables de salida y variables de estado.

De manera que para sistemas en tiempo discreto que sean lineales o no lineales y variantes en el tiempo, la ecuación de estados es:

$$\overline{x}(k+1) = \overline{f}\left[\overline{x}(k), \overline{u}(k), k\right]$$

Y la ecuación de salida:

$$\overline{y}(k) = \overline{g}\left[\overline{x}(k), \overline{u}(k), k\right]$$

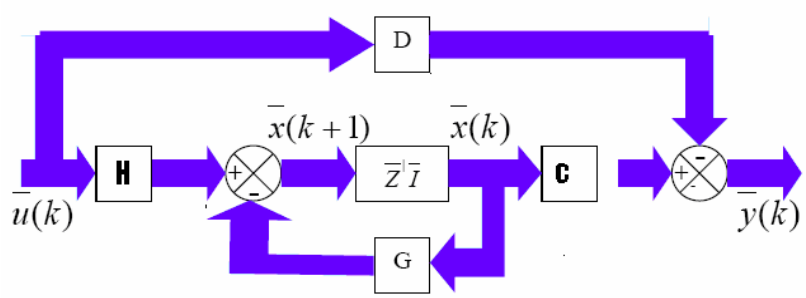
Las anteriores ecuaciones pueden representarse por:

$$\overline{x}(k+1) = \overline{G}(k)\overline{x}(k) + \overline{H}(k)\overline{u}(k)$$
$$\overline{y}(k) = \overline{C}(k)\overline{x}(k) + \overline{D}(k)\overline{u}(k)$$

x (k): Vector n (Vector de estado).
y (k): Vector m (Vector de salida).
u (k): Vector r (Vector de entradas).
G (k): Matriz n*n (Matriz de estado).
H (k): Matriz n*r (Matriz de entrada).
C (k): Matriz m*n (Matriz de salida).
D (k): Matriz m*r (Matriz de transmisión directa).

Teniendo en cuenta las anteriores ecuaciones y tomando las matrices G, H, C, D como constantes, de deduce el siguiente diagrama del espacio de estado discreto, así:

Figura 1. Diagrama de Espacio de Estados Discreto



Fuente: POLONIA, Jorge. CONTROL MODERNO. Control digital. Especialización Automatización industrial

1.1.6 Concepto de Controlabilidad y Observabilidad. El trabajo pionero de **R. Kalman** en el año de 1960 introdujo los conceptos de **Controlabilidad** y de **Observabilidad**, los cuales juegan un papel fundamental en el diseño de los sistemas de control usando técnicas de espacio de estado. De manera que las condiciones de controlabilidad y de observabilidad determinan la existencia de una solución completa para el problema del diseño de un sistema de control[1]. Ya que es posible que no exista una solución a este problema si el sistema estudiado es no controlable. Aunque la mayoría de los sistemas físicos son controlables y observables, los modelos matemáticos correspondientes pueden no tener la propiedad de controlabilidad o de observabilidad. En tal caso, es esencial conocer las condiciones bajo las cuales un sistema es controlable y observable.

[1] POLONIA, Jorge. CONTROL MODERNO. Control digital. Especialización Automatización industrial

Se dice que un sistema es controlable en el instante t_0 si es posible llevarlo de cualquier estado inicial $x(t_0)$ a cualquier otro estado, empleando un vector de control no acotado, en un lapso finito.

Se dice que un sistema es observable en el tiempo si, con el sistema en el estado **x(t)**, es posible determinar dicho estado a partir de las mediciones de la salida en lapso de tiempo finito.

1.1.7 Concepto de Estabilizabilidad. Cuando un sistema no es controlable de estado completo, pero sucede que la parte no controlable es estable, se dice entonces que el sistema es estabilizable, aunque no sea controlable. Un sistema controlable de estado completo es siempre estabilizable.

1.2 CONTROLABILIDAD COMPLETA DEL ESTADO PARA SISTEMAS EN TIEMPO DISCRETO

Un sistema de control es de estado completamente controlable, si es posible transferir el sistema de un estado inicial arbitrario a cualquier estado deseado también arbitrario, en un periodo finito (número finito de periodos de muestreo).

El concepto de controlabilidad juega un papel importante para ubicación de los polos en el sistema de control.

Se considera al sistema de control en tiempo discreto definido por:

$$x[(k+1)T] = Gx(kT) + Hu(kT)$$
$$y(kT) = Cx(kT) + Du(kT)$$

$$\bar{x}(nT) - G^n x(0) = \underbrace{\begin{bmatrix} H & \vdots & GH & \vdots & \cdots & \vdots & G^{n-1}H \end{bmatrix}}_{\text{Matriz de Controlabilidad}} * \begin{bmatrix} u(n-1)T \\ u(n-2)T \\ \vdots \\ u(0) \end{bmatrix}$$

donde

x(kT) = vector de estado de orden n en el periodo k de muestreo
u(kT) = vector de control de orden r en el periodo k de muestreo
y(kT) = vector de salida de orden m en el periodo k de muestreo

G = matriz de estado de orden n x n, constante**.**
H = matriz de control de orden n x r, constante**.**
C = matriz de salida de orden m x n, constante**.**
T = Periodo de muestreo**.**

1.3 CONTROLABILIDAD DE LA SALIDA DE SISTEMAS DE TIEMPO DISCRETO

En el diseño teórico de un sistema de control se prefiere controlar la salida en vez del estado del sistema.

El sistema de control dado por x[(k+1)T] y y(kT) es controlable a la salida si la matriz de m filas por (n+1)r columnas es de rango m. O bien:

$$\text{Rango}\left\lfloor D \;\vdots\; CH \;\vdots\; CGH \;\vdots\; \cdots \;\vdots\; CG^{n-1}H \right\rfloor = \text{m}$$

1.4 OBSERVABILIDAD COMPLETA DEL ESTADO PARA SISTEMAS EN TIEMPO DISCRETO

Se considera al sistema de control en tiempo discreto definido por

$$\mathbf{x[(k+1)T] = Gx(kT) + Hu(kT)}$$
$$\mathbf{y(kT) = Cx(kT) + Du(kT)}$$

donde

x(kT) = vector de estado de orden n en el periodo k de muestreo
u(kT) = vector de control de orden r en el periodo k de muestreo
y(kT) = vector de salida de orden m en el periodo k de muestreo

G = matriz de estado de orden n x n, constante.
H = matriz de control de orden n x r, constante.
C = matriz de salida de orden m x n, constante.
T = Periodo de muestreo.

Un sistema es completamente observable si dada la salida *y*(*k*) en un número finito de periodos de muestreo es posible determinar el vector inicial *x*(0). Esto quiere decir que un sistema cumple con esta condición si dado y(0) , y y(T) , y(2T) … es posible determinar $x_1(0)$, $x_2(0)$, …$x_n(0)$ en un numero finito de periodos de muestreo. Esto es:

$$\bar{y}(0) = C\bar{x}(0)$$
$$\bar{y}(T) = CG\bar{x}(0)$$
$$\vdots$$
$$y[(n-1)T] = CG^{n-1}\bar{x}(0)$$

$$\Rightarrow \begin{bmatrix} C \\ \cdots \\ CG \\ \cdots \\ \vdots \\ CG^{n-1} \end{bmatrix}$$

Como el rango de la matriz y de su transpuesta conjugada es el mismo esta matriz se puede expresar así:

$$\Rightarrow \left[C^* \quad \vdots \quad G^*C^* \quad \vdots \quad \cdots \quad \vdots \quad \left(G^*\right)^{n-1} C^* \right]$$

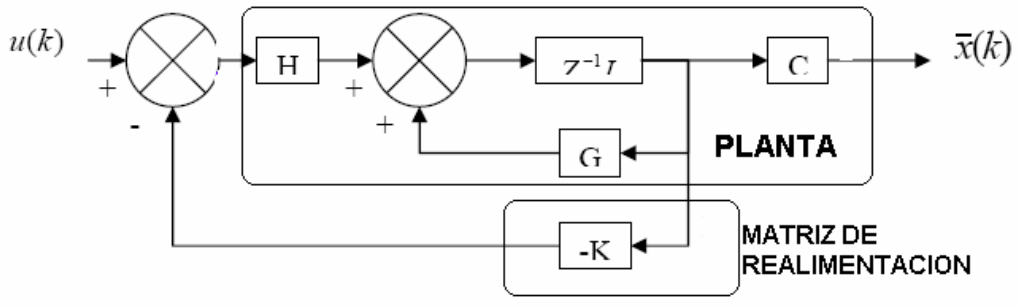
Esta matriz deberá ser de rango n para que el sistema sea completamente observable.

1.5 DISEÑO EN EL ESPACIO DE ESTADOS DISCRETO

1.5.1 Ubicación de polos. Considerando un sistema completamente controlable los polos podrán ubicarse en cualquier posición deseada, mediante la realimentación de una matriz de ganancia con los estados indicados.

A continuación se muestra un diagrama con el sistema en lazo cerrado con la matriz de realimentación.

Figura 2. Sistema de Lazo cerrado con Matriz de Realimentación



Fuente: POLONIA, Jorge. CONTROL MODERNO. Control digital. Especialización Automatización industrial

Ecuación característica:

$$|ZI - G + Hk = 0|$$

Con este se deberá garantizarse que la matriz de realimentación **K** sera de tal forma que los valores característicos sean los de los polos en lazo cerrado requeridos.

1.5.2 **Fórmula de Ackermann**. Por medio de éste método se busca determinar la matriz de ganancia de realimentación: *K*

Si el sistema es de estado completamente controlable y se desea ubicar los polos en lazo cerrado en $z = p_1$, $z = p_2$, … $z = p_n$

La ecuación característica es:

$$\left|Z\bar{I} - G + H\mathbf{k}\right| \quad (z-p_1)(z-p_2)\ldots(z-p_n)$$
$$= z^n + \alpha_1 z^{n-1} + \alpha_2 z^{n-2} + \ldots + \alpha_{n-1} z + \alpha_n = 0$$

$$K = \begin{bmatrix} 0 & 0 & \ldots & 1 \end{bmatrix} * \begin{bmatrix} H & \vdots & GH & \vdots & \cdots & \vdots & G^{n-1}H \end{bmatrix}^{-1} \phi(G)$$

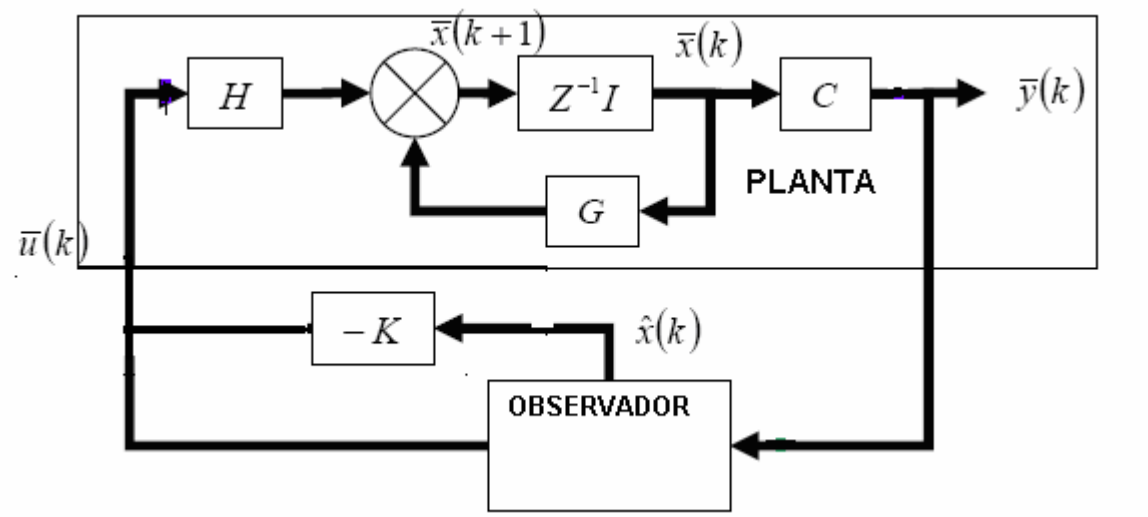
$$\phi(G) = G^n + \alpha_1 G^{n-1} + \alpha_2 G^{n-2} + \ldots + \alpha_{n-1} G + \alpha_n I$$

1.6 ESTIMADORES DE ESTADO

En los sistemas reales no todas las variables de estado están disponibles para medición directa, es allí donde los estimadores entran a ser parte fundamental del sistema de control. Un estimador es un subsistema cuya función es observar las variables a partir de las mediciones de las señales de salida $y(k)$ y control $u(k)$.

En el siguiente diagrama se observa como se integra dentro del lazo de realimentación este nuevo subsistema.

Figura 3. Sistema de Espacio de estado Discreto con Realimentación y Observador



Fuente: POLONIA, Jorge. CONTROL MODERNO. Control digital. Especialización Automatización industrial

El sistema de control esta definido por las ecuaciones de estado:

$$\left.\begin{aligned}\bar{x}(k+1) &= G\bar{x}(k)+H\bar{u}(k)\\ \bar{y}(k) &= C\bar{x}(k)\end{aligned}\right\}Planta$$

Supóngase que el estado **x(k)** debe aproximarse al estado ***x(k)*** estimado del modelo dinámico:

$$\left.\begin{aligned}\tilde{x}(k+1) &= G\tilde{x}(k)+H\bar{u}(k)\\ \tilde{y}(k) &= C\tilde{x}(k)\end{aligned}\right\}observador$$

Si las condiciones iniciales son las mismas, entonces el estado ***x(k)*** estimado y el estado ***x(k)*** son iguales. Si las condiciones iniciales son distintas, los estados son diferentes, para si **G** es una matriz estable ***x(k)*** se puede aproximar a ***x(k)*** estimado.

En ocasiones se puede monitorear el estado **x(k)** por medio de la salida **y(k)** y la salida ***y(k)*** estimada de manera que:

$$\tilde{x}(k+1) = G\tilde{x}(k)+H\bar{u}(k)+L\left[\bar{y}(k)-C\tilde{x}(k)\right]$$

L: Matriz de Ganancia del estimador

Ahora también se debe tener en cuenta que si el estimador calculado es completamente controlable y observable, **x(k)** no se puede obtener por medición directa, pero ***y(k)*** estimada si podrá medirse siempre y cuando se parta de un sistema de control con realimentación del estado indicado.

$$\bar{x}(k+1) = G\bar{x}(k) + H\bar{u}(k)$$
$$\bar{y}(k) = C\bar{x}(k)$$
$$\bar{u}(k) = -K\bar{x}(k)$$

K: Matriz de Ganancia de realimentación

1.7 REGULADOR

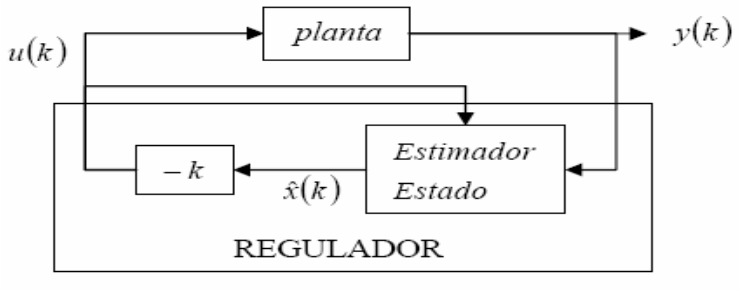
El regulador esta conformado por la ganancia de realimentación de estado K y la matriz de ganancia del estimador L.

En otras palabras se puede decir que el regulador nace de conectar la ley de realimentación de estado **u(k)=-KX** y el estimador con la matriz de ganancia **L** y se puede describir de la siguiente manera:

$$\hat{x}(k+1) = [G - LC - (H - LD)K]\hat{x}(k) + Ly(k) = \tilde{G}\hat{x}(k) + Ly(k)$$
$$\hat{x}(k+1) = C\hat{x}(k)$$
$$u(k) = -K\hat{x}(k)$$

El regulador se conecta a la planta usando realimentación positiva, como se muestra en el siguiente diagrama:

Figura 4. Planta con Regulador



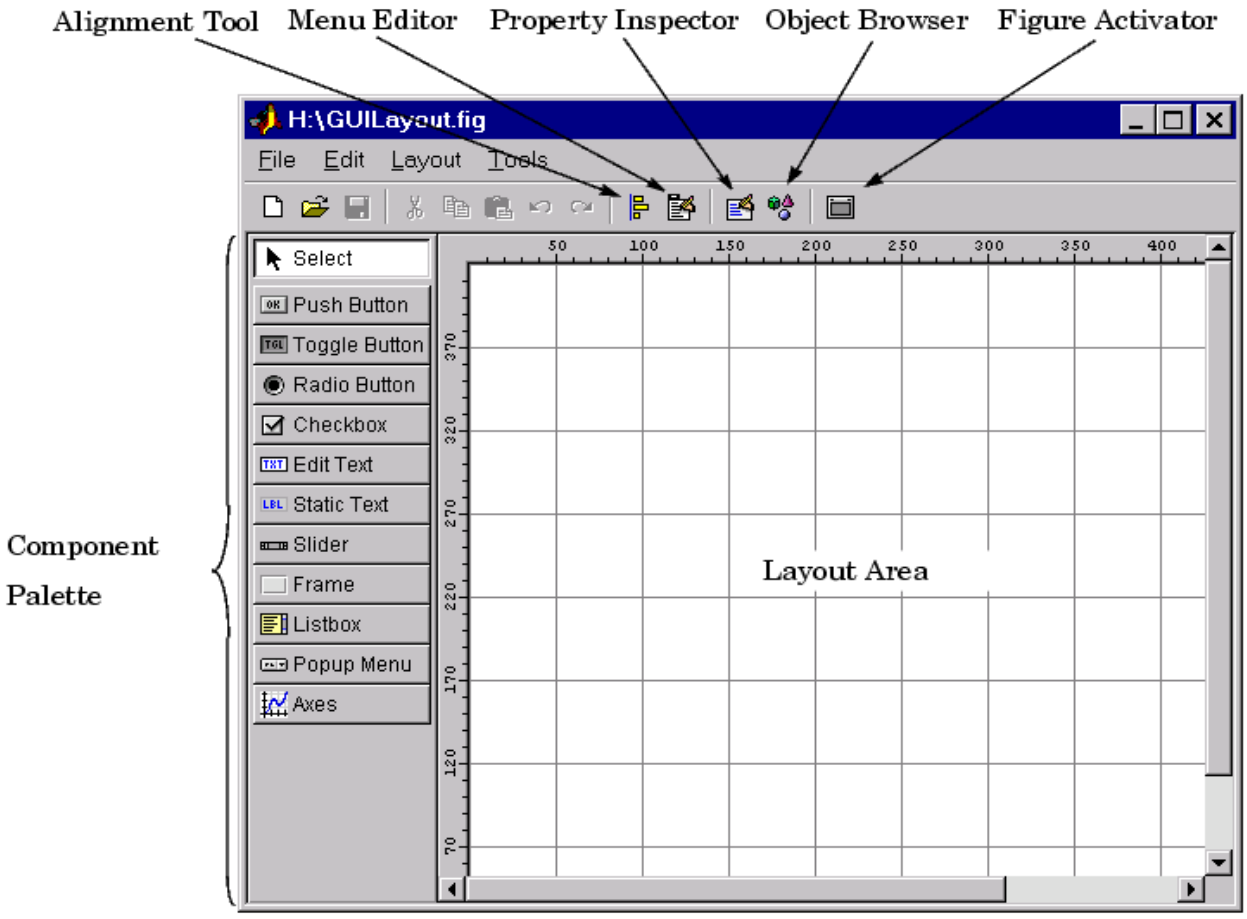
1.8 CONCEPTOS BÁSICOS DE GUI

1.8.1 Matlab GUI. Es un entorno de programación visual que ofrece Matlab para poder realizar y ejecutar programas de simulación de forma simple, tiene las características básicas de todos los programas visuales como Visual Basic o Visual C++.

1.8.2 Ejecución. Desde la ventana de comando del Matlab se debe ejecutar el comando **guide**.

Esto abre la consola de edición de la parte grafica de la aplicación a implementar (.fig), es decir, colocar botones, cuadros de dialogo, graficas, texto, etc.

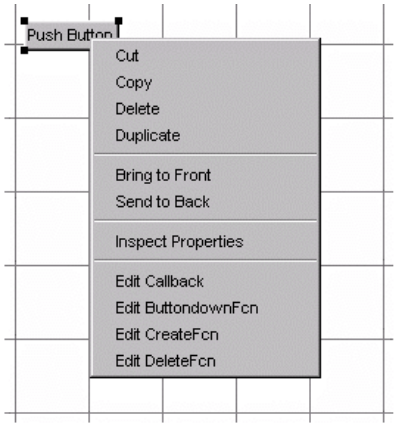
Figura 5. Ventana principal de GUIDE



Fuente: www.ingelec.uns.edu.ar/icd2763/tut.doc

Cada uno de estos elementos tienen un conjunto de propiedades a las cuales podemos acceder con el botón derecho del **mouse**, una vez damos **click** sobre cualquiera de estos objetos aparece el siguiente cuadro:

Figura 6. Cuadro de edición de propiedades



Fuente: www.ingelec.uns.edu.ar/icd2763/tut.doc

Para editar las propiedades de cada elemento seleccionamos la opción **Inspect Properties** y se abre una consola (la cual variará según que elemento se este editando) con todas las propiedades que podemos editar, entre las que están: color, posición, tamaño, fuente, etc.

Una de las opciones de mayor interés para nosotros en la figura anterior es **Edit Callback**. Esta última abre el archivo de extensión **.m** asociado (ejecutable Matlab) y nos posiciona en la sección del programa que corresponde a la subrutina que se ejecutará cuando se realice una determinada acción sobre el elemento que estamos editando.

1.8.3 Partes de una aplicación GUI. Consta de dos archivos uno ejecutable **(.m)** y otro la parte gráfica **(.fig)**. Las dos partes están unidas a través de las subrutinas **callback**. Una vez que se graba los archivos desde la consola de emisión (si salvamos la **.fig** automáticamente lo hace el **.m** asociado) se puede ejecutar el programa en la ventana de comando de Matlab solamente escribiendo el nombre del archivo.

Por ejemplo si se guarda un archivo **ratón.fig** y **ratón.m** escribiendo "**ratón**" y presionando **enter** se ejecuta el programa.

El archivo **.m** que se crea tiene una estructura predeterminada. Consta de un encabezado y a continuación viene el código correspondiente a las siguientes subrutinas.

function varargout = untitled1(varargin) ← Encabezado
% UNTITLED1 Application M-file for untitled1.fig
% FIG = UNTITLED1 launch untitled1 GUI.

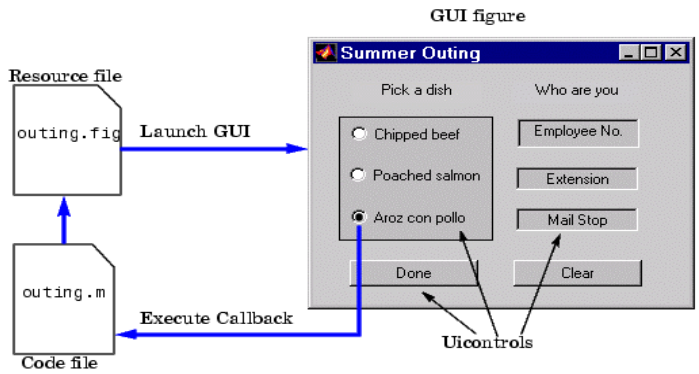
```
%    UNTITLED1('callback_name', ...) invoke the named callback.
% Last Modified by GUIDE v2.0 20-Aug-2002 19:57:33

if nargin == 0  % LAUNCH GUI

        fig = openfig(mfilename,'reuse');

        % Use system color scheme for figure:
        set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));

        % Generate a structure of handles to pass to callbacks, and store it.
        handles = guihandles(fig);
        guidata(fig, handles);

        if nargout > 0
                varargout{1} = fig;
        end

elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK

        try
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        catch
                disp(lasterr);
        end

end
```

handles, Puntero a todas variables y elementos de la aplicación

guidata(), comando para guardar las variables de la aplicación

La siguiente figura muestra la interacción entre la la figura **(.fig)** y el archivo **(.m)**:

Figura 7. Relación figura vs archivo ejecutable

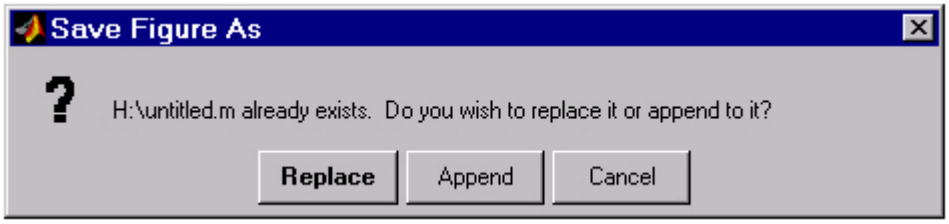


Fuente: www.ingelec.uns.edu.ar/icd2763/tut.doc

Observación sobre Salvar:

Si guarda una modificación del archivo cuando aparece el cuadro de dialogo (ver figura) que pregunta si hacemos **Replace** o **Append** debemos elegir **Append**, si se selecciona **Replace** borra todo el archivo **.m** es decir lo pone a cero, **no utilizar Replace.**[2]

Figura 8. Cuadro de Dialogo para guardar figura



Fuente: www.ingelec.uns.edu.ar/icd2763/tut.doc

[2] Fuente: www.ingelec.uns.edu.ar/icd2763/tut.doc

2. TUTORIAL PARA EL MANEJO DEL SOFTWARE

En este capitulo se dará una breve descripción de las funciones importantes de la interfaz, de manera que se tenga total claridad en el momento de realizar aplicaciones sobre esta.

El software se elaboro de manera que el usuario no tenga complicaciones a la hora de utilizarlo, para ello todo los botones y administración de este se encuentran en un mismo formulario como lo muestra la siguiente figura:

Figura 9. Pantalla inicial de la interfaz

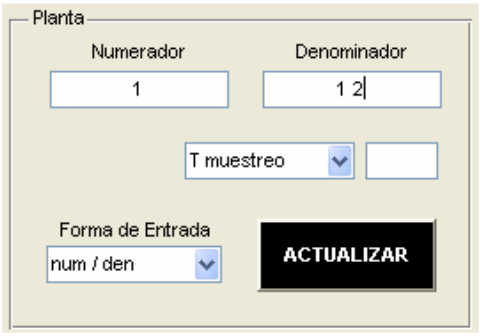
2.1 FORMATOS DE INGRESOS PARA EL SISTEMA A CONTROLAR

En esta opción, se podrá seleccionar tres tipos de entrada para el sistema o planta a partir de un menú desplegable entre las que están:

2.1.1 Formato numerador /denominador. En este formato los parámetros de entradas son:

El numerador de la función de transferencia de la planta, el denominador y el tiempo de muestreo.

Figura 10. Pantalla para parámetros de entrada num/den



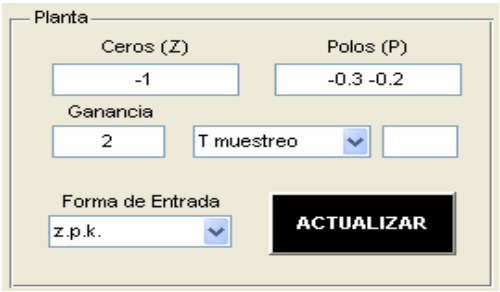
Nota 1: Los paréntesis rectos no son necesarios ya que se incluyen por programación.

Nota 2: Después de digitar un campo se debe dar **enter** para darle la entrada de datos a la variable asignada.

2.1.2. Formato Ceros, Polos y Ganancia. Para este formato los parámetros de entrada son:

Los ceros de la planta, los polos y la ganancia de este.

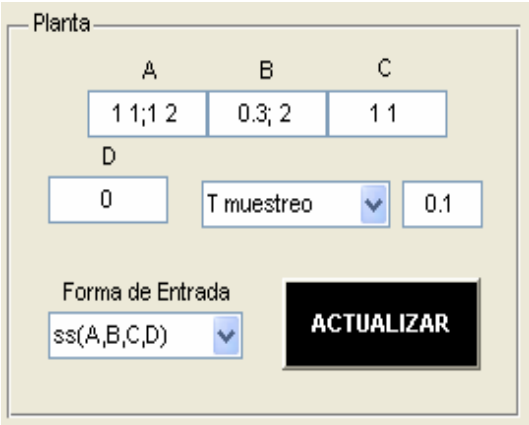
Figura 11. Pantalla para parámetros de entrada ZPK



2.1.3 Formato de Espacio de Estados Continuo. Por último se da una tercer opción y sus parámetros de entrada son:

Los campos de las matrices A, B, C y D teniendo en cuenta que cada fila debe ir separada por punto y coma (;).

Figura 12. Pantalla para parámetros de entrada en SS continúo

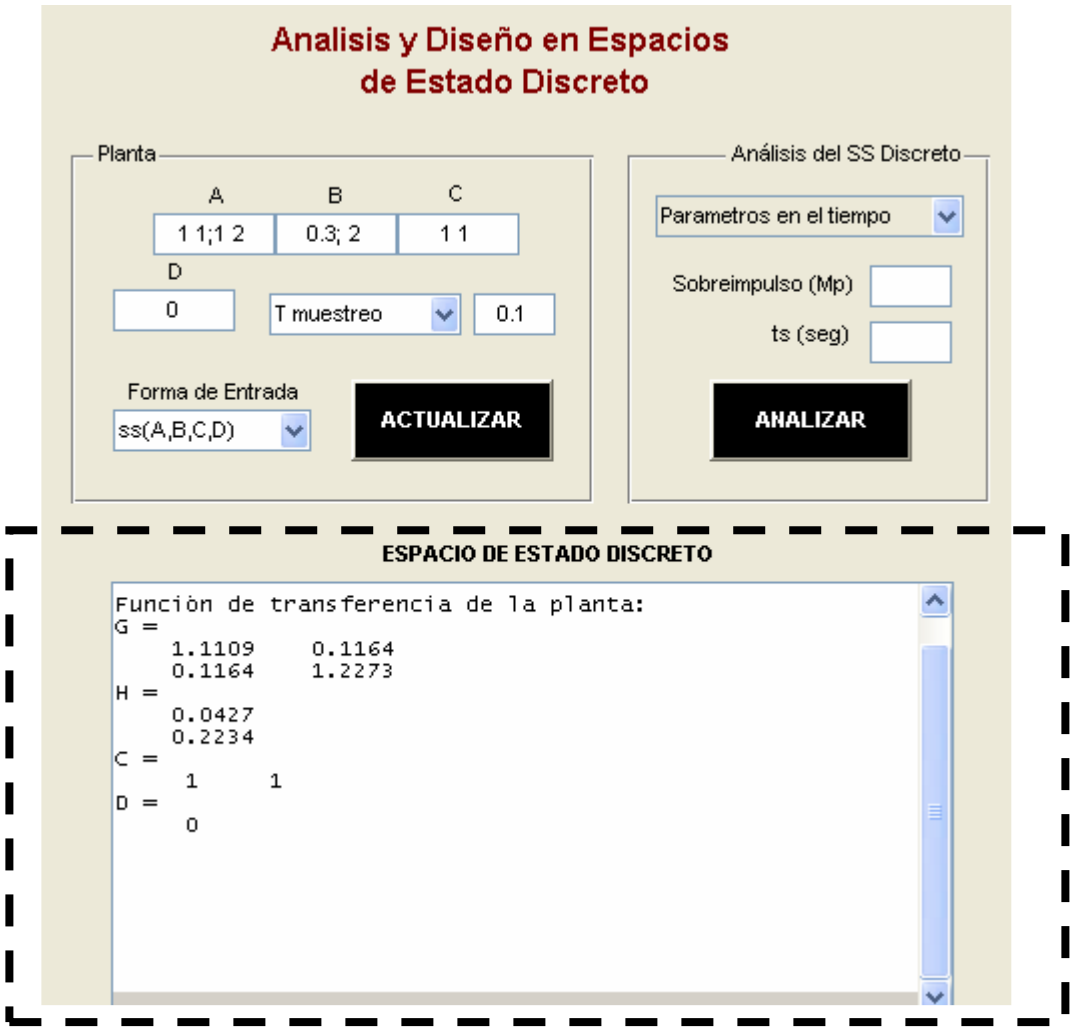


2.2 RESOLUCIÒN DEL ESPACIO DE ESTADO DISCRETO

Después de haber dado los parámetros entrada o los parámetros de la planta en cualquiera de los tres formatos anteriores, el programa efectuara las operaciones para encontrar el espacio de estado discreto, y para ello se debe hacer **click** sobre el botón "**ACTUALIZAR**" y se mostrara el resultado en el **list box** denominado "**ESPACIO DE ESTADO DISCRETO**", en este espacio se darán los resultados mas sobresalientes e importantes del proceso, de manera que el usuario puede disponer de ellos y al mismo tiempo ir revisando el proceso para corroborar los resultados de la interfaz.

Por ahora se visualiza el proceso que resulta de la conversión de la planta inicial al espacio de estado discreto en donde aparecerán los valores para las matrices G, H, C y D:

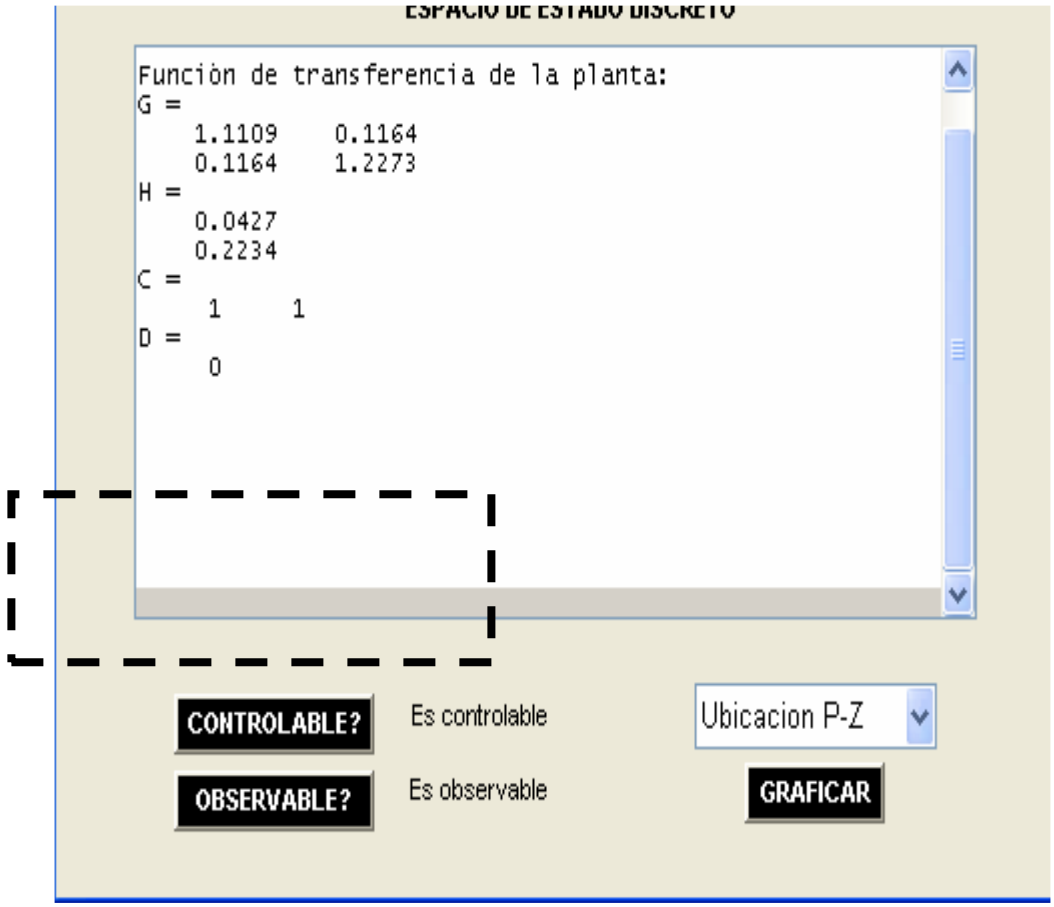
Figura 13. Pantalla para visualización del SS discreto



2.3 EVALUACION DE CONTROLABILIDAD Y OBSERVABILIDAD

Para la evaluación de los parámetros de controlabilidad y observabilidad el programa inicia con la tarea de determinar si cumple o no con estos requisitos; para ello se colocaron dos botones de forma independiente de manera que haciendo un simple **click** sobre cada uno de ellos el software arrojará si es o no observable y controlable los estados del sistema.

Figura 14. Pantalla para visualización de controlabilidad y observabilidad del sistema en SS discreto



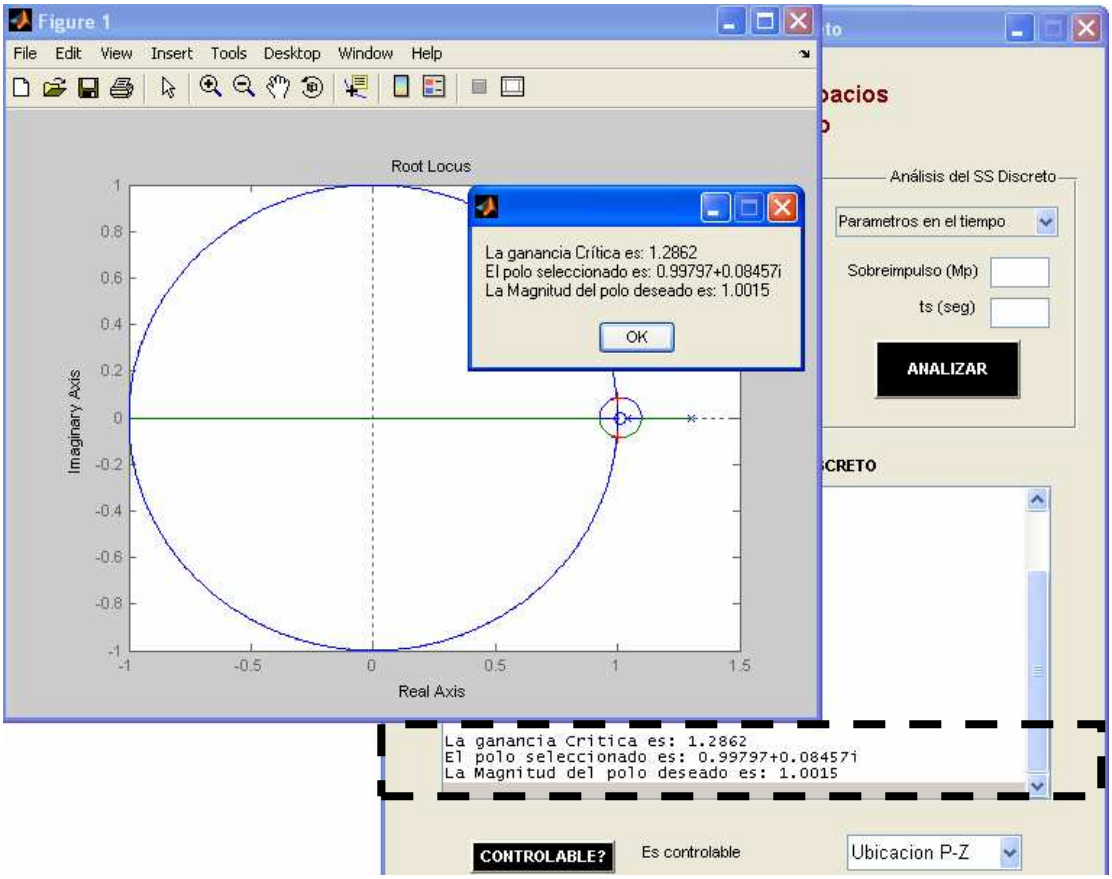
2.4 GRÁFICAS DE INTERÈS EN EL ANÀLISIS

Esta opción se encuentra en la parte inferior derecha del formulario, allí encontramos un menú desplegable con varias opciones, entre las que están:

- La ubicación de polos y ceros
- Respuesta Impulso
- Respuesta Paso
- Respuesta en Frecuencia

2.4.1 Ubicación de P – Z (Polos y ceros). Esta opción se ejecuta seleccionando de la lista "**Ubicación P-Z**" y luego dando **click** en el botón "**GRAFICAR**".

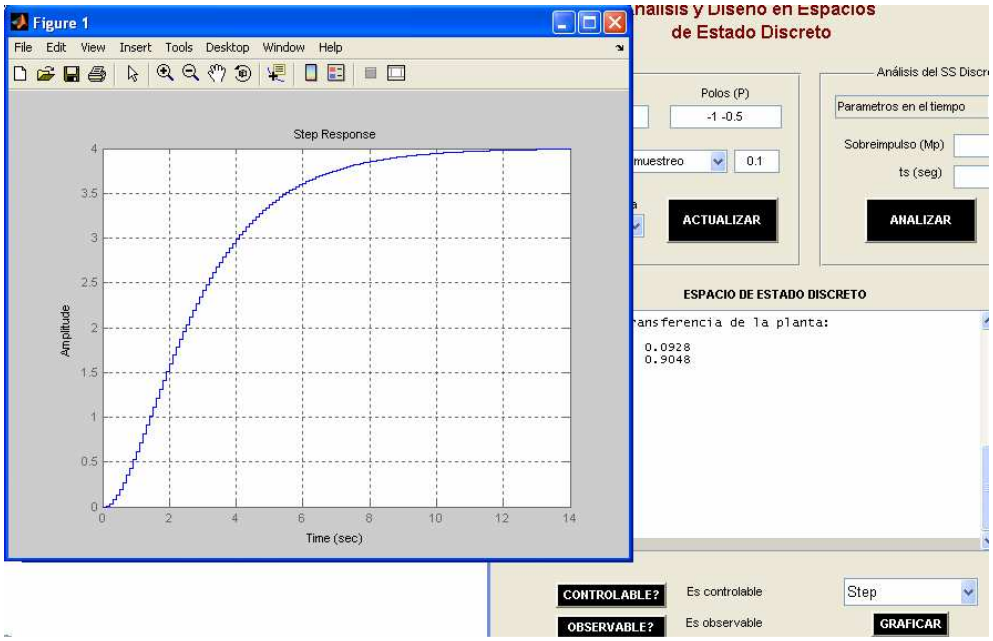
Figura 15. Pantalla para visualización del r-locus y ganancia critica



La gráfica generada será la ubicación de polos y ceros del sistema en discreto, también se observara el desplazamiento de los polos hacia los ceros a medida que varía la ganancia del sistema o en otras palabras se visualizara el **r-locus** de la planta. Agregado a esto, con un **click** sobre la grafica se muestra el valor de ese punto en la ventana de "**ESPACIO DE ESTADO DISCRETO**" , esta resulta ser una aplicación importante ya que se podrá utilizarse para obtener el valor de la ganancia critica **(Kcritico),** este valor saldrá en un **msg box** temporal que se generara en el momento de hacer la selección**.**

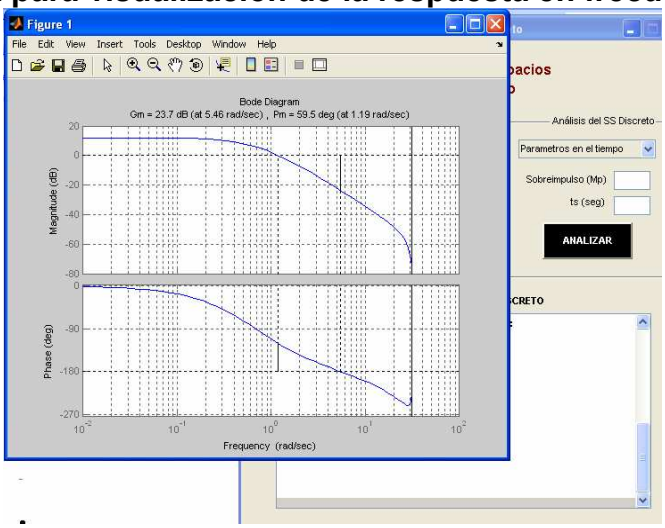
2.4.2 Gráfica de respuesta paso (step). Esta opción se puede tener dentro del **list box** y se generara de igual manera haciendo **click** sobre el botón de "**GRAFICAR**". De esta gráfica se puede obtener los parámetros del tiempo importantes que dará la partida para la escogencia del los ítems de diseño tales como son: sobreimpulso, tiempo de establecimiento, tiempo de subida, error en estado estacionario, etc.

Figura 16. Pantalla para visualización de la respuesta paso



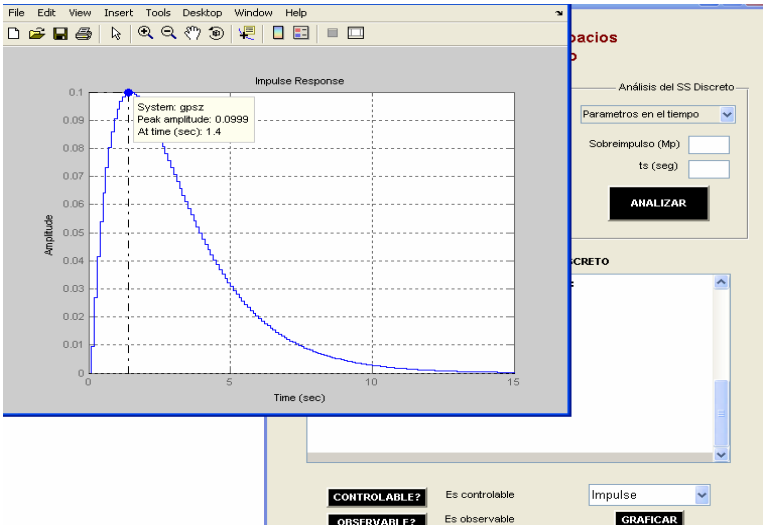
2.4.3 Gráfica de Respuesta en frecuencia (bode). No solo se tiene la opción de mirar la respuesta en el tiempo sino también en la frecuencia a través del Bode del sistema en lazo abierto, en donde se puede observar parámetros como: las frecuencias de corte de la planta inicial y los márgenes de ganancia y fase del mismo; de esta manera se brinda otro criterio mas de estabilidad del sistema no controlado y poder obtener otro criterio de mejoramiento y evaluación del futuro controlador.

Figura 17. Pantalla para visualización de la respuesta en frecuencia



2.4.4 Gráfica de Respuesta impulso. Por último se obtiene la respuesta impulso en donde se puede observar el funcionamiento del sistema a través de este criterio tan significativo, ya que como se conoce, esta señal es una creación solo de simulación y no física, pero que es de gran utilidad ya que se puede determinar la respuesta del sistema en todo el espectro de frecuencia.

Figura 18. Pantalla para visualización de la respuesta impulso



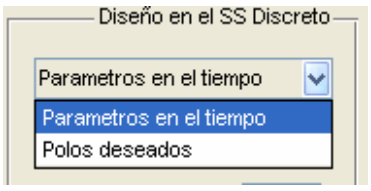
Nota 3: Las ventanas que se muestran con cada una de las gráficas pueden ser ejecutadas y quedarán a la disposición del usuario de forma simultánea o en su defecto las que requiera de manera que este quede a libertad del manejo sobre ellas.

2.5 DISEÑO DEL SISTEMA EN ESPACIO DE ESTADO DISCRETO

Previamente se deberá hacer el análisis del sistema, para poder llegar a esta etapa del programa.

Para este ciclo el software deberá recibir unos parámetros de entrada que están a disposición del diseñador del regulador, el programa da dos opciones mediante un menú desplegable:

Figura 19. Pantalla de Menú para la selección de entrada de parámetros

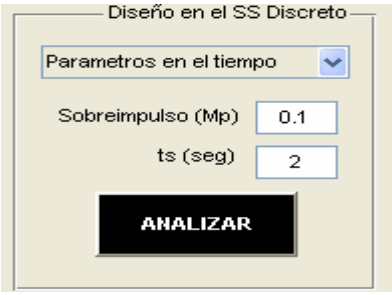


Gracias a este menú se puede contar con opciones como:

- Parámetros en el tiempo
- Polos deseados

2.5.1 Parámetros en el tiempo. Seleccionando esta opción al programa se le ingresaran dos valores importantes que son: **el sobreimpulso requerido y el tiempo de establecimiento.**

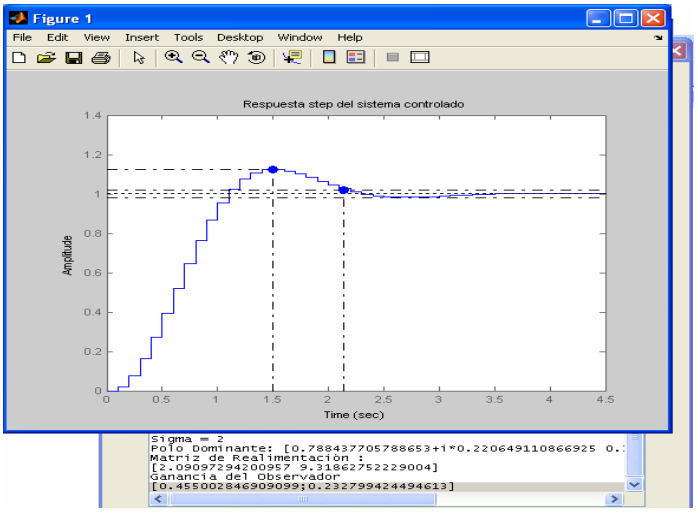
Figura 20. Pantalla de parámetros en el tiempo



Después de la inserción de estos parámetros se da **click** en "**ANALIZAR** "y el software procesara la matriz de ganancia del observador y la matriz de realimentación de estados para el sistema de control.

De esta manera la interfaz arroja como resultado las matrices antes dichas y el polo deseado junto con la respuesta paso del sistema controlado; tal cual como lo muestra la siguiente figura:

Figura 21. Respuesta paso del sistema controlado

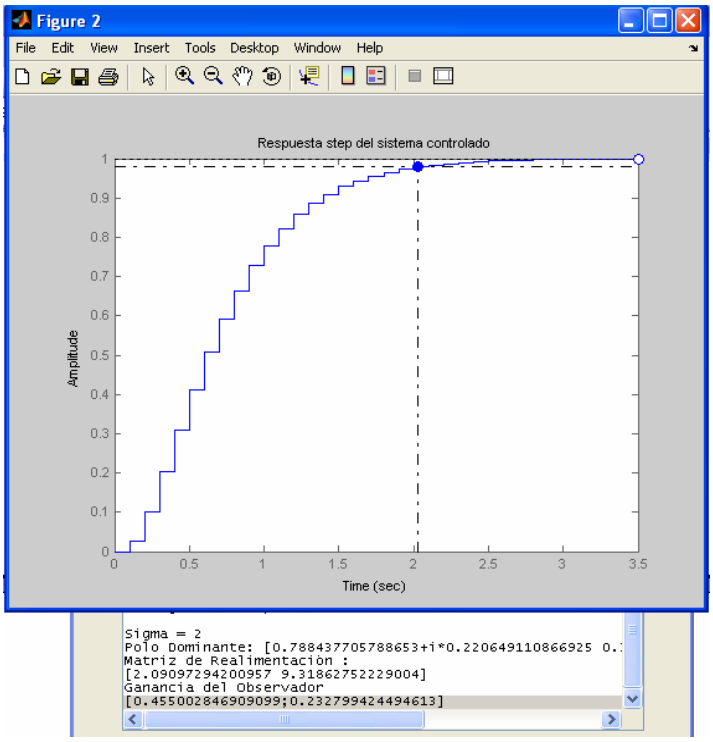


2.5.2 Polos deseados. Otra opción del programa es suministrarle los polos deseados para el cálculo del regulador.

Figura 22. Pantalla para suministro de polos deseados

En esta opción el programa valida igual que en el caso anterior los parámetros de las matrices requeridas (ganancia del observador, realimentación de estados).

Figura 23. Respuesta paso para sistema controlado con suministro de polos deseados



3. EJEMPLOS DE APLICACIÓN

Esta sección se dedica a la elaboración de algunos ejemplos típicos y su resolución de forma práctica y sencilla utilizando la interfaz.

Solo el primer ejemplo se trabaja mediante el método convencional utilizando **Matlab** y el método mediante la interfaz.

3.1 EJEMPLO 1

Con la siguiente planta se requiere una respuesta paso con un sobreimpulso **SI<=15%** , un tiempo de establecimiento de **ts=1 seg** y un tiempo de muestreo **T=0.4**.

$$\frac{2}{s\,(s+3.6)}$$

3.1.1 Método Convencional.

```
T=0.4
Gps=zpk([ ],[0 -3.6],2);
Gpss=ss(Gps);
Gpzss=c2d(Gpss,T,'zoh')
[G,H,C,D]=ssdata(Gpzss)
```

Se obtienen los siguientes vectores de espacio de estados:

```
G =
1.0000    0.2120
  0    0.2369
H =
0.1045
0.4239

C =
1     0

D =
0
```

Ahora se determina la controlabilidad y observabilidad del sistema:

```
orden=length(G)
co=ctrb(G,H);
rango=rank(co)

orden =  2

rango =  2
```

Debido a que el orden y el rango fueron iguales se argumenta que el sistema es completamente controlable.

```
ob=obsv(G,C);
rango=rank(ob)

rango =   2
```

Ahora se comprueba el rango para la observabilidad y se encuentra que son iguales con el orden de la matriz G, por lo tanto el sistema es completamente controlable.

A continuación se observan algunas gráficas de interés del sistema sin compensar y para ello se analiza la respuesta paso y la respuesta en frecuencia de la planta.

```
step(Gpzss)
margin(Gpzss)
```

Figura 24. Respuesta paso para sistema sin controlar del ejemplo 1 Método convencional

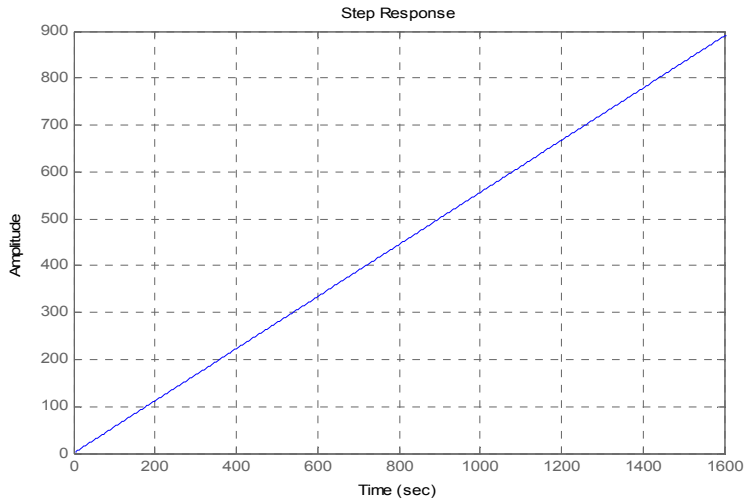
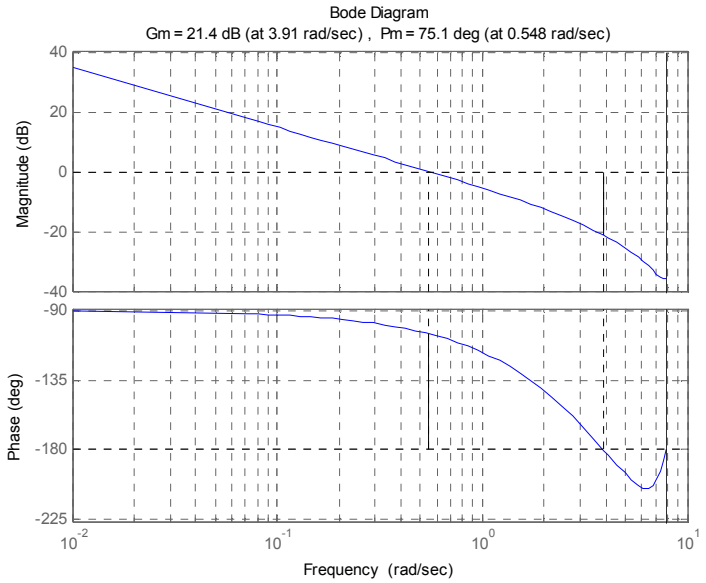


Figura 25. Respuesta en Frecuencia para el sistema sin controlar del ejemplo 1 Método convencional



Se establecen los parámetros de diseño de sobreimpulso y tiempo de establecimiento.

```
Mp=0.15;
ts=1;

sigma=4/ts;
wd=-sigma*pi/log(Mp);

Z=exp(-sigma*T)*(cos(wd*T)+sin(wd*T)*i);

P1=real(Z)+imag(Z)*i;
P2=real(Z)-imag(Z)*i;
P=[P1 P2]
```

El vector del polo deseado del sistema en lazo cerrado, es:

```
P = -0.1779 + 0.0954i  -0.1779 - 0.0954i
```

Por ultimo se encuentra la matriz de realimentación para los parámetros de diseño exigidos y la ganancia del observador de orden completo en caso de ser necesario para el sistema.

```
K=acker(G,H,P)
L=acker(G',C',P)'
```

```
Gcss=reg(Gpzss,K,L);

K =  8.2364    1.7277

L =  1.5928
     0.8549
```

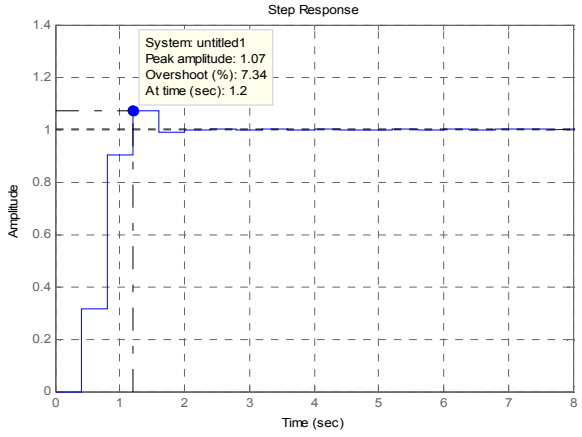
Ahora se pasa del espacio de estado del sistema a función de transferencia para poder trabajar con el límite de la función empleando el teorema del valor final y hallar la ganancia para obtener un error en estado estacionario de cero.

```
Gcz=zpk(Gcss);

Glcss=feedback(Gpzss,-Gcss);
Glcz=tf(Glcss);

syms z  Ko
[num,den]=tfdata(Glcz,'v');
Glczsym=poly2sym(num,'z')/poly2sym(den,'z');

Fz=Ko*Glczsym;

yInf=1;
y= limit(Fz,z,1)-yInf;
Ko=eval(solve(y));
```

Por último se grafica la respuesta paso del sistema compensado

```
step(Ko*Glcss)
```

Figura 26. Respuesta paso para sistema controlado del ejemplo 1 Método convencional

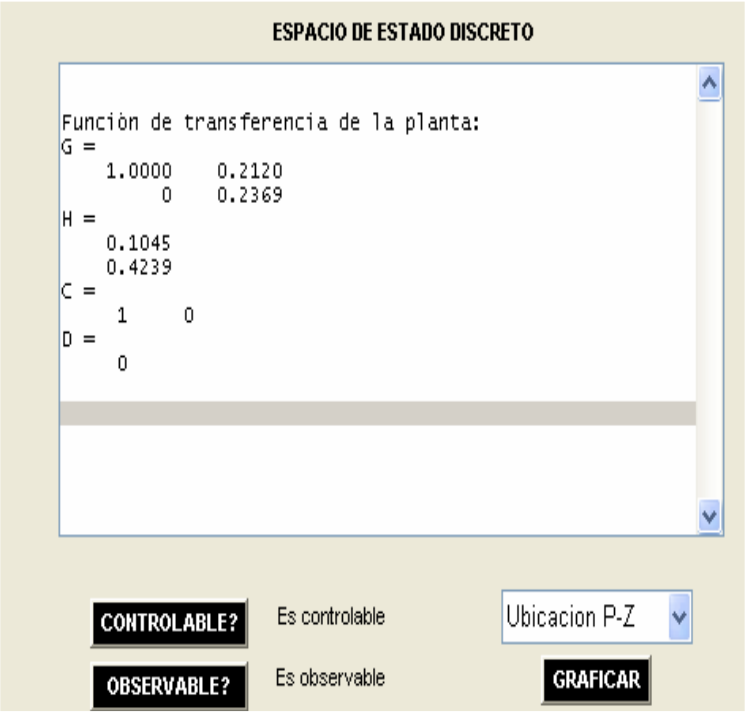


Como resultado se encuentra una sobresaliente respuesta ya que ahora se tiene un sobreimpulso del sistema controlado de solo el **7%** con un tiempo de establecimiento de **1.4** segundos, esto indica que cumple con la estimación inicial para estos parámetros.

3.1.2 Método mediante la interfaz. Para este caso se procede a discretizar la planta con un T=0.4s.

Se observa el espacio de estado en discreto, la observabilidad y controlabilidad:

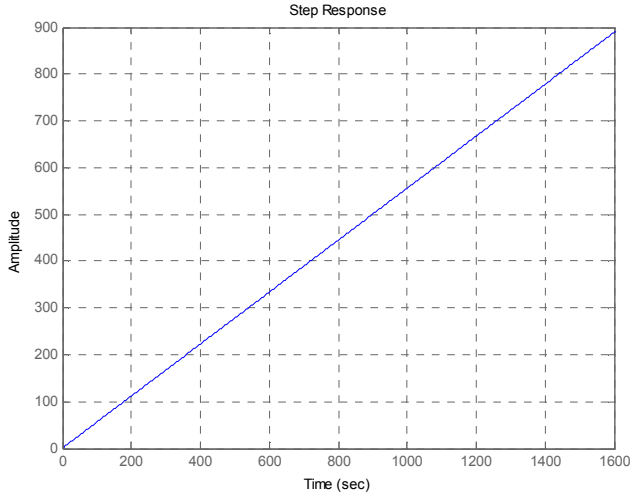
Figura 27. Pantalla para el Espacio de Estado discreto del ejemplo 1 Método de la interfaz



Con estos requisitos se observa si se puede o no seguir con el proceso de diseño para este caso, debido a que cumple con los criterios de observabilidad y controlabilidad de la planta se puede trabajar el diseño del regulador.

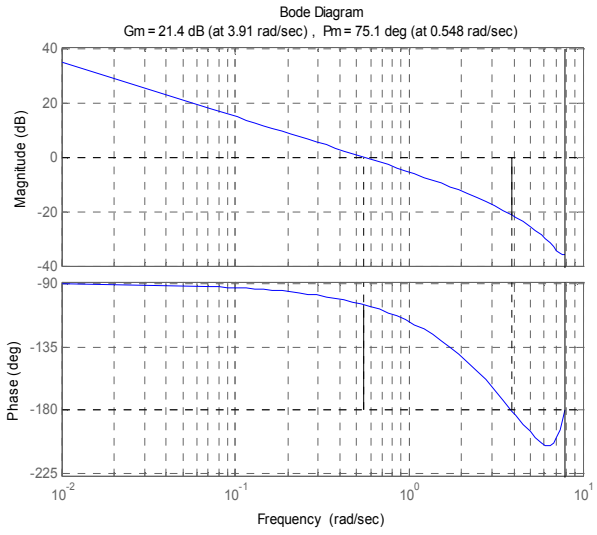
Antes de esto se observan las respuestas en el tiempo para algunas entradas, como también la respuesta en frecuencia del sistema:

Figura 28. Respuesta Paso para sistema sin controlar del ejemplo 1 Método de la interfaz



Como se puede observar para una respuesta paso el sistema se hace completamente inestable dado a los efectos del integrador.

Figura 29. Respuesta en Frecuencia para sistema sin controlar del ejemplo 1 Método de la interfaz



En la respuesta en frecuencia el sistema posee un margen de fase de 75.1

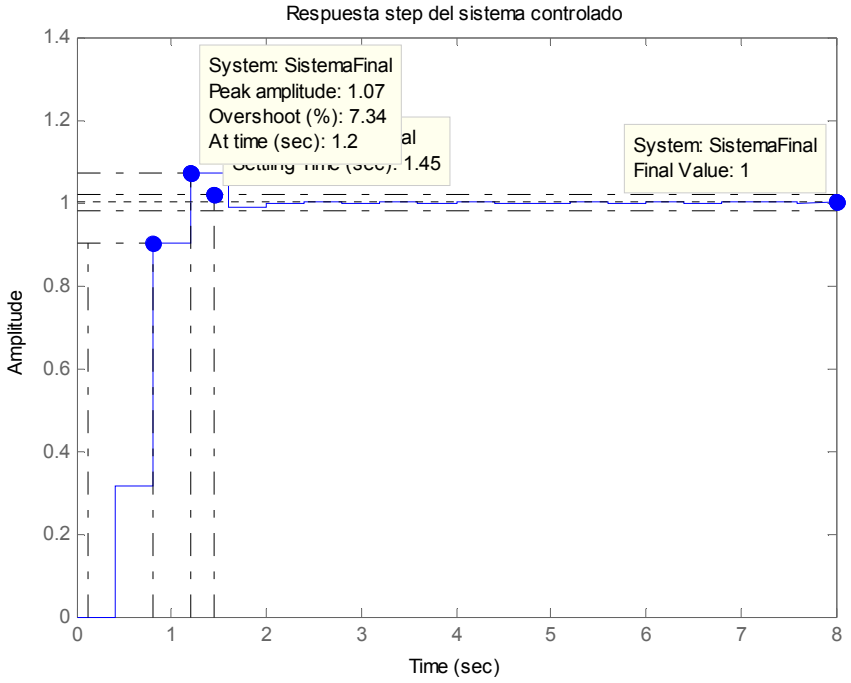
Después de introducirle los parámetros de diseño a la interfaz se obtiene:

K=[8.2364 1.7277] Matriz de realimentación

$$L=\begin{vmatrix}1.5928\\0.8549\end{vmatrix}$$ Matriz de Ganancia del observador

Se obtiene la siguiente respuesta paso del sistema controlado con el regulador:

Figura 30. Respuesta paso de sistema controlado ejemplo 1 Método de la interfaz



Un sobreimpulso de **7.34%** y tiempo de establecimiento **ts=1.45 s**, esto indica que cumple de forma satisfactoria con el requisito de diseño y que se dieron los mismos resultados por ambos métodos.

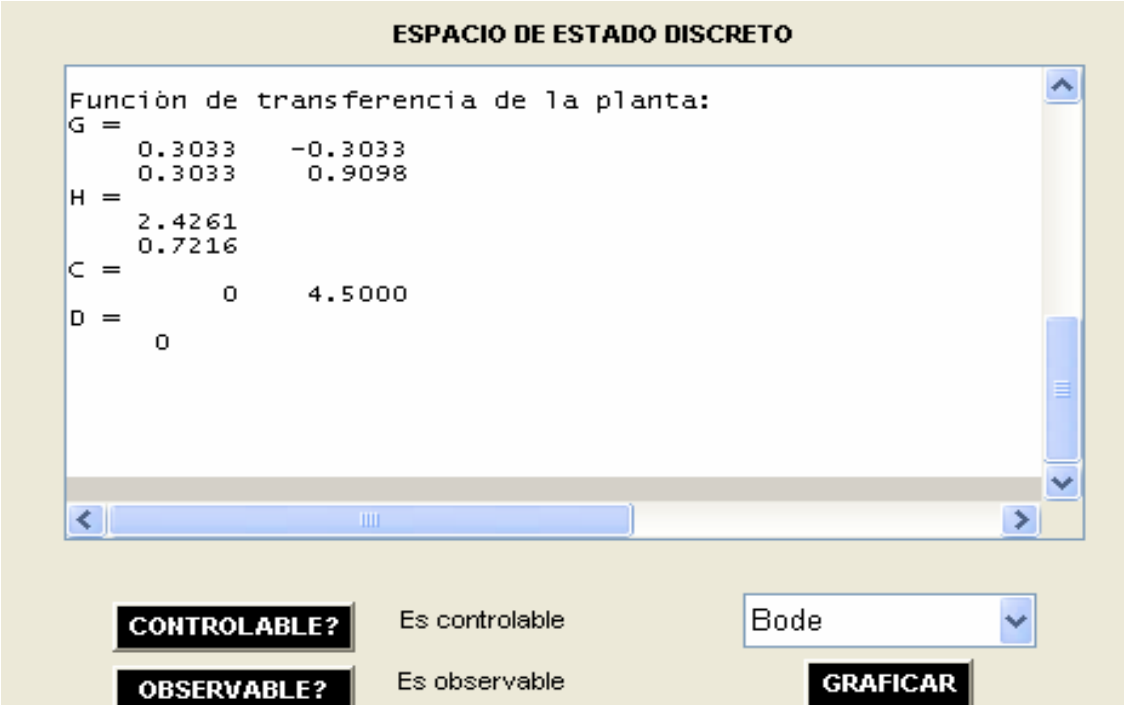
3.2 EJEMPLO 2

Se tiene la siguiente planta y se requiere para una respuesta paso un sobreimpulso **SI<=10%** y un tiempo de establecimiento **ts=1 seg.**

$$\frac{36}{s^2 + 2s + 1}$$

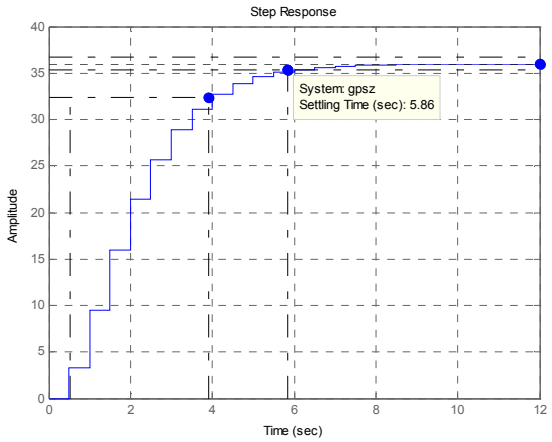
Se discretiza con un T=0.5s, se observa su espacio de estado en discreto y se evalúa la observabilidad y controlabilidad, obteniendo los siguientes resultados:

Figura 31. Pantalla para el Espacio de Estado discreto del ejemplo 2



Se observa la respuesta en el tiempo para entrada paso:

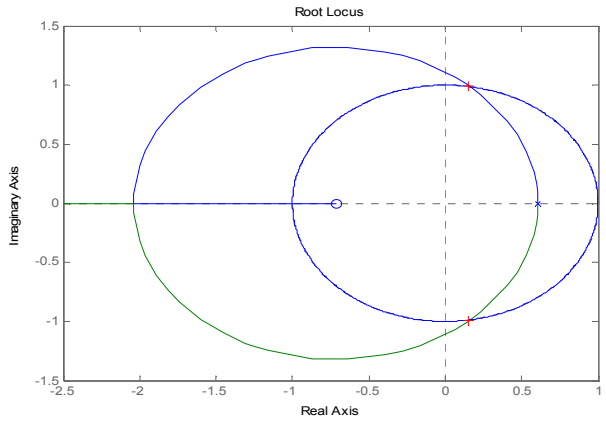
Figura 32. Respuesta paso del sistema no controlado para el ejemplo 2



Como se puede observar el sistema no es inestable en su respuesta paso, pero esta especificación en el tiempo puede ser mejorada.

Se observa a continuación la grafica de polos y ceros para verificar el r-locus.

Figura 33. Respuesta de la localización de polos y ceros para el sistema no controlado del ejemplo 2

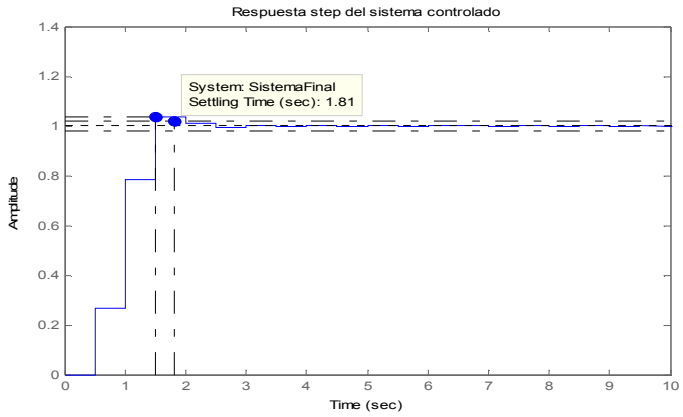


En este momento el sistema tiene una ganancia critica **Kcritica=0.2789.** Después de introducirle los parámetros de diseño a la interfaz. Se obtiene:

K=[0.3352 0.8973] Matriz de realimentación

$$L=\begin{vmatrix} 0.8851 \\ \\ 0.3246 \end{vmatrix}$$ Matriz de Ganancia del observador

Figura 34. Respuesta paso para sistema controlado del Ejemplo 2

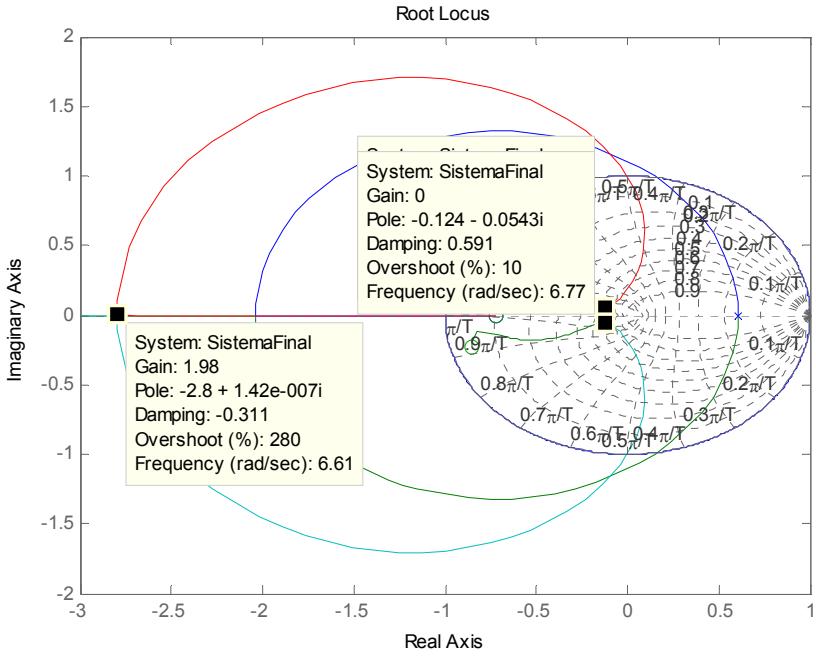


La respuesta ha mejorado notablemente en rapidez ya que ahora se tiene un tiempo de establecimiento de **ts=1.81 segundos,** sin sacrificar demasiado el sobreimpulso del sistema cumpliendo satisfactoriamente con los requisitos de diseño planteado inicialmente.

Ahora se puede observar la ubicación de polos del sistema controlado añadiendo los polos dominantes del sistema, los cuales resultaron cerca al origen para mejorar la rapidez de respuesta, efecto que se visualizo en la grafica anterior.

Gracias a la siguiente gráfica se verifica que la posición de los polos deseados sea la correcta:

Figura 35. Respuesta de la localización de polos y ceros para el sistema controlado del ejemplo 2



En el gráfico anterior se observa claramente que existen los polos dominantes en **Z=-0.124±0.0543i.**

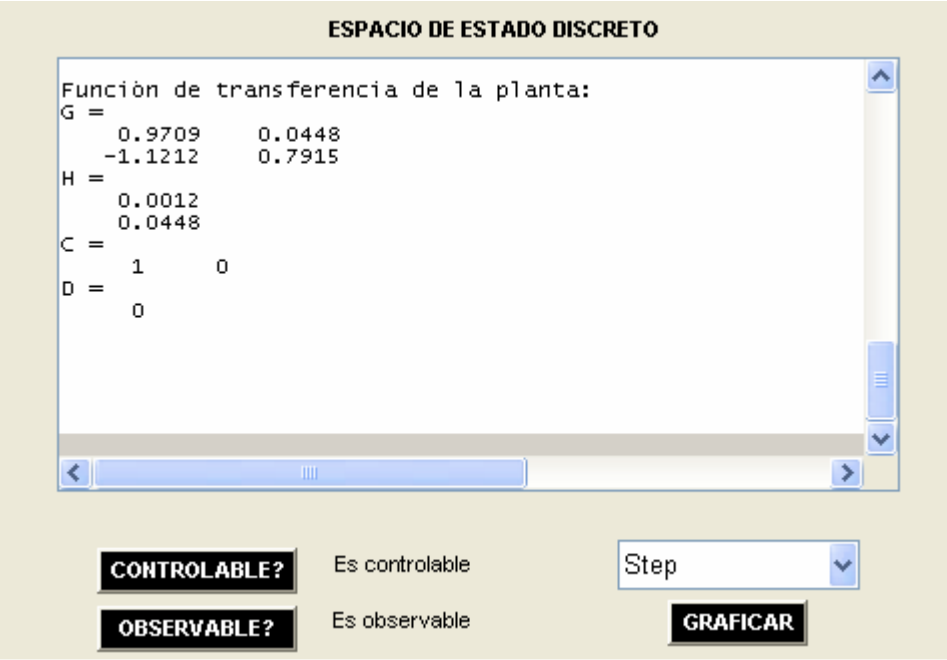
3.3 EJEMPLO 3

Se tiene la siguiente planta y se requiere una ubicación de polos dominantes en **Z=0.3 y Z=0.1**.

A=[0 1; -25 -4] B=[0 ;1] C=[1 0] D=0

Se discretiza con T=0.05seg, observando que el espacio de estado en discreto y se evalúa la observabilidad y controlabilidad, obteniendo los siguientes resultados:

Figura 36. Pantalla para el Espacio de Estado discreto del ejemplo 3



Se observa la respuesta en el tiempo para entrada paso e impulso respectivamente:

Figura 37.Respuesta paso para el sistema no controlado del ejemplo 3

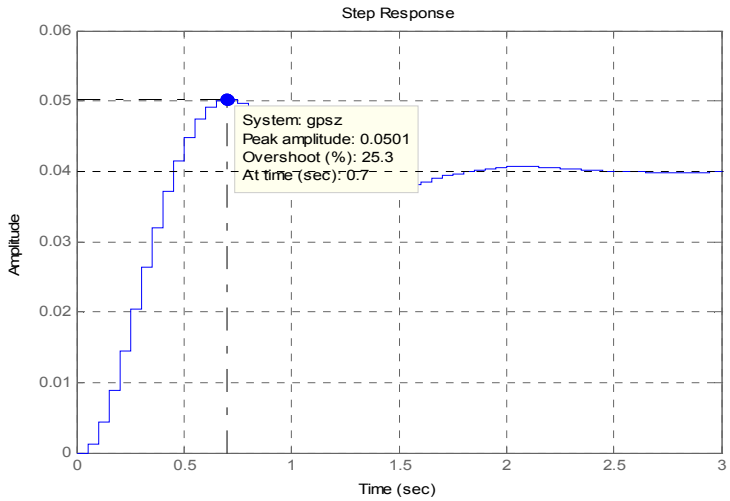
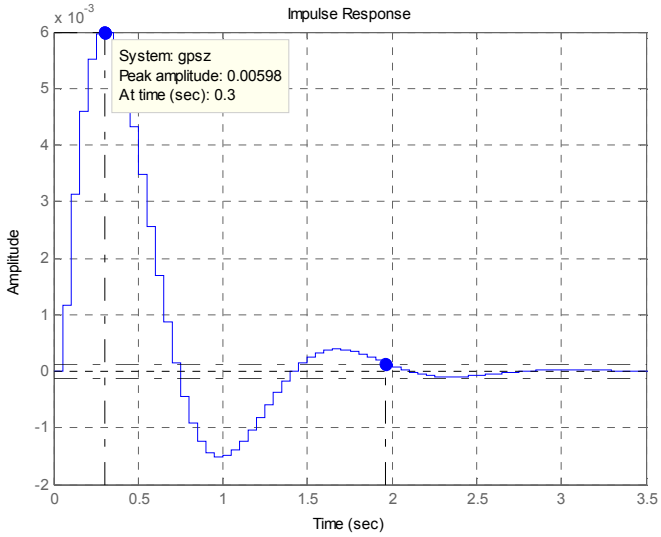


Figura 38.Respuesta impulso para el sistema no controlado del ejemplo 3



Posteriormente se introducen los parámetros de diseño a la interfaz y se obtienen los siguientes resultados:

K=[254.49 23.76] Matriz de realimentación

$$L=\begin{vmatrix} 1.3623 \\ 6.4571 \end{vmatrix}$$ Matriz de Ganancia del observador

Figura 39. Respuesta paso para el sistema controlado del ejemplo 3

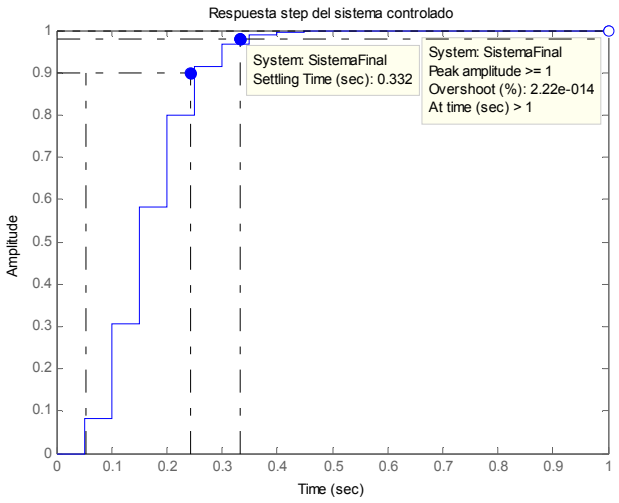
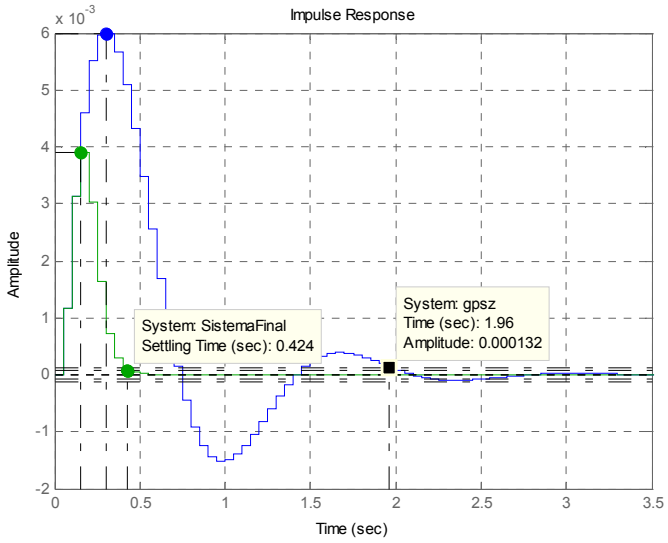
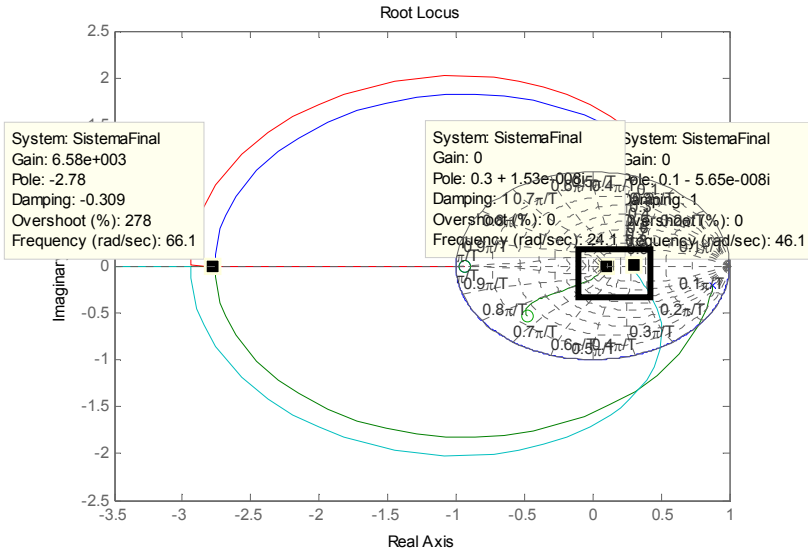


Figura 40.Respuesta impulso para el sistema controlado del ejemplo 3



Como se ve en las gráficas de paso e impulso los sobreimpulsos para ambas respuestas disminuyen notoriamente. Por último se verifica por medio de la ubicación de polos y ceros **r-locus** que los polos se encuentren tal cual se requieren.

Figura 41. Ubicación de polos y ceros para el sistema controlado del ejemplo 3



Como se muestra claramente en la gráfica el sistema final cumple con la ubicación de polos tal cual se específico inicialmente en el ejercicio, ya que se coloco dos polos en Z=0.1 y Z=0.3.

4. CONCLUSIONES

Se realizó el desarrollo de un software para las aplicaciones en diseño y análisis de reguladores en discreto por intermedio de la interfaz gráfica con ayuda del **GUI** de Matlab, de manera que el diseño de este tipo de controladores a partir de ahora es mucho mas ameno y se dedica mas tiempo al análisis y mejoramiento de las respuestas de los sistemas controlados, que en realidad al código para el procesamiento de datos y construcción de esta clase de diseños.

Es de señalar que las simulaciones hechas con esta interfaz fueron tomadas de plantas comunes y sus respuestas fueron las esperadas para los parámetros de diseño inicialmente instaurados sin la necesidad de hacer iteraciones como en casos de diseños para controladores anteriores a este; es decir, el diseño de reguladores debido a su trabajo matricial puede llegar hacer un poco más engorroso y complejo que lo anteriormente utilizado pero es mucho más exacto en su primera iteración.

Puede intuirse que el desarrollo del software aun es corto para el cubrimiento y el auge que se pretende abarcar, pero en realidad se trato de recoger los aspectos más sobresalientes y más utilizados en base a este tema y aplicarlos de la forma mas entendible y lógica posible; brindando un buen punto de partida para la implementación de futuras aplicaciones que aumenten en complejidad y robustez.

Mediante el **ejemplo 1** que se elaboró comparativamente con el método convencional demostró la confiabilidad de los resultados del software (interfaz) y las bondades de éste respecto al anterior (método convencional) ya que se hacen aplicaciones más sencillas para el usuario, sin necesidad de hacer depuración de comandos.

BIBLIOGRAFÍA

Curso Control Digital-Diseño regulador Ackerman entregado por Ing. Jorge Polonia –Clase Control Digital –Especialización Automatización Industrial. Neiva,2007

CONTROL MODERNO Pdf entregado por Ing. Jorge Polonia –Clase Control Digital –Especialización Automatización Industrial

OGATA, Katsuhiko. Ingeniería de control moderna. Tercera edición. México: Prentice – Hall, 1994.

OGATA, Katsuhiko. Sistemas de Control en Tiempo discreto. Segunda edición. México: Prentice – Hall, 1996

www.ingelec.uns.edu.ar/icd2763/tut.doc

www.prodigyweb.net.mx/saucedo8/controlIV/notas6.pdf

Anexo

Codigo gui

```
function varargout = ss_discreto(varargin)
% Hecho por: Eduard A. Rodriguez
% Last Modified by GUIDE v2.5 22-Apr-2007 00:02:39

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ss_discreto_OpeningFcn, ...
                   'gui_OutputFcn',  @ss_discreto_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before ss_discreto is made visible.
function ss_discreto_OpeningFcn(hObject, eventdata, handles, varargin)
clc
handles.output = hObject;
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = ss_discreto_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

% --- Executes during object creation, after setting all properties.
function CreateFcn(hObject, eventdata, handles)
    set(hObject,'BackgroundColor','white');

function txt_Callback(hObject, eventdata, handles)
```

```
try
   eval(['handles.data.' get(hObject,'Tag') '=[' get(hObject,'String') '];']);
   handles.error=0;
catch
   errordlg('Entrada incorrecta','vuelva a intentarlo','modal');
   handles.error=1;
end
   guidata(hObject,handles);


function actualizar_Callback(hObject, eventdata, handles)
   handles.analizado=0;
   txt_Callback(handles.campo1, eventdata, handles);
   if handles.error
      return
   end
   txt_Callback(handles.campo2, eventdata, handles);
   if get(handles.modoen,'Value')==2
      txt_Callback(handles.ganancia, eventdata, handles);
      if handles.error
         return
      end
   end

   try
      handles.data.T=eval(get(handles.S,'String'));
      handles.error=0;
   catch
      errordlg('Entrada Ts/Ws incorrecta','vuelva a intentarlo','modal');
      handles.error=1;
   end
   if handles.error
      return
   end

   try
      switch get(handles.modoen,'Value')
         case 1
            handles.data.gps=tf(handles.data.campo1,handles.data.campo2);
         case 2

handles.data.gps=zpk(handles.data.campo1,handles.data.campo2,handles.data.g
anancia);
         case 3
```

```
handles.data.gps=ss(handles.data.A,handles.data.B,handles.data.C,handles.data.
D);
        end
    catch
        errordlg('Parámetros de planta inválidos','vuelva a intentarlo','modal');
        handles.error=1;
    end
    if handles.error
        return
    end
    handles.data.gpsz=c2d(handles.data.gps,handles.data.T,'zoh');
    handles.data.gpss=ss(handles.data.gps);
    handles.data.gpzss=c2d(handles.data.gpss,handles.data.T,'zoh');

[handles.data.G,handles.data.H,handles.data.C,handles.data.D]=ssdata(handles.d
ata.gpzss);
    txtG=evalc('disp(handles.data.G)');
    txtH=evalc('disp(handles.data.H)');
    txtC=evalc('disp(handles.data.C)');
    txtD=evalc('disp(handles.data.D)');
    txt=separarenter([' ' 10 'Función de transferencia de la planta:' 10 10 'G = ' ...
        10 txtG 10 'H = ' 10 txtH 10 'C = ' 10 txtC 10 'D = ' 10 txtD]);

    actual=get(handles.listbox1,'String');

    if strcmp(class(actual),'char')
      actual={actual};
    end

    for n=1:(numel(txt)-2)
        actual{numel(actual)+1}=txt{n};
    end

    set(handles.listbox1,'String',actual,'Value',numel(actual));

    set(handles.analizar,'Enable','on');
    guidata(hObject,handles);


function salida=separarenter(texto)
salida={};
filas=numel(strfind(texto,10));
if filas
    for n=1:(filas)
```

```
        [salida{numel(salida)+1} texto]=strtok(texto,10);
        salida{numel(salida)}=[salida{numel(salida)}];
    end
    salida=salida';
else
    salida={texto};
end



function plotear_Callback(hObject, eventdata, handles)
figure();
gpsz=handles.data.gpsz;
if handles.analizado
    SistemaFinal=handles.data.SistemaFinal;
end
T=handles.data.T;

switch get(handles.tipoplot,'Value')
    case 1
        rlocus(gpsz)
        hold on
        r = 0:0.01:2*pi;
        x = sin(r);
        y = cos(r);
        plot(x,y)
        if handles.analizado
            rlocus(SistemaFinal)
        else
            [K,P] = rlocfind(gpsz);
            Mag = abs(P);
            Mensaje=['La ganancia Crítica es: ' num2str(K) 10 ...
                    'El polo seleccionado es: ' num2str(P(1)) 10 ...
                    'La Magnitud del polo deseado es: ' num2str(Mag(1)) 10 ' ' 10];
            txt=separarenter(Mensaje);


            actual=get(handles.listbox1,'String');
            for n=1:numel(txt)
                actual{numel(actual)+1}=txt{n};
            end
            set(handles.listbox1,'Value',numel(actual),'String',actual);

            msgbox(Mensaje);
        end
```

```
        grid on
    case 2
        Ws = 2*pi/T;
        w = [0.1:1:Ws];
            margin(gpsz);  % antes tenía margin (gpsz,w)
            if handles.analizado
                hold on
                margin(SistemaFinal);
            end
        grid on
    case 3
        step(gpsz);
        if handles.analizado
            hold on
            step(SistemaFinal);
        end
        grid on
    case 4
        impulse(gpsz);
        if handles.analizado
            hold on
            impulse(SistemaFinal);
        end
        grid on
end


function ctrl_Callback(hObject, eventdata, handles)
    orden=length(handles.data.G);
    co=ctrb(handles.data.G,handles.data.H);
    rango=rank(co);
    if rango==orden
        set(handles.txtctrl,'String','Es controlable')
    else
        set(handles.txtctrl,'String','No es controlable')
    end
    handles.data.ctrl=(rango==orden);
    guidata(hObject,handles);


function obsv_Callback(hObject, eventdata, handles)
    orden=length(handles.data.G);
    ob=obsv(handles.data.G,handles.data.C);
    rango=rank(ob);
    if rango==orden
```

```
    set(handles.txtobs,'String','Es observable')
  else
    set(handles.txtobs,'String','No es observable')
  end
  handles.data.obs=(rango==orden);
  guidata(hObject,handles);



% --- Executes on button press in analizar.
function analizar_Callback(hObject, eventdata, handles)
  handles.analizado=0;
  if ~(handles.data.ctrl&&handles.data.obs)
    errordlg('El sistema debe ser controlable y observable','Error');
    return
  end

  try
    handles.data.mp=eval(get(handles.mp,'String'));
    handles.data.ts=eval(get(handles.ts,'String'));
    handles.data.T=eval(get(handles.S,'String'));
    handles.error=0;
  catch
    errordlg('Entrada Ts/Ws incorrecta','vuelva a intentarlo','modal');
    return
  end
  guidata(hObject,handles);
  if get(handles.modoanalisis,'Value')==1
    mp=handles.data.mp;
    ts=handles.data.ts;
  end
  gpzss=handles.data.gpzss;
  T=handles.data.T;
  G=handles.data.G;
  H=handles.data.H;
  C=handles.data.C;


  disp('CACULAMOS EL SIGMA CON UNA TOLERANCIA DEL 2% ');
  if get(handles.modoanalisis,'Value')==1
    sigma=4/ts(1)
    wd=-sigma*pi/log(mp);
    Z=exp(-sigma*T)*(cos(wd*T)+sin(wd*T)*i);
    disp('NUESTRO POLO DOMINANTE ES: ');
```

```
   P=[real(Z)+imag(Z)*i real(Z)-imag(Z)*i]
   a=size(G);
   a=a(1);
   a=a-numel(P);
   if a
     P=[P ones(1,a)*.2]
   end
   K=acker(G,H,P)              % Matriz de realimetación
   L=acker(G',C',P)'           % Ganancia del observador
   txt={['Sigma = ' num2str(sigma)], ...
   ['Polo Dominante: ' mat2str(P)],...
   ['Matriz de Realimentación :'],...
   mat2str(K),['Ganancia del Observador'],...
   mat2str(L)};
else
   P=handles.data.polos;
   try
      K=acker(G,H,P)           % Matriz de realimetación
      L=acker(G',C',P)'        % Ganancia del observador
   catch
      errordlg(['Para usar acker la cantidad de polos debe ser igual a' ...
         ' la cantidad de estados.']);
   end
   txt={['Polo Dominante: ' mat2str(P)],...
   ['Matriz de Realimentación :'],...
   mat2str(K),['Ganancia del Observador'],...
   mat2str(L)};
end


actual=get(handles.listbox1,'String');
for n=1:numel(txt)
   actual{numel(actual)+1}=txt{n};
end
set(handles.listbox1,'String',actual,'Value',numel(actual));



Gcss=reg(gpzss,K,L);     % controlador en ecuaciones de estado

Gcz=zpk(Gcss);

Glczss=feedback(gpzss,-Gcss);
Glcz=tf(Glczss);
```

```
        % Obtencion de la ganancia Ko :
        % Ko se ajusta para que la respuesta al paso en estado
        % estacionario sea 1, estoes y(inf)=1 , Ess=0

    syms z  Ko
    [num,den]=tfdata(Glcz,'v');
    Glczsym=poly2sym(num,'z')/poly2sym(den,'z');

    Fz=Ko*Glczsym;

    yInf=1;
    y= limit(Fz,z,1)-yInf;
    Ko=eval(solve(y));

    figure();
    SistemaFinal=Ko*Glczss;
    step(SistemaFinal)       % Respuesta al paso
    title('Respuesta step del sistema controlado');
    handles.data.SistemaFinal=Glczss;
    handles.analizado=1;

[handles.data.G2,handles.data.H2,handles.data.C2,handles.data.D2]=ssdata(Glcz
ss);
    txtG=evalc('disp(handles.data.G2)');
    txtH=evalc('disp(handles.data.H2)');
    txtC=evalc('disp(handles.data.C2)');
    txtD=evalc('disp(handles.data.D2)');
    txt=separarenter([' ' 10 'Función de transferencia de la planta:' 10 10 'G = ' ...
        10 txtG 10 'H = ' 10 txtH 10 'C = ' 10 txtC 10 'D = ' 10 txtD]);

    actual=get(handles.listbox1,'String');
    if strcmp(class(actual),'char')
      actual={actual};
    end

    for n=1:numel(txt)
       actual{numel(actual)+1}=txt{n};
    end
    guidata(handles.analizar,handles);

% --- Executes on selection change in modoanalisis.
function modoanalisis_Callback(hObject, eventdata, handles)
    h1=[handles.mpt handles.tst handles.mp handles.ts]; h2=[handles.polos
handles.polost];
```

```
    v={'on','off'}; v2={'off','on'};
    set(h1,'visible',v{get(hObject,'Value')});
    set(h2,'visible',v2{get(hObject,'Value')});


% --- Executes during object creation, after setting all properties.
function modoanalisis_CreateFcn(hObject, eventdata, handles)



% --- Executes on selection change in modoen.
function modoen_Callback(hObject, eventdata, handles)
    handles.data.modo=get(hObject,'Value'); %1=numden, 2=zpk, 3=ss
    modo3=[handles.A handles.taga handles.B handles.tagb handles.C
handles.tagc handles.D handles.tagd];
    modo12=[handles.campo1 handles.tagnum handles.campo2 handles.tagden
handles.ganancia handles.taggan];
    set(modo12,'visible','on');
    set(modo3,'visible','on');
    switch handles.data.modo
        case 1 %numden
            set(handles.tagnum,'String','Numerador');
            set(handles.tagden,'String','Denominador');
            set([handles.taggan handles.ganancia],'Visible','Off')
            set(modo3,'visible','off');
        case 2 %zpk
            set(handles.tagnum,'String','Ceros (Z)');
            set(handles.tagden,'String','Polos (P)');
            set(modo3,'visible','off');
        case 3 %ss
            set(modo12,'visible','off');
    end
```